Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik
Lehrstuhl für Kognitive Systeme

Efficient Realization of Large-Area E-Skin based on Biologically Plausible Principles

Florian Bergner

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:

Prof. Dr.-Ing. Wolfgang Utschick

Prüfer der Dissertation:

1. Prof. Gordon Cheng, Ph.D.
2. Prof. Takao Someya, Ph.D.

Die Dissertation wurde am 11.11.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 19.03.2021 angenommen.

Abstract

Skin endows us, humans, with the capability to interact with our surroundings. Skin, with its millions of tactile receptors on the large area of the human body surface, provides, with the sense of touch, essential feedback. The feedback that allows us, humans, to control the way we physically interact with surroundings, with objects, with other living beings. Despite the skin's importance to provide feedback for interactions, few autonomous machines are equipped with the sense of touch. This is mainly due to the challenge to efficiently handle a high amount of multi-modal tactile information that the huge number of distributed sensors in a large-area electronic skin system produces. This thesis introduces novel methods based on biologically plausible principles to efficiently handle tactile information and realize efficient large-area electronic skin (e-skin). These methods employ the efficient principle of handling information according to its novelty. In the novelty-driven large-area e-skin, information is only sensed, transmitted, and computed when the information is new. The novelty-driven system gains its efficiency by a significant reduction of transmission rates and demands on computational power without dropping important information. The advantage of the novelty-driven approach presented in this thesis is hardware independence, efficiency, scalability, applicability to existing e-skins, and effectiveness in very complex systems. Starting from the biologically plausible principle of novelty, this thesis homogenizes the different descriptions and points of view from different research fields. This leads to a formal and consistent theory and foundations, forming the fundamentals for novelty-driven systems. These fundamentals allow for the correct design and optimal parameterization of novelty detectors for the e-skin's sensors to optimize for sensitivity, transmission rate, minimizing encoding errors, and noise. Furthermore, they lead to a model for extrapolating the computational demand of novelty-driven e-skin systems. Additionally, design methods for the systematic realization of the novelty-driven approach in existing e-skins and standard computer systems were developed in this thesis. The subsequent realization of these designs in an e-skin system with more than 10 000 multi-modal tactile sensors validates their feasibility. The experimental evaluation of the large-area e-skin system presented in this thesis demonstrates the efficiency of the novelty-driven approach in e-skin systems. The approach significantly boosts performance by reducing transmission rates and computational demands. This improvement, for the first time, allows the effective handling of tactile information of a large-area e-skin with more than 10 000 sensors. Providing the feedback of large contact areas, the integration of novelty-driven large-area e-skin in robotic systems shows its efficiency and effectiveness in complex reactive control algorithms for physical interactions. Furthermore, experiments with a humanoid robot demonstrate the system's efficiency to scale, whereby previous approaches failed, allowing very challenging utilization of large-area e-skin in whole-body controllers.

Kurzfassung

Haut verleiht uns Menschen die Fähigkeit, mit ihrer Umgebung zu interagieren. Mit ihren Millionen taktilen Rezeptoren auf der gesamten Oberfläche des menschlichen Körpers stellt sie über den Tastsinn unverzichtbare Feedbackinformationen zur Verfügung. Dieses Feedback gibt den Menschen die Möglichkeit, den Kontakt mit der Umgebung, mit Objekten und anderen Lebewesen zu steuern. Trotz dieser wichtigen Funktion der Haut, bei Interaktionen ein Feedback zu liefern, sind nur wenige autonome Maschinen, wie z.B. Roboter, mit einem vergleichbaren Tastsinn ausgestattet. Dies liegt hauptsächlich in der großen Herausforderung, die umfangreichen taktilen Informationen, die von einer großen Anzahl verteilter Sensoren eines großflächigen, elektronischen Hautsystems erzeugt werden, effizient zu verarbeiten. Diese Dissertation stellt deshalb neuartige, auf biologisch plausiblen Prinzipien beruhende Methoden vor, um taktile Informationen effizient zu verarbeiten und um eine effiziente, großflächige elektronische Haut (E-Skin) zu realisieren. Diese Methoden nutzen das effiziente Prinzip, Informationen nur entsprechend ihrem Neuheitsgrad zu behandeln. In einer von der Neuheit der Information gesteuerten, großflächigen E-Skin wird eine Information nur erkannt, übermittelt und weiter verarbeitet, wenn sie sich geändert hat und es sich somit um eine neue Information handelt. Das neuheitsgesteuerte System erzielt seine Effizienz durch eine erhebliche Reduzierung der Übertragungsrate und der zur Verarbeitung erforderlichen Rechenleistung. Die Vorteile, der in dieser Dissertation vorgestellten neuheitsgesteuerten Verfahren sind die Unabhängigkeit von der Hardware, die Effizienz, die Skalierbarkeit, die Anwendbarkeit bei vorhandenen E-Skins, sowie die Effektivität in sehr komplexen Systemen. Ausgehend vom biologisch plausiblen Prinzip, Informationen entsprechend ihrer Neuheit zu behandeln, vereinheitlicht die Dissertation unterschiedliche Beschreibungen und Sichtweisen verschiedener betroffener Forschungsgebiete. Daraus entstehen eine konsistente Theorie und konzeptuelle Grundlagen, welche die Basis für neuheitsgesteuerte Systeme bilden. Diese Grundlagen ermöglichen das richtige Design und eine optimale Parametrisierung von Neuheitsdetektoren für die E-Skin Sensoren, um dadurch die Empfindlichkeit, die Übertragungsrate, die Minimierung von Kodierungsfehlern und das Rauschen zu optimieren. Darüber hinaus wird aus diesen Grundlagen ein Modell zur Ermittlung des Rechenbedarfs neuheitsgesteuerte E-Skin Systeme abgeleitet. Zusätzlich werden in dieser Dissertation Designmethoden für die systematische Umsetzung der neuheitsgesteuerten Vorgehensweise bei vorhandenen E-Skins und Standardcomputersystemen entwickelt. Die anschließende Realisierung dieser Designs in einem E-Skin System mit über 10 000 multimodalen taktilen Sensoren bestätigt deren Umsetzbarkeit. Eine experimentelle Beurteilung des in dieser Dissertation präsentierten, großflächigen E-Skin Systems beweist die Effizienz des neuheitsgesteuerten Verfahrens in E-Skin Systemen. Dieser Ansatz steigert die Leistung des Systems deutlich durch reduzierte Übertragungsraten und gesenkte Anforderungen an die Rechenleistung. Die erzielte Verbesserung ermöglicht zum ersten Mal die effektive Verarbeitung taktiler Informationen einer großflächigen E-Skin mit mehr als 10 000 Sensoren. Mit der Bereitstellung von Feedback aus großen Kontaktbereichen zeigt die Integration der neuheitsgesteuerten, großflächigen E-Skin

in Robotersystemen ihre Effizienz und Effektivität in komplexen, reaktiven Steuerungsalgorithmen für Interaktionen. Darüber hinaus verdeutlichen Experimente mit einem humanoiden Roboter die Effizienz des Systems für Größenordnungen, bei denen vorhergehende Ansätze gescheitert sind, wodurch sehr anspruchsvolle Anwendungen großflächiger E-Skin Systeme für Ganzkörpersteuerungen möglich sind.

Acknowledgements

This thesis would not have been possible without the kind support and help of all the people who accompanied me along this journey. Creative work is only possible in an open environment that invites to discussions, fosters the exchange of views and ideas, and lives from team spirit and solidarity. Many thanks to all of you for offering such an environment and letting me be a part of it.

First of all, I would like to express my deep gratitude to my advisor **Prof. Gordon Cheng**, who kindly gave me the opportunity to be part of his team and carry out my research as a doctoral candidate. I especially thank him for the inspiration and encouragement he gave me, for his great support, patience, motivation, and guidance. All this has been invaluable in learning for life and for the success of this thesis.

Many thanks go to my family and friends. Without their support, patience, and tolerance for academic working hours this thesis would not have been possible. I'm deeply grateful to my mother, father, and sister for their unlimited support and for keeping my life in balance.

Furthermore, I would like to express my deep gratitude to my mentor **Dr. Emmanuel Dean-Leon** for his great support in all the challenges I encountered in this journey on the professional but also on the personal level. Thank you for all the long technical and philosophical discussions, the exchange of ideas, the introduction of different points of view, and your guidance. Thank you also for teaching me robotics and control.

My special thanks also go to **Prof. Takao Someya** for many inspiring discussions and exchanges of ideas, and for introducing me into the very interesting field of flexible and stretchable sensing and electronics. I'm deeply grateful that he gave me the opportunity to stay for two months in his Organic Transistor Lab at the University of Tokyo, Japan and work with his research group. I'm very thankful for many fruitful discussions with his group members and the support and insights they kindly provided.

I'm much obliged for all the help of my former and present colleagues at the Chair for Cognitive Systems. Thank you for all the support, inspiring discussions, and a perfect working environment: Simon Armleder, Nicolas Berberich, Dr. Emmanuel Dean-Leon, Dr. Stefan Ehrlich, Julio Rogelio Guadarrama-Olvera, Prof. Pablo Lanillos, Quentin Leboutet, Dr. Philipp Mittendorfer, Prof. Karinne Ramirez-Amaro, Dr. Christoph Söllner, Constantin Uhde, Prof. Agnieszka Wykowska. Many thanks also to the administration team for all the patience and support with complex administrative processes: Wibke Borngesser, Ilona Nar-Witte, Brigitte Rosenlehner. My special thanks go to the technicians **Katharina Stadler** and **Sebastian Stenner** for all their patience with unconventional ideas, and their technical expertise and

support. I also thank the chair's countless visitors for inspiring discussions and their different points of view.

I would never have had the idea and courage to undertake this journey without the inspiring influence of my past professors: **Prof. Klaus Diepold**, **Prof. Josef A. Nossek**, **Prof. Ulf Schlichtmann**, **Prof. Wolfgang Utschick**. Therefore I'm very grateful.

Furthermore, I thank all my thesis proof readers for their patience and support to improve the quality of this thesis. Many thanks for the keen eye on typos, wording issues, grammar errors, weak descriptions, and many more: Simon Armleder, Nicolas Berberich, Peter Bergner, Wibke Borngesser, Dr. Emmanuel Dean-Leon, Julio Rogelio Guadarrama-Olvera, Quentin Leboutet, Prof. Karinne Ramirez-Amaro, Katrin Schulleri, Andreas Wagner.

At last, I would like to thank the funding sources that made this thesis possible, in particular, the European FP7 Project Factory-in-a-Day, the Deutsche Forschungsgemeinschaft (DFG), the TUM Institute for Advanced Study (IAS), and the TUM Faculty Graduate Center for Electrical and Computer Engineering (FGZ-EI).

List of Publications

Parts of this thesis have already been published in:

Journals

J1 Kobayashi, T., Dean-Leon, E., Guadarrama-Olvera, J. R., **Bergner, F.**, Cheng, G., "Whole-Body Multicontact Haptic Human–Humanoid Interaction Based on Leader–Follower Switching: A Robot Dance of the "Box Step"". In: *Advanced Intelligent Systems* (2021), p. 2100038.

J2 **Bergner, F.**, Dean-Leon, E., Cheng, G., "Design and Realization of an Efficient Large-Area Event-Driven E-Skin". In: *Sensors* 20.7 (2020), p. 1965.

J3 Cheng, G., Dean-Leon, E., **Bergner, F.**, Guadarrama-Olvera, J. R., Leboutet, Q., Mittendorfer, P., "A comprehensive realisation of Robot Skin: Sensors, Sensing, Control and Applications". In: *Proceedings of the IEEE* 107.10 (2019), pp. 2034–2051.

J4 **Bergner, F.**, Dean-Leon, E., Guadarrama-Olvera, J. R., Cheng, G., "Evaluation of a Large Scale Event Driven Robot Skin". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4247–4254.

J5 Dean-Leon, E., Ramirez-Amaro, K., **Bergner, F.**, Cheng, G., "Robot Skin: Fully-Compliant Control Framework Using Multi-modal Tactile Events". In: *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI* 7 (2019), pp. 4–13.

J6 Guadarrama Olvera, J. R., Dean Leon, E., **Bergner, F.**, Cheng, G., "Plantar Tactile Feedback For Biped Balance and Locomotion on Unknown Terrain". In: *International Journal of Humanoid Robotics* 17.1 (2019), p. 1950036.

J7 Guadarrama-Olvera, J. R., Dean-Leon, E., **Bergner, F.**, Cheng, G., "Pressure-Driven Body Compliance Using Robot Skin". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4418–4423.

J8 Leboutet, Q., Dean-Leon, E., **Bergner, F.**, Cheng, G., "Tactile-Based Whole-Body Compliance With Force Propagation for Mobile Manipulators". In: *IEEE Transactions on Robotics* 35.2 (2019), pp. 330–342.

J9 Ramirez-Amaro, K., Dean-Leon, E., **Bergner, F.**, Cheng, G., "A Semantic-Based Method for Teaching Industrial Robots New Tasks". In: *KI-Künstliche Intelligenz* 33.2 (2019), pp. 117–122.

J10 Dean-Leon, E., Ramirez-Amaro, K., **Bergner, F.**, Dianov, I., Cheng, G., "Integration of Robotic Technologies for Rapidly Deployable Robots". In: *IEEE Transactions on Industrial Informatics* 14.4 (2018), pp. 1691–1700.

Book chapters

B1 **Bergner, F.**, Cheng, G., “Sensory Systems for Robotic Applications – Making sense of the world”. In: ed. by Ravinder S. Dahiya, Oliver Ozioko, and Gordon Cheng. The Institution of Engineering and Technology, 2020, Submitted. Chap. Neuromorphic Principles for Large-Scale Robot Skin.

Peer-Reviewed Conference Papers

C1 Leboutet, Q., **Bergner, F.**, Cheng, G., “Online Adaptive Configuration Selection for Redundant Arrays of Inertial Sensors: Application to Robotic Systems Covered with a Multimodal Artificial Skin”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020.

C2 Leboutet, Q., Guadarrama-Olvera, J. R., **Bergner, F.**, Cheng, G., “Second-order Kinematics for Floating-base Robots using the Redundant Acceleration Feedback of an Artificial Sensory Skin”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020.

C3 Dean-Leon, E., Guadarrama-Olvera, J. R., **Bergner, F.**, Cheng, G., “Whole-Body Active Compliance Control for Humanoid Robots with Robot Skin”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Montreal, Canada, 2019, pp. 5404–1510.

C4 Kobayashi, T., Dean-Leon, E., Guadarrama-Olvera, J. R., **Bergner, F.**, Cheng, G., “Multi-Contacts Force-Reactive Walking Control during Physical Human-Humanoid Interaction”. In: *IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*. 2019, pp. 33–39.

C5 Deistler, M., Yener, Y., **Bergner, F.**, Lanillos, P., Cheng, G., “Tactile Hallucinations on Artificial Skin Induced by Homeostasis in a Deep Boltzmann Machine”. In: *IEEE International Conference on Cyborg and Bionic Systems*. 2019.

C6 **Bergner, F.**, Dean-Leon, E., Cheng, G., “Efficient Distributed Torque Computation for Large Scale Robot Skin”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, 2018, pp. 1593–1599.

C7 Bader, C., **Bergner, F.**, Cheng, G., “A Robust and Efficient Dynamic Network Protocol for a large-scale artificial robotic skin”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, 2018, pp. 1600–1605.

C8 Guadarrama-Olvera, J. R., **Bergner, F.**, Dean-Leon, E., Cheng, G., “Enhancing Biped Locomotion on Unknown Terrain Using Tactile Feedback”. In: *International Conference on Humanoid Robots (Humanoids)*. Beijing, China, 2018, pp. 47–52.

C9 **Bergner, F.**, Dean-Leon, E., Cheng, G., “Efficient Event-Driven Reactive Control for Large Scale Robot Skin”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Singapore, Singapore, 2017, pp. 394–400.

C10 Dean-Leon, E., Pierce, B., Mittendorfer, P., **Bergner, F.**, Ramirez-Amaro, K., Burger, W., Cheng, G., “TOMM: Tactile Omnidirectional Mobile Manipulator”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Singapore, Singapore, 2017, pp. 2441–2447.

C11 **Bergner, F.**, Dean-Leon, E., Cheng, G., “Event-based signaling for large-scale artificial robotic skin - Realization and performance evaluation”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, Korea, 2016, pp. 4918–4924.

C12 Dean-Leon, E., **Bergner, F.**, Ramirez-Amaro, K., Cheng, G., “From multi-modal tactile signals to a compliant control”. In: *IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. 2016, pp. 892–898.

C13 Dean-Leon, E., Ramirez-Amaro, K., **Bergner, F.**, Dianov, I., Lanillos, P., Cheng, G., “Robotic Technologies for Fast Deployment of Industrial Robot Systems”. In: *The 42nd Annual Conference of IEEE Industrial Electronics Society (IECON)*. 2016, pp. 6900–6907.

C14 Ramirez-Amaro, K., Dean-Leon, E., Dianov, I., **Bergner, F.**, Cheng, G., “General recognition models capable of integrating multiple sensors for different domains”. In: *IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. 2016, pp. 306–311.

C15 Dianov, I., Ramirez-Amaro, K., Lanillos, P., Dean-Leon, E., **Bergner, F.**, Cheng, G., “Extracting general task structures to accelerate the learning of new tasks”. In: *IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. 2016, pp. 802–807.

C16 **Bergner, F.**, Mittendorfer, P., Dean-Leon, E., Cheng, G., “Event-based signaling for reducing required data rates and processing power in a large-scale artificial robotic skin”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. Hamburg, Germany, 2015, pp. 2124–2129.

Abstracts

A1 Hoxha, I., Del Duca, F., Ehrlich, S. K., **Bergner, F.**, Berberich, N., Cheng, G., “Improving user comfort in auditory-steady-state-response brain-computer interface by using a co-adaptive stimulus”. In: *Federation of European Neuroscience Societies (FENS)*. 2020.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

A

ACES - Asynchronous Encoded Skin 1, 25, 27, 30
ADC - Analog-Digital Converter 1, 40–42, 49, 50
AER - Address-Event-Representation 1, 25–27, 30, 58–61
AWGN - Additive White Gaussian Noise 1, 51, 99

B

BWLS - Band-Width Limited System 1, 45, 46

C

CDS - Clock-Driven System 1, 10, 21, 22, 24, 26, 28, 30, 74, 77, 79–81, 94, 132, 143
CNS - Central Nervous System 1, 12
CORDIC - Coordinate Rotation Digital Computer 1, 160
CPU - Central Processing Unit 1, 21, 70, 93–96

D

DMA - Direct Memory Access 1, 87
DOF - Degrees Of Freedom 1, 140, 156, 158, 189

E

ECC - Error Correction Code 1, 60
ED - Event-Driven 1
EDS - Event-Driven System 1, 10, 13, 22–26, 28, 30, 74, 80, 94, 98, 144–146, 148
EOF - End of Frame 1, 60, 88
e-skin - electronic skin ..... ii, 1–4, 10, 14–17, 31

F

FPGA - Field Programmable Gate Array 1, 21
FPU - Floating Point Unit 1, 158, 159
FT - Force-Torque 1, 192

G

GPU - Graphics Processing Unit 1, 21

H

HT - Hyper Threading, Intel 1

I

IPG - Interpacket Gap 1, 87, 200
IPS - Instructions Per Second 1, 63
IR - Infrared 1, 16

L

M

N

O

P

R

S

T

U

1. Introduction

1.1. Motivation

The sense of touch plays an essential role in our daily life. While most people immediately understand how dependent we humans are on perceiving our world using our eyes, ears, nose, and tongue, people often forget about their sense of touch and its importance in our everyday life. The sense of touch endows us with the ability to perceive and control our contacts with our surroundings, other living beings, and objects. It provides us with essential feedback – feedback on contact structure (soft/hard, smooth/rough), contact force (magnitude/directions), contact shapes, contact temperature, contact friction (slippery/sticky), that is, feedback on contacts which make interactions secure. This feedback lets us safely master insecure, slippery, and fragile contacts. It prevents us from hurting ourselves, other people, or damaging objects. It is surprising how much we humans depend on interaction and how many benefits the sense of touch provides us.

For example, walking is a process where we continuously interact with the floor, a process that highly depends on the tactile feedback describing when and how we touch the floor. Without this essential feedback, the process of walking is very insecure. We would tread too hard or misjudge the reliability of the tread and slip. Other examples for interactions are the lifting and carrying of bulky objects, sitting on a chair, climbing up a wall, or negotiating in narrow spaces. The sense of touch is the key to all these interactions. It provides feedback on where and how contacts happen such that physical interactions succeed and are safe.

The sense of touch is increasingly important for machines when they are fully autonomous or when interactions are not directly observable by the human operator, for instance, in teleoperation. A human remotely operating a robot arm requires feedback on the location, force, shape, and area of contacts to achieve purposeful interactions. A machine or robot autonomously negotiating its way in a corridor requires similar feedback for controlling its interactions with objects, machines, surroundings, and humans. For the same reasons as we humans require the sense of touch, machines realizing increasingly complex and autonomous behaviors require it to safely and purposefully interact with their surroundings, allowing them to coexist, collaborate, and support us, humans.

Large-area tactile sensing, which provides machines with tactile feedback for the large areas of their surface, and enables them to implement effective controlled reactions to contacts on their surface, has not been achieved so far. Endowing machines with the sense of touch is challenging and has been a research topic for decades [64, 34, 35]. The challenges are connected to the fact that the sense of touch is, in contrast to our other senses, a multimodal and distributed sense. For instance, our auditory and visual senses are localized in

our ears and eyes, the sense of touch, however, is distributed throughout the whole body. We humans employ around five million cutaneous receptors [32], that is tactile sensors, in around $2\,m^2$ [111] of skin, and connected to around 1.1 M nerve fibers [54]. The sheer amount of multi-modal sensors and their huge amount of information to transmit and process renders the realization of a comparable system for machines challenging. The distributed nature of the sensor system further complicates wiring and robustness, simply due to the spatial extension of the system. Robustness is additionally impacted by the skin's wear in its continuous physical interaction with the environment.

Recent advances in realizing the sense of touch for machines succeeded in solving or mitigating most of the encountered challenges [89, 134, 117, 112, 6, 91, 109]. Electronic skin (e-skin) systems emerged successfully covering larger areas, mitigating wiring, robustness, and deployment issues with modularity, and solving the issue of localizing thousands of sensors with self-organization. These e-skin systems successfully provided machines – and robots in particular – with the tactile feedback required for purposeful physical interactions. All e-skin systems so far focused on addressing the scalability of tactile sensing. The challenge of adequately handling a large amount of tactile information in large-area e-skin systems is not yet solved. Currently, tactile information handling does not scale due to computational efficiency deficits and limitations in computational power. Some works [28, 26, 84, 12] started to address this challenge but lack a systematic approach and concept that specifically address the complex system and constraints of a large-area e-skin. The lack of a method for systematically, efficiently, and flexibly handling the tactile information of a large-area e-skin so far limits its successful utilization, especially where machines require tactile feedback of large contact areas to realize physical interactions. This lack contributes to the fact that tactile sensing is yet not as established and widely utilized in machines as auditory or visual sensing.

One promising and increasingly prominent way to elevate the efficiency of information handling in technical systems is the *novelty-driven* approach. A novelty-driven system divides information into two categories. In *novel information*, that is essential and leads to a new system state, and in the information that repeats a known system state and is redundant in time. A novelty-driven system only acts on novel information. It is *driven* by it. The gain in efficiency of the novelty-driven approach lies predominantly in the fact that a system implementing it only transmits and processes information when it is strictly necessary and is inactive in all the other cases.

The concept of the novelty-driven approach emerged in the two different research fields of bio-inspired engineering [90, 22, 130, 86, 120, 31, 87, 121, 68, 51, 122, 28, 14, 67, 88, 95, 123, 118, 11, 12, 110, 38, 113] and energy-efficient signal processing [114, 97, 99]. Both fields developed realizations for novelty-driven systems and proved their effectiveness in sensing and computing.

As will be demonstrated in this thesis, the novelty-driven approach constitutes a very promising method for eventually realizing an efficient information handling system for large-area e-skin. In fact, the homogenization of the findings for novelty-driven systems of both fields will lead to efficient yet flexible and effective information handling for e-skin systems. Because, when flexibility allows for hardware independence and seamless system integration, the novelty-driven approach can boost the performance of existing e-skin systems and employ standard computing systems. A flexible, efficient, and rapidly deployable novelty-driven system tailored for e-skin will constitute a substantial contribution to the eventual realization of e-skin systems for effective large-area interactions. By substantiating these concepts to a general approach for e-skin systems, followed by its practical realization and experimental validation, this thesis significantly contributes to the advancement of large-area e-skin systems that will endow machines with human-like whole-body tactile interaction capabilities.

1.2. Challenges

The distributed nature of the sense of touch poses many challenges [64, 34, 35] that are particularly eminent when realizing a large-area e-skin with a large number of distributed multi-modal sensors in large areas. Challenges, such as robustness and reliability, deployability, wiring, and localization received a considerable amount of attention [131, 91, 109, 40]. Nevertheless, so far, none of the existing e-skins provide a systematic and efficient solution for the challenge of handling the large amount of tactile information they produce. Tactile information is multi-modal and has a complex structure that is neither spatially static nor aligned to flat matrices. Hence, standard information handling approaches cause high demand for communication bandwidth and computational power when handling tactile information to provide real-time feedback. For this reason, the feedback of current e-skin systems is often limited to body-parts rather than whole bodies [2, 40] or limited in spatial or temporal resolution [112, 70, 10]. Current e-skin systems yet have to prove their effectiveness in large-area tactile interaction where machines react to large-area contacts of many kinds without human supervision. Addressing the following challenges will lead to an efficient large-area e-skin system that will endow machines with human-like whole-body tactile interaction capabilities.

A Need to Formalize an Efficient Information Handling Method for Large-Area E-Skin
Standard information handling approaches cause high demand for communication bandwidth and computational power when handling tactile information. However, both resources are limited in distributed and autonomous systems. Solutions and mitigations that enable large-area sensing resolve to specific system characteristics (serial communication, low wire count, robust connections, low bit rates per wire due to physical conditions). These characteristics constrain feasible transmission bandwidths. Furthermore, effective e-skin couples with the autonomy of the system, but autonomous systems constrain computational and electrical power. A machine/system that achieves safe interaction with e-skin, but is too bulky to operate, is not effective. Thus, it is not feasible to solve the issue of handling tactile information by increasing communication bandwidth and computational power with the area of the e-skin. Instead, this issue results in the challenge of devising an efficient method for handling the tactile information of large-area e-skin. First, this method would have to reduce communication bandwidths and the computational demand for the handling of tactile information. Additionally, this method would have to comply with the existing solutions that enable the realization of distributed e-skin systems. The challenge of devising such a method also incorporates the difficulty of formalizing its fundamentals. Foremost, these fundamentals would have to provide the theoretical basis for correct designs and optimal parameterizations to ensure efficient realizations of the method. Furthermore, these fundamentals would be essential to evaluate the efficiency of the method, which is difficult. In this case, the fundamentals would have to provide the theory for the optimal design of experimental protocols and the analysis of their results. Another challenge of the fundamentals is that they would also have to connect to signal and control theory, such that correct comparisons to standard methods will be possible. Furthermore, it is challenging to extrapolate evaluation results for larger e-skins when

the sensing areas and the number of sensors increase further. Such an extrapolation would require models that the fundamentals of the devised method have to support.

To Realize Flexible Information Handling for Large-Area E-Skin Systems The flexible realization of the devised efficient information handling method for large-area e-skin is of paramount importance but challenging. The realization of the method in real e-skin systems is difficult but required to validate the method's correctness and feasibility and to confirm its efficiency with evaluations. The major challenge for realizing the method lies in devising a flexible design that allows for the very difficult integration of the method in existing e-skin and computer systems. This integration is critical since then, the realization of the designs would be practically applicable, and the method would lead immediately to efficient solutions, without the need to create e-skins from scratch, and without the need to transform the whole system to comply with the novel method. Another challenge is that a flexible design has to provide efficient interfaces that allow for combining the new method with standard clock-driven systems and algorithms. These interfaces would have to allow for using standard sampled sensors (clock-driven sensors) and connecting them to the more efficient communication protocol of the devised method. Such interfaces are difficult to realize, but they would allow the method to be realizable within existing e-skins. Similarly, the interfaces would have to allow for using the tactile feedback provided by the method's efficient information handling mechanisms in standard clock-driven algorithms, which is also challenging but would facilitate the method's integration in complex systems.

To Determine the Realization of Efficient Tactile Information Handling in Complex Systems Realizing efficient information handling in complex systems to process and react to the feedback of large-area e-skin is very challenging. While a novel flexible method might boost the performance of large-area e-skin and the efficiency of handling the tactile feedback it provides, the algorithms and calculations that consume the feedback would still impose significant limitations. These limitations would prevent the scalability of systems that depend on large-area tactile feedback and would thus curtail physical interactions to body-parts or constrain temporal resolution similar to previous approaches. Therefore, one challenge an efficient method for efficient large-area e-skin has to solve is providing efficient interfaces to standard clock-driven algorithms. Then, the overall interaction system would at least profit from the efficiency gain of the e-skin system. Nevertheless, standard clock-driven algorithms processing tactile feedback will still not scale when their demand for computational power increases proportionally with the size of contact areas or the number of tactile sensors, which unfortunately is true for many algorithms. This issue is particularly challenging, but, to ensure the effectiveness of the proposed method for e-skins, it has to be solved or mitigated. Solving this issue either breaks down into the challenge of transforming standard clock-driven algorithms to completely comply with the efficient information handling method for the e-skin or into the challenge of exploiting the efficient representation of tactile information in calculations without transforming the whole algorithm.

1.3. Contributions

This thesis contributes a systematic, efficient, and effective solution for the very challenging task of handling tactile information in large-area e-skin systems. The solution is inspired by biology, feasible to realize in existing e-skin and computer systems, and allows for the seamless integration in complex systems. For the first time, this solution enables tactile-driven large-area physical interactions and tactile-driven whole-body control.

A Systematic Approach for Efficient Information Handling in Large-Area E-Skin This thesis devises the biologically plausible principle of novelty-driven systems as the ideal method to solve the challenge of efficiently handling tactile information in large-area e-skin systems. Therefore, this thesis contributes a homogenized view of novelty-driven systems that fuses bio-inspired, neuromorphic interpretations with those from the perspective of information and control theory. The contributed homogenized view of novelty-driven systems result in mutual fundamentals that provide the theoretical background for the novelty-driven method this thesis utilizes for efficient e-skin systems. This thesis employs these fundamentals to devise correct and optimal designs for implementing the novelty-driven approach in existing e-skin and computer systems. This thesis contributes designs for detecting novelty, for novelty-driven communication, and novelty-driven information handling such that all designs comply with the requirements of large-area e-skin and standard computer systems. In these designs, the fundamentals allow for optimizing the sensitivity, transmission rate, and minimizing encoding errors and noise. Furthermore, these fundamentals allow this thesis to contribute new insights for novelty detectors regarding the interdependencies between their sensitivity, encoding error, system bandwidth, signal bandwidth, and the standard deviation of the noise. The fundamentals additionally provide insights that have the potential to lead to more sophisticated novelty detectors with enhanced robustness and error correction features. The fundamentals this thesis contributes for novelty detectors also takes realizations in electronic circuits into account. This consideration allows this thesis to propose a new realization of a hybrid novelty detector that merges the advantages of continuous novelty detectors with those of the hardware-independent novelty-driven communication protocol that this thesis utilizes. Furthermore, exploiting the fundamentals of its novelty-driven method, this thesis contributes a CPU usage model that describes the relationship between the event rate in a novelty-driven e-skin and the computational load the handling of these events causes. This model allows for understanding the behavior of the realized novelty-driven system and thus for extrapolating computational loads for larger e-skin systems.

A Realization of Flexible Information Handling for Large-Area E-Skin Systems This thesis solves the challenge of realizing flexible and efficient information handling in large-area e-skin systems. This thesis implements its methods and designs for efficient information handling in an existing e-skin system and thus contributes a novelty-driven large-area e-skin. By doing so, this thesis demonstrates that its flexible, novelty-driven design is feasible

and correct. To empirically prove the efficiency of its novelty-driven approach and designs, this thesis contributes a comprehensive evaluation for e-skin systems. These evaluations consider the dependency of novelty-driven systems on stimulus shapes in their experimental protocols. This thesis exploits the evaluation results to contribute a procedure that optimizes the event communication protocol in cases where e-skin systems require fixed packet sizes. This thesis can demonstrate that the contributed procedure succeeds in minimizing communication overheads. Furthermore, the evaluation results, together with the devised extrapolation models, allow this thesis to empirically prove the scalability and efficiency of its novelty-driven approach for large-area e-skins. To solve the challenge of flexible information handling, this thesis contributes efficient interfaces from clock-driven systems (standard systems) to novelty-driven systems and from novelty-driven systems to clock-driven systems. Furthermore, all the designs this thesis contributes to realize novelty-driven information handling are general and hardware-independent. The contributed designs emphasize minimal prior requirements that do not clash with the solutions of existing e-skins and that allow implementations on standard computer systems. This implementation flexibility, combined with the interfaces to standard clock-driven sensors and clock-driven algorithms, allows for realizing the novelty-driven approach in existing e-skins and integrating tactile feedback in complex clock-driven systems.

This thesis solves the challenge of realizing flexible and efficient information handling in large-area e-skin well beyond the scale of existing e-skin systems. This thesis contributes, for the first time, a large-area e-skin with more than 10 000 multi-modal tactile sensors. This realization validates the correctness and feasibility of this thesis' approach beyond existing scales for large-area e-skin systems. Furthermore, its subsequent evaluation demonstrates, that this thesis' approach for the first time allows for the perceptual handling of tactile information of more than 10 000 sensors where all previous approaches failed.

A Realization of Efficient Tactile Information Handling in Complex Control Systems
This thesis contributes solutions to realize efficient tactile information handling in complex systems, and in particular, to the very challenging integration of large-area tactile feedback in robotic systems. To achieve desired behaviors in their physical interactions, these robot systems have to fuse tactile information with proprioceptive information to motor actions. One solution this thesis contributes is the hybrid event-driven reactive contact control system. This system utilizes the tactile feedback of a novelty-driven e-skin in its clock-driven control algorithm. This thesis successfully realizes this system demonstrating its feasibility and efficiency without negative effects on control performance. Thus, the novelty-driven e-skin succeeds in providing tactile feedback to complex systems, even when the algorithms consuming it are still clock-driven, underlining the flexibility and immediate applicability of this thesis approach. Furthermore, this thesis contributes a novel concept to distribute the computationally expensive calculations of reactive contact controllers. Thereby, this thesis succeeds in removing any limitations on the contact area and the total number of tactile sensors from reactive contact control. This unique approach allows the clock-driven reactive control system to scale

with the number of tactile sensors whereby all previous approaches failed. To realize the distributed calculations on platforms with very limited computational power (for instance, without Floating Point Units (FPUs)), this thesis contributes general optimizations for the efficient computation of forward kinematics and joint torques. These optimizations are general and thus allow for faster calculations of tactile-driven reactions on any hardware, improving the scalability of reactive contact control algorithms for larger numbers of tactile sensors. Moreover, this thesis contributes strategies that further improve the scalability and efficiency of tactile-driven reactive contact control. For one, these strategies exploit the efficient properties of novelty-driven information to improve clock-driven algorithms. Or, these strategies exploit the mechanisms of the novelty-driven approach to convert clock-driven algorithms to equivalent novelty-driven algorithms with superior performance characteristics. This thesis successfully validates these strategies, and evaluations prove their efficiency. The strategies for the efficient integration of novelty-driven tactile feedback in clock-driven reactive contact control significantly contribute to the feasibility of tactile-driven whole-body control. This thesis demonstrates this feasibility with the integration of the novelty-driven large-area e-skin system (with more than 10 000 sensors) in an autonomous humanoid robot. This integration allows this thesis to demonstrate whole-body tactile interactions with a humanoid robot that have previously not been possible. For instance in applications, where tactile feedback enables whole-body interaction with a humanoid robot that balances. Or in the locomotion of a humanoid robot, where tactile information allows for obstacle avoidance. Or in realizing tactile-driven leader-follower interactions, as they occur in human-robot dancing.

All in all, the contributions of this thesis, realize, for the first time, efficient large-area tactile feedback allowing for novel applications of large-area e-skin whereby all previous approaches failed. Thereby, this work will hopefully positively impact a novel trend where increasingly more autonomous machines successfully exploit the sense of touch and thus improve their physical interaction capabilities with their surroundings.

1.4. Thesis Outline

This thesis consists of six chapters.

Chapter 2 – Background & Related Work This chapter introduces the neuroscientific view of the sense of touch, the engineering challenges of e-skin systems. It furthermore introduces the information handling system from the perspectives of neuromorphic engineering, signal processing, and information theory, and novelty-driven systems. Eventually, a comparative study determines the most flexible and applicable for realizing the novelty-driven principle for large-area e-skins.

Chapter 3 – Fundamental Elements of the Event-Driven Approach This chapter presents the homogenized fundamentals of novelty-driven systems. It presents the theory for the correct design and optimal parameterization of novelty detectors for the e-skin's sensors, the model for the computational demand of novelty-driven e-skin systems, and the foundations for evaluating the performance of novelty-driven e-skin systems.

Chapter 4 – Realizing an Event-Driven Large-Area Skin System This chapter presents the design, realization, and experimental validation and evaluation of the efficient novelty-driven large-area e-skin. The experiments demonstrate the feasibility of e-skin systems with more than 10 000 multi-modal tactile sensors.

Chapter 5 – Realization and Validation in Applications This chapter presents very complex systems that utilize the feedback of large-area contacts. It presents the effective and efficient integrations of the novelty-driven large-area e-skin in robotic systems and their validation and evaluation in experimental setups. Further experiments with the large-area e-skin on a humanoid robot demonstrate the e-skins effectiveness in the very challenging utilization of large-area tactile feedback in whole-body controllers.

Chapter 6 – Conclusion This chapter summarizes the thesis in a final conclusion and provides an outlook for future steps.

2. Background & Related Work

This chapter presents the background, foundations, and related works connected to novelty-driven large-area e-skin systems. It presents and addresses related research topics such as the neuroscience of the sense of touch, the engineering of large-area tactile sensing, and standard and bio-inspired information handling systems. The presented literature provides the reader with an overview of the covered fields and consolidates different views on the concepts and terminology relevant to this thesis.

The development of e-skin is chiefly motivated and inspired by the human sense of touch. Insights from neuroscience, such as the tuning of tactile receptors to specific stimulus features, the early structuring of information and representation in tactile images of the self, or the neural encoding of information, reflect in the realizations of e-skin systems. They furthermore provide the foundations for the bio-inspired novelty-driven approach, this thesis proposes for large-area e-skin systems.

The structure of this chapter is as follows. First, Section 2.1 briefly introduces and describes the sense of touch with a particular focus on the neuroscientific insights relevant to this thesis. The definition and description of large-area e-skin follow in Section 2.2. This section relates large-area e-skin to the sense of touch, surveys existing approaches for e-skins, and summarizes the challenges of large-area e-skins. The efficient handling of tactile information is still a significant challenge that this thesis addresses. Therefore, Section 2.3 continues with a description of information handling systems to introduce the topic and establish connections to standard signal processing theory and biologically inspired information handling. Novelty-Driven Systems (NDSs) are a subgroup of the biologically inspired Event-Driven Systems (EDSs). The section introduces both information handling systems and establishes a connection to standard signal processing theory. Therefore, it additionally introduces Clock-Driven Systems (CDSs). CDSs implement information handling according to standard signal processing theory. Afterward, Section 2.4 introduces and describes different existing approaches for NDSs and their implementation. With the comparison of these novelty-driven approaches, this section subsequently assesses their applicability in large-area e-skin. This assessment results in the selection of an NDS implementation that suits best for this thesis's objective to tailor an efficient novelty-driven approach for realizing efficient large-area e-skin.

2.1. Neuroscientific Insights into the Sense of Touch

The human sense of touch efficiently perceives complex contact features throughout the whole surface of the body and conveys the acquired information to the brain [54, 1, 119]. Neuroscience delivers valuable insights into the biological mechanisms involved in sensing, encoding, and transmission. These insights, for instance, encompass efficient sensing, efficient and sparse information representation, and the early structuring of information. They and play a critical role in the process of developing bio-inspired principles (Section 2.3.1) that improve the efficiency of technical systems, for instance, e-skin (Section 2.2). The following sections briefly summarize the notable mechanisms in the sense of touch's pathway, which is from the peripheral skin receptors to the central nervous system and the brain.

2.1.1. Skin Receptors are Tuned to Sense Specific Stimulus Features

The sense of touch employs specialized receptors for sensing mechanical, thermal, and noxious (potentially dangerous/destructive) stimuli [53, 1, 127, 54, 55, 128]. These receptors are tuned to sense specific stimulus features that focus on deciphering distinct pieces of contact/object properties. The dominant stimulus features are normal pressure (Merkel cell receptors [53, 55, 1]), horizontal motions and slip (Meissner corpuscle receptors [53, 55, 1]), vibrations (Pacinian corpuscle receptors [55, 1]), stretch (Ruffini endings [55, 1]), and proximity/approach (tylotrich-hair receptors [55, 1]). Figure 1 depicts the two kinds of skins found in humans and the location of these receptors.

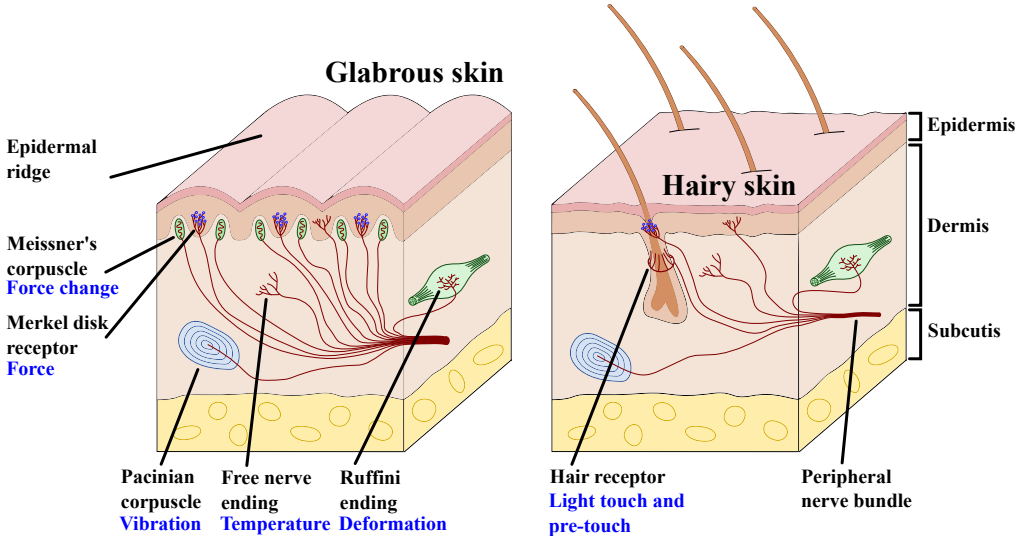


Figure 1 The two types of Human Skin. Right: The glabrous (hair-less) skin. Left: the hairy skin. Adapted from [55, 1].

The work presented in Section 2.1 was in part published in:
Bergner, F., Dean-Leon, E., Cheng, G., "Design and Realization of an Efficient Large-Area Event-Driven E-Skin". In: *Sensors* 20.7 (2020), p. 1965, and
Bergner, F., Cheng, G., "Sensory Systems for Robotic Applications – Making sense of the world". In: ed. by Ravinder S. Dahiya, Oliver Ozioko, and Gordon Cheng. The Institution of Engineering and Technology, 2020, Submitted. Chap. Neuromorphic Principles for Large-Scale Robot Skin.
Copyright permissions: see Appendix D.

The receptors' stimulus feature selectivity is influenced by the skin receptors' location in the different dermal layers, by the deployment pattern, and by mechanical filter mechanisms. The receptors' selectivity separates complex multi-modal stimuli to simple distinct uni-modal stimulus features, allowing for the encoding of complex tactile information and selective attention. Peripheral axons connect these tactile receptors to the nerve cell bodies in the dorsal root ganglion next to the spinal cord [54, 1, 58, 13, 119], forming together the tactile part of the peripheral somatosensory system (Section 2.1.4).

2.1.2. Skin Receptors Transduce Stimulus Features to Binary Action Potentials

The representation and conveyance of information in biology follow schemes quite different from the principles utilized in technical systems. The representation in biology could neither be described as analog nor digital. Biology uses *binary action potentials*, often also termed *spikes* or *events*, to represent and convey information between neurons [72]. Action potentials alone convey only a very limited amount of information. Action potentials in nerve fibers are either present or not. They do not convey any additional information, for instance, in their shape or amplitude. Information in biology is encoded in the *spatio-temporal activity patterns* in massively parallel nerve bundles or populations of neurons [55, 128] (Section 2.1.3).

The specialized skin receptors (Section 2.1.1) apply different functional principles to transduce stimulus features of different modalities to action potentials. Nevertheless, the final step of all transduction chains is similar for all types of skin receptors. Stimulus features either directly or indirectly gate ion-channels in the terminals of peripheral nerve fibers [133, 54]. The influx of ions into the peripheral axons of the dorsal root ganglion neurons depolarizes these neurons and eventually creates action potentials/spikes [53, 54]. Therefore, the generation of novel sensory information encoded in action potentials takes place at the periphery, at the receptor level, allowing for efficient novelty-driven encoding at the earliest stages, before conveying information to the higher processing layers in the Central Nervous System (CNS).

Action potentials travel through nerve fibers completely asynchronously without any strict relation to a clock, as it would be the case in synchronous systems. This asynchronism allows for very high temporal precision since the occurrence time of the action potentials is time continuous. The propagation speeds and transmission rates of action potentials in nerve fibers depend on their myelination and diameter [133, 54]. The transmission of action potentials can reach rates up to 1000 times per second. Thus, since an action potential is binary, this rate compares to a rate of 1 kbit/s.

2.1.3. Skin Information is Encoded by Different Neural Codes

The information representation in biology, the neural codes, explain how action potentials encode information. Biologically inspired, event-driven systems (Section 2.3.1) replicate these principles to encode information to their events in a similar way. Therefore, this section pro-

vides the reader with an overview of the known neural codes. NDSs, as a special type of EDSs, are principally drawn from the principle of temporal codes.

Information in biology is conveyed and represented by spatio-temporal activity patterns in bundles of nerve fibers and populations of neurons [55, 128]. These neural codes employ a set of different information representation principles, which are: 1) type code, 2) spatial code, 3) rate code, 4) temporal code, and 5) latency code [139, 55, 128]. All these principles show that biology uses *structure* and *time*, that is spatio-temporal features, to encode and represent information. Although there has been a long debate if biology employs rate coding or temporal coding [39, 139], the nervous system employs both.

The type and spatial neural codes exploit structure for encoding information. Nerve fibers of a peripheral neuron only innervate receptors of one particular type and consequently apply the type code. That is, the nerve fiber itself encodes the type of the stimulus feature. Spatially distributed populations of receptors of the same type can encode spatial information through activity patterns [128]. These populations apply the spatial code. The sense of touch extensively employs these spatial codes for Merkel cell and Meissner corpuscle receptors [128, 53, 56, 55]. The spatial codes encode information in the spatial relations between nerve fibers, which is in the distance between nerve fibers. Therefore, biological systems have to keep the ordered structure, which is the *somatotopic order* (Section 2.1.4), in ascending bundles of nerve fibers (nerves).

Rate codes exploit the time to encode intensities by proportionally modulating spike rates. Receptors employing rate codes, such as the Merkel cell receptors and the Ruffini endings in the sense of touch, are innervated by Slowly Adapting (SA) nerve fibers [128, 53, 56, 55].

Temporal codes also exploit time and encode information in the temporal sequence of spikes. Receptors employing temporal codes are innervated by Rapidly Adapting (RA) nerve fibers. Meissner corpuscle receptors and Pacinian corpuscle receptors are innervated by RA nerve fibers [54, 1]. RA nerve fibers, and respectively the receptors they innervate, are exclusively sensitive to stimulus feature changes and encode the exact time of each change.

The latency coding scheme combines spatial and temporal information [128]. There, the relative spike arrival time in the population of neurons encodes moving stimulus features by employing the latency coding scheme.

2.1.4. Skin Information Ascends Somatotopically Ordered

Throughout all its parts, the somatosensory system maintains the somatotopic order of the conveyed and relayed information, that is, the order of its nerve fibers and nerve cells reflects the relative spatial structure of its receptors [54, 1]. Ascending along the spinal cord via relay centers in the medulla and thalamus towards the primary somatosensory cortex (S1), the somatosensory system assembles somatotopically ordered information of different body

parts to a comprehensive sensory representation of the whole body, the homunculus [54, 1, 4, 119].

The two dominant principles in the biological sense of touch are the decomposition of complex contact features of physical interactions to simple uni-modal stimulus features, and the maintenance and assembly of relative spatial information. These two principles significantly impacted the development of large-area e-skin systems [34, 33, 37] (Section 2.2). Furthermore, the insights of the neural encoding principles significantly impacted the development of efficient biologically inspired information handling systems (Section 2.3).

2.1.5. Human Skin in Large-Areas

An in-depth study of the sense of touch reveals its fundamental differences in comparison to humans' other senses. These differences break down to two facts:

1. The sense of touch is a highly *distributed* sense. The sense of touch spreads out its five million cutaneous receptors [32] (mechanoreceptors, thermoreceptors, nociceptors) through the whole body in large areas up to around $2\,m^2$ [111]. Then, it conveys its tactile information through around 1.1 million ascending nerve fibers [54] to the somatosensory cortex. In contrast to the sense of touch, vision is a very concentrated sense. The human eye accommodates approximately 137 million receptors (130 million rods and 6.5 million cones per retina) [129] and approximately one million nerve fibers [73] in an area of around $1100\,mm^2$. Hence, the sensing area of the human skin is around 1000 times larger than the sensing area of both eyes. In this sense, the term *large-scale* attributes to large resolutions in vision and to *large-areas* in tactile sensing.

2. The sense of touch acquires its information through *physical contacts*. In contrast to vision or audition, which protects their receptors against physical contacts, the sense of touch depends on these contacts with the environment to acquire information. Tactile receptors cannot acquire tactile information through distant observations – they need to access the information in the area of contact.

Along with the sensing and information handling principles of Section 2.1, the distributed nature and the dependency on physical contacts of the sense of touch profoundly impact how the sense of touch organizes sensing in biology [74]. These insights gathered from the sense of touch lead to the formulation of guidelines for the effective design of electronic skin (e-skin) [34, 36, 37].

2.2. Large-Area E-Skin

The sense of touch plays an essential role in our lives. It allows us to interact and to control our contacts with our surroundings. The sense of touch not only allows us to characterize and evaluate contacts, but it also allows us to locate them. It provides us with a tactile image of our interactions with the world.

The human brain represents this tactile image in the somatosensory cortex. There, it combines proprioceptive information with cutaneous information and assembles an internal model. This internal model, the homunculus, associates the postural information of the body with the spatial location and tactile information of the skin receptors [119, 4] (Section 2.1.4).

The following sections present existing realizations of electronic skin (e-skin) and explain the encountered challenges and their solutions and mitigations. Furthermore, they highlight the absence of a solution for handling the information of large-area e-skin.

2.2.1. Large-Area E-Skin Systems

The developments of e-skin focus on two different kinds of skins, similar to the two found in humans [1, 55] (Section 2.1.1, Figure 1). One skin is mainly located in the inner sides of the hands and the soles of the feet, while the other skin covers the remaining parts of the body.

The skin on the hands and feet covers small regions and targets very high spatial resolution, supersensitive sensing, shear-force, and vibration sensing, and slip detection [1]. Thus, e-skin targeting this kind of skin deemphasizes the challenges of distributed sensing and focuses on high sensing density and the challenges connected with supporting physical contacts. The works of [138, 49, 79] develop fingertip e-skins which aim for high-density tactile sensing.

The skin covering large areas of the body emphasizes the distributed nature of the sense of touch. While large area skin may slightly deemphasize high spatial resolution, it has to specifically focus on efficient and feasible methods to deploy, connect, and determine the poses (location and orientation in 3D space) of a *large number* of spatially distributed tactile sensors over *large areas*. These e-skin systems are *Large-Area Skin Systems (LASSs)*.

Realizing LASSs has been the topic of research in the last decade. Researchers focused on different approaches towards scalable multi-modal e-skin systems suited for large-area applications.

The work presented in Section 2.2 was in part published in:
Bergner, F., Dean-Leon, E., Guadarrama-Olvera, J. R., Cheng, G., “Evaluation of a Large Scale Event Driven Robot Skin”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4247–4254, and
Bergner, F., Dean-Leon, E., Cheng, G., “Design and Realization of an Efficient Large-Area Event-Driven E-Skin”. In: *Sensors* 20.7 (2020), p. 1965.
Copyright permissions: see Appendix D.

The early work of [30, 89] utilizes arrays of IR based proximity sensors (80 sensors in total) on a robot arm. The authors feed the acquired tactile proximity information into a planner such that the robot can avoid obstacles in real-time. The work demonstrates the feasibility of tactile feedback in the implementation of physical interactions and motivated the development of larger, multi-modal e-skins.

The work of [134] introduces a flexible, bendable and stretchable matrix of pressure and temperature sensors as conformable e-skin. The authors demonstrate the feasibility of manufacturing such an e-skin up-to a size of 12×12 (144 sensors). The authors of [136] present another flexible e-skin that employs matrices of force sensors. The authors [117] improve the deployability of flexible tactile sensor matrices through modularization and enabling simple customization (the e-skin can be cut into shape).

To overcome the main drawbacks of sensor matrices, namely the high number of wires and the susceptibility to row-/column-wise failures, modular solutions for e-skin have emerged. These systems combine tactile sensors covering a specific predefined area in a module. Modular systems have been introduced in the following works.

The RI-MAN robot [112] employs five tactile sensing modules (two per arm, one on the chest). Each module embeds an 8×8 force sensor matrix. The e-skin utilizes in total 320 force sensors. Similarly, the ARMAR-III robot [6] utilizes modules that employ force sensor matrices on the robot's shoulders and arms. The TWENDY-ONE robot [70] achieves a more complete coverage with force sensors distributed in its hands (2×241 sensors), in its arms (2×54 sensors), and its trunk (26 sensors). The sensing density of these systems is low and concentrated in the modules.

E-skins building upon similar, exchangeable sensing modules further advanced modularity, deployment flexibility, robustness, and scalability. This approach resulted in larger e-skin systems that are presented in the following.

The work of [25] introduced the e-skin, RoboSkin. This e-skin utilizes skin modules with a triangular shape. Each skin module employs 12 capacitive force sensors. The assembly of these triangular skin modules forms the e-skin that can cover arbitrary surfaces. This modular approach allows great deployment flexibility, an effective reduction of wires, and the containment of failures. The works of [131, 91] demonstrate that the modular approach of RoboSkin e-skin allows for the full coverage of robots. These works demonstrate the feasibility of large-area e-skin by successfully covering a Nao robot with 200 force sensors [131] and an iCub robot with 2000 force sensors [91].

Nevertheless, the RobotSkin e-skin only employs one sensor modality (normal force) and hinges on the complex localization of the tactile sensors [24, 47, 3]. Furthermore, this e-skin utilizes a query-based readout system [10] and a query-based middleware that organizes and

represents tactile information [144]. A query-based information handling system is inefficient, exhibits large latencies, and does not scale. In a query-based system, the system has to ask for the information of each sensor. The system has to continuously loop serially through all sensors, request each sensor value, and wait for the answer. The communication latency for each sensor doubles, and the overall latency scales with the number of sensors.

Mittendorfer et al. [101, 100, 109] introduced a significantly improved modular e-skin with hexagonally shaped multi-modal sensing modules. Each module employs up-to nine sensors that sense force, distance, temperature, and vibration [101, 103]. The advantage of this e-skin, in comparison to all previous works, is its capability to self-organize its sensing modules. The self-organization eases deployment and allows for robustness, flexibility, and self-localization. The e-skin can automatically determine the locations of its modules and reconstruct the 3D surface it covers [102]. Furthermore, the e-skin transmits tactile information with a predefined sample rate of up-to 250 Hz. Thus, the information handling system does not have to query information. The e-skin provides the information with a defined sample rate. The capability to self-organize, to self-localize, and to provide information without queries allow this e-skin to scale. This e-skin addresses most of the challenges of distributed e-skin systems. It has been successfully deployed on robots with up-to 300 skin modules (2700 multi-modal sensors) [109, 40].

So far, all these works addressing LASSs solely focus on the scalability of sensing. But they do not address the significant challenge to handle a large amount of tactile information the upscaling of sensing induces. The lack of a systematic approach to tackle this challenge explains why LASSs are not yet as widely utilized as other sensing systems, for instance, auditory or visual. This lack also explains why the tactile feedback of LASSs is often limited to interactions with body parts rather than whole bodies [2, 40], or limited in spatial or temporal resolution [112, 70, 10].

2.2.2. Challenges of Large-Area E-Skin

This section consolidates the specific and most notable challenges encountered when designing and realizing LASSs [34, 36, 35, 37, 109]. The subsequent Section 2.2.3 then presents how previous works addressed these challenges.

2.2.2.1 Reliability and Robustness

Mechanical interactions stress and wear on a physical system. The standard method to protect these systems against these effects is caging and protecting fragile parts while keeping the spatial extension as compact as possible. Ensuring reliability and providing robustness is a non-trivial task in LASSs since these systems are highly distributed and have to endure physical contacts.

2.2.2.2 Deployability and Wiring

Distributing thousands of tactile sensors throughout the body and wiring them to supply the sensors with power and low latency connections is challenging. Deploying and wiring the sense of touch strictly requires a systematic approach. A large number of discrete sensors are infeasible to place and connect one-by-one without a systematic concept. The time required to set up the system would be impractical and the maintenance time consuming, error-prone, and complex.

2.2.2.3 Localization

The coupling between sensing and *locating a stimulus*, that is, determining of its pose (position and orientation) with respect to a point of reference, is of paramount importance in vision as well as in touch. Body surfaces covered with skin are three dimensional and not completely rigid. Additionally, an actuated body changes the relative positions between its body parts and thus the positions between the skin sensors. Consequently, the identification of skin sensor locations with respect to a body part exhibits a complex challenge, even if the sensors have been systematically deployed. Manually assigning the poses of thousands of sensors can become infeasible, and a LASS could benefit from automated methods to acquire the location of its sensors.

2.2.2.4 Low-Latency and Efficiency

Sensing systems need to provide and represent information with *low-latency* and enable *efficient information handling* in order to realize fast system responses. Ensuring low-latency and efficient information handling becomes more challenging with an increasing number of sensors and is particularly challenging in LASSs [35]. In visual or tactile applications, sensing systems have to handle a large amount of information (> 10 MB/s) within short periods (< 10 ms). High-speed connections between sensors and information handling systems are feasible in concentrated systems, for example, cameras. However, they are hard to realize in distributed systems with many connections over long, varying distances. Long-distance connections and high bandwidths increase the influences of noise, crosstalk, reflection, and distortion. All these effects contribute to the loss of signals and failures in the power distribution. That is, signal and power integrity are harder to maintain when distance and bandwidth increase. Thus, low-latency connections between the distributed tactile sensors, and handling a large amount of tactile information are both demanding challenges in LASSs. For example, as it has been shown, an e-skin system can handle, with one computer, around 300 sensor modules with 2700 sensors sampled at 250 Hz [109, 40, 16]. This information handling approach is not feasible for a larger number of sensor modules, for instance, 1260, as demonstrated in [20].

2.2.3. Key Principles for Large-Area E-Skin

The tackling of the challenges summarized in Section 2.2.2 requires a systematic approach. Solutions targeting only one particular challenge might completely contradict or hinder solutions targeting the other ones. Recent progress in realizing e-skin systems revealed that following the bio-inspired principles of *modularity* and *self-organization* within a system approach contributes towards solving and mitigating the first five challenges (reliability, robustness, deployability, wiring, and localization) of LASSs [25, 109]. The following sections briefly summarize the impacts of these two principles and their limitations towards solving/mitigating the challenges of low-latency and efficiency.

2.2.3.1 Modularity of Sensing Elements

Modularity breaks down a complex system into smaller, less complicated, and exchangeable modules [25, 109]. The simpler a module, the lower the number of points-of-failure is. Modules allow the containment of errors and the introduction of redundancy, enhancing the robustness of systems. A system divided into exchangeable and modular parts is better to maintain, is more flexible, and easier to deploy. A modular system is changeable and customizable, thus optimizable for specific deployment scenarios. Structuring modules into hierarchical entities (e.g., skin patches) further simplifies the deployment of a large number of modules. A module can deploy a set of components or other modules, rather than each component or module on its own. Modularity significantly contributes to the deployability of a system.

2.2.3.2 Self-Organization of Communication and Structure

A system that is not only modular but additionally self-organizing significantly contributes to the feasibility of large systems with thousands of modules. Self-organizing modules can form networks with short-distance connections between neighbors rather than requiring long point-to-point connections between each module and a hub. A meshed network of modules simplifies the wiring challenge while it introduces at the same time connection redundancy that enhances the robustness of the system [109]. Self-organizing modules are also instrumental to automatically determine the structure of the deployed e-skin and the spatial distribution of its sensors. They allow for a feasible solution for the infeasible task to manually determine the poses of thousands of tactile sensors [102]. Dynamic network routing can further enhance the robustness of a self-organizing network of modules. This dynamic routing allows for the automatic on-line reshaping of communication trees in meshed networks to handle broken connections or unbalanced communication loads without the need to restart the system [8].

2.2.3.3 Limitations of Modularity and Self-Organization

Systems of self-organizing modules rely on modules with local processing capabilities. Besides realizing self-organization, these local processing capabilities can be exploited to filter [109] or fuse information [18] in the modules, reducing the computational load at the higher processing layers. However, the concepts of modularity and self-organization cannot directly contribute to realizing efficient information handling in large-area e-skin with low latency.

2.3. Information Handling Systems

Complex systems, artificial or biological, combine sensation, communication, processing, and actuation to achieve desired system behaviors; they need to *handle information*. Handling information not only refers to processing information, but furthermore addresses the complete information flow in a perception-action loop [52] or a system control loop, that is, acquiring, transmitting, processing, and acting on information [72]. In this sense, achieving the desired system behavior fundamentally depends on the fast, efficient, and loss-less representation, processing, and exchange of information.

Up to date, most technical systems are clock-driven and handle the information strictly following the Nyquist-Shannon sampling theorem (Section 2.3.2). Clock-Driven Systems (CDSs) have been successfully applied in many different applications such as the high-speed precision motor control in hard drives [143], control of robot arms [65], and many more. These systems not only prove that CDSs provide viable solutions, they usually achieve excellent performance [143, 65].

Nevertheless, CDSs that require fast reaction times in real-time applications depend on high-bandwidth information resulting in high sample rates. For example, the high-speed precision motor control in hard drives responds within 150 µs, that is, its sample rate is 7.7 kHz [143]. High sample rates only marginally impact systems that handle a limited amount of information of a few sensors, for example, the position control of electrical motors. However, when systems have to handle a large amount of information with high sample rates, for instance, in tactile sensing or vision, their realization may become challenging or even infeasible.

The following works depict the challenges in such systems. The work of [69] introduced real-time high-performance attention in color video streams. To realize their system, the authors had to distribute the processing to nine computers. The work of [93] introduced distributed real-time processing for humanoid robots to provide the large amount of computation power required in perception, planning, and control. Two computers embedded in the robot could not provide enough computation power. The work of [5] introduced real-time feature tracking on small vision-guided unmanned vehicles. The authors had to implement their algorithm in a specialized high-performance hardware implementation in a Field Programmable Gate Array (FPGA) to meet the complex constraints in size and performance. The work of [57] demonstrate a real-time system for visual processing optimized for Central Processing Units (CPUs) rather than Graphics Processing Units (GPUs) or FPGAs, avoiding the demand for high power, space, and hardware customizations. The complexity and challenges of handling the information of large-area e-skin have been discussed in [34] and addressed in [9, 144].

The work presented in Section 2.3 was in part published in:
Bergner, F., Dean-Leon, E., Cheng, G., "Design and Realization of an Efficient Large-Area Event-Driven E-Skin". In: *Sensors* 20.7 (2020), p. 1965.
Copyright permissions: see Appendix D.

The current approaches do not scale, have large latency, and a high demand on computation power.

Thus, the challenge of handling a large amount of information with high sample rates emerges in systems that need to react fast to the input of many sensors. As all these works demonstrated, CDSs have to employ very performant transmission and computing systems with severe demands on power and space to handle a large amount of information with low latency. While power and space mainly cause financial and environmental disadvantages in stationary systems, both factors tremendously impact systems in mobile applications, for instance in distributed low power sensor networks [114, 97], in autonomous embedded vision [5, 57], and in autonomous robotics [93].

The depicted limitations of traditional approaches in applications that need to handle a large amount of information within short periods triggered the development of spike-based bio-inspired and neuromorphic systems. These systems mirror the incredibly high information handling efficiency of biological systems [74]. Some principles of these bio-inspired systems might contribute to the development of efficient information handling in large-area e-skin.

The following sections focus on introducing bio-inspired information handling (Section 2.3.1), and presenting the definition and characteristics of CDSs (Section 2.3.2), and NDSs and EDSs (Section 2.3.3). These sections provide the background for the subsequent comparison of implementations for bio-inspired information handling regarding their applicability in LASS (Section 2.4).

2.3.1. Bio-Inspired Information Handling

The works of [90, 22] had been among the first that introduced event-driven information handling (Section 2.3.3) in bio-inspired neuromorphic systems more than two decades ago. Event-driven information handling systems employ spike-based information representation principles in sensing, communication, and processing [87, 88, 11]. These spike-based representations resemble the representations in biology which employ action potentials (Section 2.1.2) and neural codes (Section 2.1.3) [87].

Event-driven neuromorphic systems report significant improvements in efficiency and speed [87, 88]. That is, the systems require less power and handle information with higher temporal resolution and less latency. The following works document this efficiency.

Event-driven neuromorphic sensors transduce audio [130], visual [86, 120, 121, 122], and force signals [28] to event trains and achieve temporal resolutions and latencies that have previously not been possible with sample-based sensors (clock-driven sensors). The event-driven implementations for representation and communication demonstrate encoding and transmission efficiency in [90, 22]. Event-driven algorithms for visual processing [14, 118] are feasible and reduce computational costs. For instance, the work of [31] presented a

high-speed vision controlled pencil balancer to emphasize the low latencies possible in event-driven systems. Many works presented approaches towards efficient event-driven computing. These systems employ specialized hardware that mix analog and digital circuits [68, 95, 123, 110, 113], or network specialized computational units [51, 67, 38]. Despite these systems' efficiency and scalability, the dependency on specialized hardware and software currently limit their applicability in complex systems and complicate system integration and maintenance.

Novelty-driven systems (NDSs) are a subgroup of event-driven systems (EDSs). Novelty-driven systems focus on two bio-inspired principles that significantly contribute to the efficiency and speed in event-driven systems. First, novelty-driven systems simplify the concept of neural codes in spike-based systems to the principle of *novelty-driven information handling*, that is, only novel information in the form of events drives the whole system. This simplified principle is still biologically inspired. Even employing different coding principles, the activity in the populations of neurons is usually triggered (or inhibited) by the arrival of stimuli. This principle is particularly true for the afferents of sensory neurons. Their peripheral axons only generate action potentials when their receptors register stimuli and are otherwise silent, regardless of the neural code they utilize for conveying information [72, 55]. Second, novelty-driven systems discard rate coding. Focusing on the sparsity aspect of spiking neural networks allows for exploiting their capabilities for *temporal redundancy reduction* and *saliency* enhancement. Both capabilities, temporal redundancy reduction, and saliency, significantly contribute to a system's efficiency since the system has to handle less information.

Novelty-driven systems especially proved their efficiency in the very complex and computationally expensive real-time processing of visual information [31, 14, 118]. The dynamic vision sensors of [86, 120, 121, 122] particularly focus on the sparse coding of novel visual information in events.

2.3.2. Clock-Driven Systems

This section defines clock-driven systems and presents their theory.

Standard time-discrete systems follow the Nyquist-Shannon sampling theorem that defines constraints for the lossless conversion of time-continuous signals to time-discrete signals. The Nyquist-Shannon sampling theorem [132] states that any bandwidth-limited time-continuous signal $x(t)$ with $t \in \mathbb{R}$ can be represented by a time-discrete signal $x(t_k)$ with $t_k = k\,T_\mathrm{s}$ and $k \in \mathbb{Z}$ as long as the sampling frequency f_s surpasses the bandwidth B of $x(t)$ by at least a factor of two:

$$f_\mathrm{s} > 2\,B. \tag{2.1}$$

Consequently, time-discrete systems ensure that a clock with at least a frequency of f_s drives the information handling such that the constraint of the Nyquist-Shannon sampling theorem is fulfilled at all times and information loss is zero through all stages of the system. These

systems are termed Clock-Driven Systems (CDSs). Standard computing systems are CDSs and usually either implement the von Neumann [115] or the Harvard architecture [135].

2.3.3. Event-Driven Systems

This section defines event-driven systems and presents their connection to standard signal processing theory. Existing work on the implementations of event-driven systems and their applicability in large-area e-skin will follow afterward in Section 2.4.

Within the scope of this thesis, Event-Driven Systems (EDSs) refer to the subgroup of Novelty-Driven Systems (NDSs). In contrast to the general group of spike-based neuromorphic systems (Section 2.3.1), which use all neural coding principles (Section 2.1.3), NDSs focus on sparse information representation, that is, neural time coding, and follow the idea that only novel information should drive a system. Novelty-driven sensing, in particular, has been addressed in vision [86, 120, 122] and force sensing [27].

Novelty-driven systems relate very well to one core statement in information theory [132]. In many applications, following the guideline of the Nyquist-Shannon sampling theorem, that is, realizing CDSs, results in a stream of samples containing a large amount of uncontrolled redundant information. Temporally redundant information is especially apparent when the system continuously samples the same value. Systems can avoid temporal redundancy when they only handle novel information, that is, when they are only active when sensors register activity. Shannon's information entropy and his source coding theorem formally describe the information rate of information sources, and thus how much information, or respectively redundancy, a signal contains [132]. The information entropy $H(X)$ evaluates the probabilities $\mathrm{P}(x_i)$ of symbols x_i, which encode the information produced by the information source $x(t)$

$$H(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_2 \mathrm{P}(x_i) \tag{2.2}$$

and is measured in bits. Thus, if an information source $x(t)$ continuously emits the same signal level, then all probabilities $\mathrm{P}(x_i)$ beside one are zero, and the information entropy $H(X)$ is zero. Consequently, the signal does not contain information, and repeatedly sampling it only produces uncontrolled redundancy and wastes resources. On the other hand, if $x(t)$ is continuously changing, then the probabilities $\mathrm{P}(x_i)$ are more distributed, and the information entropy $H(X)$ is well beyond zero. Thus, sensors that register substantial changes in $x(t)$ produce a considerable amount of information.

In summary, novelty corresponds to activity, whereby changes express it. Thus systems, that are driven by novelty and where events solely express novel information avoid uncontrolled temporal redundancy throughout all stages. These systems gain efficiency because events represent information more sparsely, and the system has to transmit and process less.

2.4. Comparative Survey of Event Representations and Protocols

This section surveys the most notable approaches towards Event-Driven Systems (EDSs) (Section 2.3.3), namely the neuromorphic Address-Event-Representation (AER) [90, 22], the Send-on-Delta Principle (SoDP) [114, 97, 99], and, more recently, the Asynchronous Encoded Skin (ACES) [85]. These EDSs have been used in applications with e-skin [28, 12, 85], supporting their effectiveness and efficiency. However, none of these event-driven e-skin approaches fully consider the implications and challenges of effective deployment over large areas and its eventual system integration (Section 2.2). They either lack the flexibility and modularity for the distributed deployment of sensors [28, 12], or they lack an efficient interface to standard computer systems [85], whereby they obstruct system integration.

The subsequent study first introduces the different event-driven approaches and then discusses their applicability towards realizing an effective novelty-driven event handling system for LASSs.

2.4.1. Address Event Representation (AER)

The Address-Event-Representation (AER) [90, 22] is one of the first bio-inspired systems developed for representing and conveying events in technical systems. Originally, AER has been developed for the communication between spiking artificial neurons in VLSI ICs (Very Large Scale Integrated Circuits) [90, 22] and rigorously takes advantage of high-speed digital asynchronous parallel bus systems that are readily available on such devices. AER realizes event-driven point-to-point connections between event generators and consumers. But instead of encoding the information source by individually wiring each event generator to an event consumer, as nature does, the AER employs addresses to identify sources and time-multiplexes these addresses onto a common asynchronous parallel bus. A valid address on this bus represents an event and this address identifies the event generator of that event. The AER exploits the superior communication speed of technical systems per wire in integrated systems ($>$ 100 MBit/s) in comparison to nerve fibers ($\approx$ 1 kBit/s) to reduce the number of wires and still achieve a comparably high temporal resolution. Besides the address bus, the AER employs a request and an acknowledge line to realize a self-timed bus arbitration mechanism that avoids any clock resynchronization. AER represents events through addresses. To convey information, AER event generators can employ encoding principles that are similar to neural codes. The AER can encode the type of events in additional address lines such that an AER event generator can create change events, events that indicate an increase or decrease of the observed signal (up and down events), respectively [86, 120]. To encode absolute values instead of increments, AER event generators can employ low and high events where the time between these two events represents the encoded value [121].

The work presented in Section 2.4 was in part published in:
Bergner, F., Dean-Leon, E., Cheng, G., "Design and Realization of an Efficient Large-Area Event-Driven E-Skin". In: *Sensors* 20.7 (2020), p. 1965.
Copyright permissions: see Appendix D.

More recent research introduced serial AER [126] to reduce the wiring complexity in more distributed EDSs [12]. Serial AER packs the event address into a datagram on a serial bus, which reduces the number of wires at the cost of reducing the temporal resolution.

The AER is an established bio-inspired protocol for representing and transferring events and could successfully demonstrate its use in auditory [130], visual [86, 120], and force [28] sensing applications and in event-driven processing hardware such as SpiNNacker [51], TrueNorth (IBM) [67], BrainScaleS [95], ROLLS [123], DYNAP [110], Loihi (Intel) [38], and BrainDrop [113].

2.4.2. Send-on-Delta Principle (SoDP)

The Send-on-Delta Principle (SoDP) [114, 97, 99] is a hybrid system that exploits standard digital hardware to realize EDSs. The SoDP has been first proposed for efficiently reducing the number of transmissions in wireless, battery-powered, and widely distributed sensors networks [114, 97]. In these application scenarios, the reduction of the number of transmissions is essential to increase the lifetime of the distributed sensors. SoDP systems employ time-discrete digital change detectors, that can even be implemented in software, to trigger the creation of events. The SoDP represents events by packets (event packets) transported in asynchronous arbitrated networks. An event packet usually contains the ID of its information source and the absolute value of the signal at its creation time. Similar to the AER, the presence of an event packet signifies the availability of novel information and drives the information handling of the system. Since the SoDP not only conveys the information source but also the magnitude of the signal, the bit rate of SoDP events is higher than the bit rate of AER events. As a consequence, the temporal resolution of SoDP systems is lower than that of AER systems when both systems employ the same transmission rate. Nevertheless, decoding the information of SoDP events is by far less complex than for AER events whenever an application enforces clock-driven information, for example, in low-level control of closed hardware/software systems, such as in robots. Furthermore, SoDP systems do not require specialized hardware and are realizable with off-the-shelf sensors and well-established information transport layers [114, 97]. Thus, if CDSs possess the flexibility to modify their information handling procedures, and if they can employ asynchronous transmission and processing capabilities, then these systems can be turned into EDSs without the need for any hardware modifications.

The SoDP provides good flexibility and availability for realizing low cost and scalable event-driven applications [114]. However, the hardware of SoDP systems is clock-driven such that SoDP cannot reach the temporal precision and energy efficiency of systems that employ event-driven neuromorphic hardware.

2.4.3. Asynchronous Encoded Skin (ACES)

The Asynchronous Encoded Skin (ACES) [85] is an event-driven hardware system that has been recently proposed to realize a neuro-inspired artificial nervous system. The ACES implements a many-to-one protocol for transmitting and representing events. Rather than time-multiplexing events to a shared transportation medium, such as in the AER or SoDP, the ACES fuses events as pulse signatures onto one single shared wire. A pulse signature is a sequence of pulses within a constant time window, where the relative timing of the pulses encodes the signature. Similar to the addresses in AER and the IDs in SoDP, the pulse signature identifies the information source and represents the event. Interestingly, the ACES manages to fuse these pulse signatures on one single wire without applying time-multiplexing or requiring an arbitration method. The ACES superimposes all pulse signatures by applying a logical OR operation on the pulses. To minimize the probability that pulse signatures cannot be separated, the set of pulse signatures has to have minimal auto-correlation and cross-correlation. Theoretically, ACES could support up to 138 000 information sources per wire, when a pulse signature has a time window of 1 ms, consists of 10 pulses, and each pulse lasts for 100 ns [85]. In such a setup, ACES events have a latency of at least 1 ms when employing up and down events, or respectively a latency of at least 2 ms when employing low and high events for time-coding absolute values [85]. However, the temporal precision of ACES is very high (in the range of the pulse length) since no arbitration mechanisms impair the temporal precision with non-deterministic uncertainties in delay, which correlate with the utilization of a shared communication medium. Additionally, since the ACES event transmission is arbitration-less, connection redundancy could be introduced by adding wires, as long as the propagation speed and the reflection of high-speed connections do not degrade the transmission quality. While the hardware for encoding and representing events in ACES has low complexity, acquiring a set of pulse signatures is more demanding, and the demerging of ACES events is very complex. An ACES event demerger has to repeatedly correlate the currently observed pulse pattern of superimposed events with all pulse signatures of the set. Therefore, the ACES event demerger has to keep a history of received pulses, which matches the length of a pulse signature. To preserve the temporal information of the events, the demerger has to perform this correlation continuously for each potential event in parallel within the time length of a pulse. For the example numbers mentioned earlier, the demerger would at least have to perform continuously 138 000 correlations with a bit length of 100 000 bit (assuming a pulse can be represented by one bit) within 100 ns. The ACES event decoder is clearly not event-driven since the decoding has to be driven by the pulse time, and the information in the superimposed pulse stream is not salient. Nevertheless, the demerged events can drive information handling in subsequent stages.

2.4.4. Comparative Study of Representations and Protocols

The following study analyzes the characteristics of the previously introduced event representations and protocols to assess their applicability in Large-Area Skin Systems (LASSs). The ideal approach towards an EDS for LASSs solves the remaining challenge of latency and efficiency without obstructing existing solutions and mitigations for robustness, deployability, wiring complexity, and sensor poses (Section 2.2.2). Therefore, the EDS should support the central principles of modularity and self-organization (Section 2.2.3).

Table 1 summarizes the relevant properties of existing implementations for EDSs. The table additionally contains the properties of nerve bundles and CDSs to provide the biological and standard technical references for comparisons.

System	Connections	Bandwidth	Arbitration	Representation	Encoding	Decoding	Bit Rate	Latency	Temporal Precision
Nerve Bundle [133, 54]	thousands of nerve fibers	low, $\leq$ 1 kEvents/s per fiber	none	binary action potential (1 bit) asynchronous events	neural codes	complex	very low	medium	extremely high
Clock-Driven Section 2.3.2	few, serial bus	very high, $>$ 10 MSamples/s	standard network routing, time-multiplexed, flexible	samples / packets of samples (many bits) synchronous stream	absolute values	none	continuously very high	medium/ high	low
AER [90, 22]	many, parallel bus	high, $\sim$ 100 MEvents/s	complex handshaking, time-multiplexed, inflexible	address (address bits) asynchronous events	neural codes	complex	low	low	high
Serial AER [126]	few, serial bus	high, $\sim$ 5 MEvents/s	complex handshaking, time-multiplexed, inflexible	serialized address asynchronous events	neural codes	complex	low	higher than AER	lower than AER
ACES [85]	1 wire, upto 138k sources	low, $\sim$ 1 kEvents/s	none	pulse signature asynchronous events	neural codes	very complex, not event-driven	low	medium	very high
SoDP [114, 97, 99]	**few**, serial bus	medium, $>$ 1.5 MEvents/s	standard network routing, time-multiplexed, **flexible**	event packet (sample bits) asynchronous events	absolute values	**simple**	medium	**low**	medium

Table 1 Comparison of different information encoding, representation, and transmission systems. The first row provides the biological reference, and the second row the stand technical one. Besides the first row, all other rows present electronic systems.

The connection, bandwidth, and arbitration of EDSs have a significant impact on the system's flexibility and performance. The ideal EDS for LASSs is flexible. That is, it does not depend on specific hardware and its components (for instance, sensors, connections, event consumers) can be easily added, removed, and exchanged. Then, the system is modular. Additionally, the distributed character of LASSs and the required robustness ask for an EDS with a small number of wires in connections and a communication protocol supporting self-organization and dynamic routing. At the same time, the ideal EDS fully supports the event-driven mechanism, which delivers the required gain in efficiency and latency. Furthermore, the ideal EDS encodes information with a low encoding/decoding complexity. A low encoding/decoding complexity significantly contributes to the application flexibility of the LASS. It allows for efficient bridges between clock-driven and event-driven systems that are required in many applications utilizing LASSs.

As Table 1 depicts, neither of the existing EDSs (AER, SoDP, and ACES) fulfills all these characteristics for the optimal integration of the event-driven approach in LASSs. The AER and ACES provide excellent performance with very low bit rates in communication and high temporal precision. However, their encoding/decoding complexity is high and impedes the efficiency of the required interfaces to clock-driven systems in LASSs. Furthermore, both systems require special hardware and thus hinge on overall deployment. The systems additionally obstruct the LASS's methods for robustness, wiring, and self-organization. The lower performance of the SoDP (especially regarding bit rate and temporal precision) in comparison to AER and ACES is more than compensated by its flexibility, low encoding/decoding complexity, and hardware independence. The overall performance of SoDP is well beyond CDSs such that a significant performance gain of the resulting event-driven LASSs can be expected. The clear advantage of SoDP lies in its great flexibility. It does not depend on specific hardware and can thus exploit standard hardware for the rapid realization of complex but yet efficient EDSs.

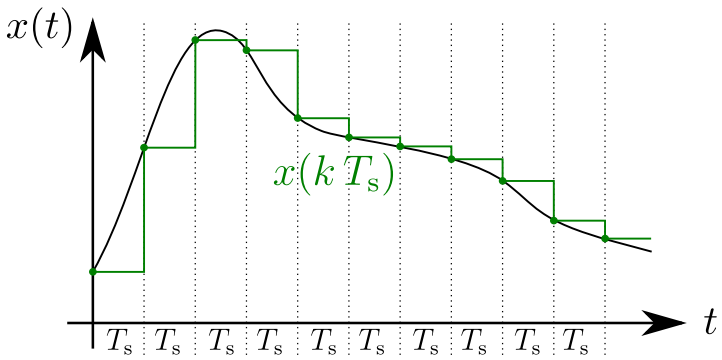
In summary, SoDP emerges as the most suitable EDS towards tackling all the challenges of LASSs. It has the potential for realizing an event-driven LASS capable to efficiently and effectively handle its large amount of tactile information in complex systems in real-time.

2.5. Summary

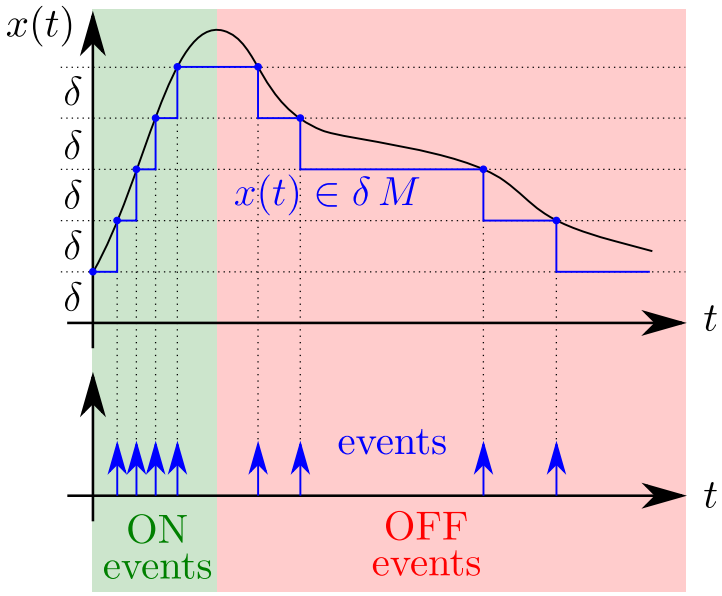
The human sense of touch inspired the development of various realizations of e-skin systems. These e-skin systems successfully proved their usefulness in various applications, such as grasping and physical interaction. Up to now, implementations and applications of e-skin systems are limited to rather small areas, that is body parts, for instance, arms. This limitation is not because these e-skin systems lack scalability regarding their sensing capabilities. These systems rather lack a systematic approach to efficiently and effectively handle a large amount of tactile information in real-time. On the other hand, works targeting other sensing modalities such as vision successfully reported a significant boost of performance when implementing bio-inspired approaches that allow for the event-driven handling of information. However, the distributed nature of large-area tactile sensing so far limited the effective realization of event-driven large-area e-skin systems, and thus large-area physical interactions building upon tactile feedback. Nevertheless, the survey of the most prominent event-driven approaches and their implementations presented in this chapter identified the flexible and hardware-independent SoDP as a good candidate to eventually turn standard clock-driven e-skin systems to efficient event-driven large-area e-skin systems.

3. Fundamental Elements of the Event-Driven Approach

This chapter presents the three essential elements of the event-driven approach: i) event generation (Section 3.1): generating events for event-driven systems by the means of novelty detectors; ii) event representation and communication (Section 3.2): representing and conveying information by events in event-driven communication systems; and iii) event-driven information handling (Section 3.3): handling information in systems such that they are driven by novelty rather than the clock. These three elements provide the fundamentals of the makeup of this thesis. Figure 2 provides an example of information flow and highlights the differences in representing information in Clock-Driven Systems (CDSs) and in Event-Driven Systems (EDSs).



(a) Clock-driven systems sample signals $x(t)$ with constant time intervals T_s. Information in clock-driven systems is represented by the samples $x(t_k) = x(k\,T_\mathrm{s})$.



(b) Event-driven systems detect changes δ in monitored signals $x(t)$. Between two events the value of $x(t)$ changed at most by δ. The generated events could be tagged with type information such as ON/OFF. Here, ON events represent an increase by δ and OFF events an decrease by δ.

Figure 2 Information as it is represented in clock-driven systems (Figure 2a) and event-driven systems (Figure 2b).

This chapter provides the theoretical basis for developing and validating the event-driven approach towards large-area e-skin systems that are to be presented in the upcoming chapters.

3.1. Event Generation / Event-Driven Sensing

This section's focus is on novelty-driven event generation, which bases on change detectors. Change detectors detect novelty in signals and couple system activity with the information rate of information sources, see Section 2.3.3. This section provides the theoretical basis for the correct design and optimal parameterization of novelty detectors for the e-skin's sensors to optimize for sensitivity, transmission rate, minimizing encoding errors, and noise.

First, Section 3.1.1 introduces the theory of and formalisms for change detectors. Then, Section 3.1.2 details different general approaches to implementing change detectors. After that, Section 3.1.3 provides the theory for estimating the event rates of change detectors and connects the findings to signal processing theory. This theory results in descriptions that relate the signal shape and system properties to event rates. The outcomes of Section 3.1.3 leads to insights in the sensitivity of change detectors (Section 3.1.4), the conversion error of change detectors (Section 3.1.5), and the influence of noise (Section 3.1.6). Combining all these results lead to guidelines to correctly parameterize change detectors for event generators (Section 3.1.7).

3.1.1. The Theory of Change Detectors

This section presents *novelty detection*, that is, on the procedure and formalisms to decide when signals, for example, of sensors, provide valuable information. Since the amount of information a signal provides correlates with the magnitude and frequency of its changes, see Section 2.3.3, a *novelty detector* is a *change detector* that triggers activity, or respectively the generation of events. Novelty-Driven Systems (NDSs) and change detectors have been introduced in two different points of view: 1) in the bio-inspired neuromorphic point of view [90, 22, 130, 86, 120, 88, 121, 28], and 2) in the information and control theory point of view [114, 97, 99].

These introductions of change detectors in both fields lead to similar formalisms. Neuromorphic systems [86, 120, 121, 122, 26] use time- and range-continuous change detectors to convert signals $x(t)$ to events $\mathbf{e}_i$. The change detectors of the other field [114, 97, 99], with more focus on information and control theory, use time-discrete change detectors to convert sampled signals $x(t_k)$ with a sample rate of $f_\mathrm{s} = 1/T_\mathrm{s}$ at time instances $t_k = k\,T_\mathrm{s}$, $k \in \mathbb{Z}$ to events $\mathbf{e}_i$.

The work presented in Section 3.1.1 was in part published in:
Bergner, F., Dean-Leon, E., Cheng, G., "Design and Realization of an Efficient Large-Area Event-Driven E-Skin". In: *Sensors* 20.7 (2020), p. 1965.
Copyright permissions: see Appendix D.

3.1.1.1 Continuous Change Detectors

Continuous change detectors monitor a signal $x(t)$ and track the change of this signal until the accumulated change exceeds a predefined threshold δ. Therefore, the change detector integrates the derivative $\dot{x}(t)$ of the input signal $x(t)$ until the integration reaches or passes the threshold δ

$$\delta \leq \left| \int_{t_{i-1}}^{t} \dot{x}(t) \, \mathrm{d}t \right| \qquad \Longleftrightarrow \qquad \underbrace{\mathbf{e}_i \quad \text{at} \quad t = t_i = t(\mathbf{e}_i)}_{\text{event generation}} \tag{3.1}$$

at time instance $t = t_i$. At the time instance t_i, the information of the monitored signal is classified as novel and the change detector triggers the creation of an event $\mathbf{e}_i$ that contains this novel information. The more precise the occurrence time t_i of the event matches with the time instance of the actual signal change, the higher the *temporal precision* of the event generator is. Thus, any non-deterministic or non-constant delay between the actual signal change and the occurrence of the event reduces the temporal precision. Considering the properties of Riemann integrals

$$\int_a^b f(t) \, \mathrm{d}t = \lim_{\Delta t \to 0} \sum_{i=1}^{n} f(t_i) \Delta t \tag{3.2}$$

with

$$\Delta t = \frac{b-a}{n} \, , \qquad t_i = a + \frac{\Delta t}{2} \, n \tag{3.3}$$

and the definition of the arithmetic average of a function

$$\left[\overline{f(t)} \right]_a^b = \lim_{\Delta t \to 0} \frac{1}{n} \sum_{i=1}^{n} f(t_i) \tag{3.4}$$

we can derive a relationship between an integral of a signal $f(t)$ and its average $\overline{f(t)}$ in an interval $t \in [a, \, b]$:

$$\left[\overline{f(t)} \right]_a^b = \frac{1}{b-a} \int_a^b f(t) \, \mathrm{d}t. \tag{3.5}$$

Combining Equations (3.1) and (3.5) yields

$$\delta \leq \left| \left[\overline{\dot{x}(t)} \right]_{t_{i-1}}^{t} \right| \cdot (t - t_{i-1}). \tag{3.6}$$

Therefore, one could observe that the change detector evaluates the accumulated average change of the input signal since the occurrence of the last event $\mathbf{e}_{i-1}$ at time instance t_{i-1}. With

$$\overline{\dot{x}}_i = \left[\overline{\dot{x}(t)} \right]_{t_{i-1}}^{t} = \frac{1}{t - t_{i-1}} \int_{t_{i-1}}^{t} \dot{x}(t) \, \mathrm{d}t = \frac{x(t) - x(t_{i-1})}{t - t_{i-1}} \tag{3.7}$$

Equation (3.1) further simplifies to:

$$\delta \leq |x(t) - x(t_{i-1})| \qquad \Longleftrightarrow \qquad \mathbf{e}_i \quad \text{at} \quad t = t_i \tag{3.8}$$

or respectively

$$\boxed{\delta \leq |x(t(\mathbf{e}_i)) - x(t(\mathbf{e}_{i-1}))|} \tag{3.9}$$

Thus, a change detector in fact triggers the creation of events whenever the difference between the signal $x(t_{i-1})$ that caused the last event $\mathbf{e}_{i-1}$ and the currently monitored input $x(t)$ exceeds a specified limit δ. Because the continuous change detector can trigger events at any time, its temporal resolution is only limited by the bandwidth of the system that implements it. As a result, a continuous change detector can theoretically achieve an almost infinite equivalent sampling rate.

3.1.1.2 Discrete Change Detectors

Discrete change detectors monitor the samples $x(t_k)$ with $t_k = k\,T_\mathrm{s}$, $k \in \mathbb{Z}$ of a continuous signal $x(t)$. The underlying principle for detecting changes is similar to the continuous change detector (Section 3.1.1.1). A discrete change detector also integrates the derivative $x(t_k)$ of the sampled input signal $x(t_k)$. But in contrast to the continuous detector, the integration process is digital and clock-driven. The integration continues until it passes the threshold δ

$$\delta \leq \left| \int_{t_{K_{i-1}}}^{t_k} \dot{x}(t)\,\mathrm{d}t \right| \qquad \Longleftrightarrow \qquad \underbrace{\mathbf{e}_i \quad \text{at} \quad t_k = t_{K_i} \quad \text{with} \quad k = K_i}_{\text{event generation}} \tag{3.10}$$

at time instance $t_k = t_{K_i}$ or respectively at the sample $k = K_i$. At time instance t_{K_i}, the sample is classified as novel and the change detector triggers the creation of an event $\mathbf{e}_i$. Similarly to Equation (3.5), we can derive a time-discrete relationship between the average and the integral of a signal

$$\begin{aligned} \left[\overline{f(t)}\right]_{t_{K_1}}^{t_{K_2}} &= \frac{1}{t_{K_2} - t_{K_1}} \int_{t_{K_1}}^{t_{K_2}} f(t)\,\mathrm{d}t = \frac{1}{t_{K_2} - t_{K_1}} \sum_{l=K_1}^{K_2} f(t_l)\,T_\mathrm{s} \\ &= \frac{1}{K_2 - K_1} \sum_{l=K_1}^{K_2} f(t_l), \end{aligned} \tag{3.11}$$

which combined with Equation (3.10) leads to

$$\delta \leq \left| \left[\overline{\dot{x}(t)}\right]_{t_{K_{i-1}}}^{t_k} \right| \cdot (t_k - t_{K_{i-1}}). \tag{3.12}$$

With

$$\bar{\dot{x}}_i = \left[\overline{\dot{x}(t)}\right]_{t_{K_{i-1}}}^{t_k} = \frac{1}{t_k - t_{K_{i-1}}} \int_{t_{K_{i-1}}}^{t_k} \dot{x}(t)\,\mathrm{d}t = \frac{x(t_k) - x(t_{K_{i-1}})}{t_k - t_{K_{i-1}}} \tag{3.13}$$

Equation (3.10) then simplifies to:

$$\delta \leq \left|x(t_k) - x(t_{K_{i-1}})\right| \qquad \Longleftrightarrow \qquad \mathbf{e}_i \quad \text{at} \quad t_k = t_{K_i} \quad \text{with} \quad k = K_i \tag{3.14}$$

or respectively

$$\boxed{\delta \leq |x(t(\mathbf{e}_i)) - x(t(\mathbf{e}_{i-1}))|} \tag{3.15}$$

which is identical to Equation (3.9) of the continuous change detector. Thus, the discrete change detector has to remember the signal sample $x(t_{K_{i-1}})$ of the previous event $\mathbf{e}_{i-1}$ and compare it to the current sample $x(t_k)$. Then, at the time when the absolute difference between these two samples exceeds the threshold, the change detector triggers the creation of the event $\mathbf{e}_i$ and updates the memory with the current sample. In comparison, discrete change detectors naturally exhibit a lower temporal resolution than their analog counterparts and consume more power since the monitoring is clock-driven. The temporal resolution of discrete change detectors is limited by the sampling rate of the digital system. Nevertheless, discrete change detectors do not require special analog circuits and allow for integrations in existing systems, for instance, in the microcontrollers that sample sensors.

3.1.1.3 Change Detectors as Supervisors of Predictors

This section provides a new point of view on change detectors that allows deriving advanced event encoding schemes and has the potential to lead to more sophisticated event generators with enhanced robustness and error correction features.

In general, a change detector can be interpreted as a supervisor that ensures that the error between the input signal $x(t_k)$ and its prediction $\hat{x}(t_k)$ is below a specified limit δ [137, 98]. This interpretation allows for generalizing the change detection rule of (3.15) to:

$$\delta \leq |x(t_k) - \hat{x}(t_k)| \qquad \Longleftrightarrow \qquad \underbrace{\mathbf{e}_i \quad \text{at} \quad t_k = t_{K_i}}_{\text{event generation}} \tag{3.16}$$

The change detector produces a new event $\mathbf{e}_i$ at time t_{K_i} whenever the error between prediction and the actual signal exceeds δ. A good choice for a predictor $\hat{x}(t_k)$ may be a n-th-order Taylor approximation of the signal $x(t)$ that is evaluated at the time instance $t_{K_{i-1}}$ of the previous event e_{i-1} with $L = K_{i-1}$ [137, 98]:

$$\hat{x}(t_k) = \sum_{l=1}^{n} \frac{x^{(l)}(t_L)}{l!} (t_k - t_L) \tag{3.17}$$

The zero-order/constant, first-order/linear, and second-order/quadratic predictors are

$$\hat{x}_{\mathrm{const}}(t_k) = x(t_L) \tag{3.18}$$
$$\hat{x}_{\mathrm{lin}}(t_k) = x(t_L) + \dot{x}(t_L)\,(t_k - t_L) \tag{3.19}$$
$$\hat{x}_{\mathrm{quad}}(t_k) = x(t_L) + \dot{x}(t_L)\,(t_k - t_L) + 0.5\,\ddot{x}(t_L)\,(t_k - t_L)^2 \tag{3.20}$$

with the following Euler approximations for the derivatives:

$$\dot{x}(t_L) \approx \frac{x(t_L) - x(t_{L-1})}{T_{\mathrm{s}}} \tag{3.21}$$
$$\ddot{x}(t_L) \approx \frac{x(t_L) - 2\,x(t_{L-1}) + x(t_{L-2})}{T_{\mathrm{s}}^2}. \tag{3.22}$$

Substituting k with $l = k - L$ such that $l = 0$ when $k = L$, that is at the time instance of the previous event e_{i-1}, yields

$$\hat{x}_{\mathrm{const}}(t_{L+l}) = a \tag{3.23}$$
$$\hat{x}_{\mathrm{lin}}(t_{L+l}) = a + b\,l \tag{3.24}$$
$$\hat{x}_{\mathrm{quad}}(t_{L+l}) = a + b\,l + 0.5\,c\,l^2 \tag{3.25}$$

with:

$$a = x(t_L) \tag{3.26}$$
$$b = x(t_L) - x(t_{L-1}) \tag{3.27}$$
$$c = x(t_L) - 2\,x(t_{L-1}) + x(t_{L-2}). \tag{3.28}$$

The predictors can be written in recursive form

$$\hat{x}_{\mathrm{const}}(t_l) = a \tag{3.29}$$
$$\hat{x}_{\mathrm{lin}}(t_l) = \hat{x}_{\mathrm{lin}}(t_{l-1}) + b \tag{3.30}$$
$$\hat{x}_{\mathrm{quad}}(t_l) = 2\,\hat{x}_{\mathrm{quad}}(t_{l-1}) - \hat{x}_{\mathrm{quad}}(t_{l-2}) + c \tag{3.31}$$

with the following initializations at $l = 0$:

$$\hat{x}_{\mathrm{lin}}(t_{l=0}) = x(t_L) \tag{3.32}$$
$$\hat{x}_{\mathrm{quad}}(t_{l=0}) = x(t_L) \tag{3.33}$$
$$\hat{x}_{\mathrm{quad}}(t_{l=-1}) = 0.5\,x(t_L) + 0.5\,x(t_{L-2}) \tag{3.34}$$

The interesting observation that one can take from this point of view is that the continuous change detector and its discrete counterparts are change detectors that use a zero-order prediction. Thus, these change detectors can also be seen in the following way: the change detector assumes/predicts that the input signal $x(t)$ will not change, and when the error of this assumption becomes larger than δ, then an update is required and an event is generated. Respectively, a change detector with a first-order predictor assumes that the input signal $x(t)$

will continue with the signal's slope at the time of the previous event, and a change detector with a second-order predictor that $x(t)$ will continue with the signal's curvature at the time of the previous event. The higher order predictors add more context to events by assuming more complex underlying models that rely on more history. Higher order predictors produce less activity but need to transmit more values per activity, that is one value for zero-order, two for first-order, and three for second-order predictors. However, on average, higher order predictors transfer less information [137]. Therefore, the reduction in activity outweighs the larger number of values to transmit. The encoding of context into event sequences, following also different regimes, is discussed with additional examples in Section 3.2.1.

Interpreting change detectors as supervisors of predictors has a large potential for the development of novel change detectors that reach beyond the capabilities of zero-order predictors. For example, a more sophisticated predictor might predict the decay of a signal such that even if events are lost in communication the event decoder will eventually settle down to the most probable constant value. Another predictor could use the slope to create additional events on important transitions such that intended information redundancy can mitigate the effects of loss in situations that are highly important for system stability.

3.1.2. Realizing Change Detectors

While the previous section provided the theoretical background for continuous and discrete change detectors, this section outlines different approaches for their realization in electronic circuits. First, an analog circuit for a continuous change detector is introduced (in Section 3.1.2.1). This circuit is often applied in novelty-driven neuromorphic systems that mimic neural codes and neural computing principles [90, 22, 130, 86, 120, 88, 121, 28]. Second, a digital circuit for a discrete change detector is presented (in Section 3.1.2.2). This change detector finds its application in the field of energy efficient-sensing, signal processing, and control from the information and control theory point of view [114, 97, 99]. Section 3.1.2.3 goes beyond these two approaches and proposes a hybrid change detector with analog and digital circuits. The hybrid change detector merges the advantages of the previously presented approaches and provides a suitable asynchronous change detector for the Send-on-Delta Principle (SoDP) event protocol.

3.1.2.1 Analog/Continuous Change Detectors

Analog change detectors implement the event generation rule of Equation (3.1) employing analog circuits for the differentiation stage, the integration stage, and the absolute comparison stage, see Figure 3. These analog change detectors have been used in novelty driven neuromorphic system in the works of [90, 22, 130, 86, 120, 88, 121, 28].

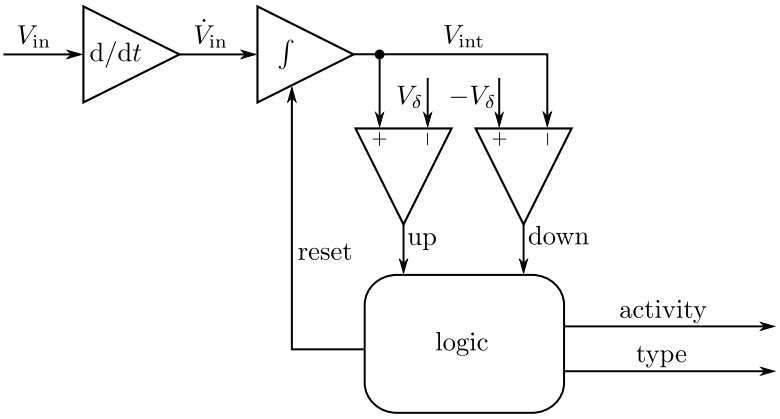


Figure 3 The analog change detector realizes the continuous change detection rule of Equation (3.1). The change detector only employs analog circuits to detect novelty. Two comparators implement the absolute comparison to the event threshold δ and trigger up or down pulses when the monitored input signal $x(t)$ increased or decreased by δ. The voltage V_{in} represents $x(t)$ and the voltage V_δ the threshold δ. The asynchronous logic resets the integrator on the detection of a change. It also creates an activity pulse while the type signal encodes the type of the activity: logic 1 for an increase by δ and logic 0 for a decrease δ.

It is important to note that the analog change detector implements the original Equation (3.1) for change detectors rather than the resulting Equation (3.9). This selection results from the fact that a differentiator and an integrator (Equation (3.1)) are much easier to implement in analog circuits than a memory and a subtraction (Equation (3.9)).

The analog change detector is completely asynchronous and thus does not rely on a system clock in any of its stages. Its circuit directly converts the analog input signal V_{in} into activity pulses and a signal encoding the type of activity. The type encodes if the input signal increased or decreased by δ. Through its asynchronous nature, the analog change detector reaches a high temporal resolution. Its system bandwidth f_{B} is only limited by its analog components and the round trip time from the integrator, to the comparators back to the reset line of the integrator. For instance, the work of [86] reports for their vision sensor a temporal resolution of 15 µs (effectively 66.7 kHz). However, the analog change detector provides only differential information and absolute information has to be reconstructed from the up/down activity. This reconstruction from differential information is challenging for two reasons, namely: 1) initialization, and 2) drift. Initialization is challenging when the input signal cannot be assumed to be zero and drift occurs when activity pules are lost in unreliable communications. Drift may also occur when the input signal violates the bandwidth limitations of the change detector, which will be discussed in detail in Section 3.1.3.2.

3.1.2.2 Digital/Discrete Change Detectors

Digital change detectors implement the event generation rule of Equation (3.15) employing an Analog-Digital Converter (ADC) and digital circuits for memory, the absolute difference, and the inequality, see Figure 4.

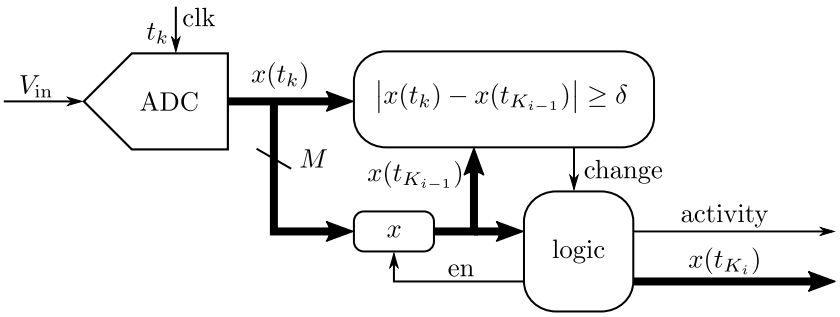


Figure 4 The digital change detector realizes the discrete change detection rule of Equation (3.15). The change detector first converts the time-continuous analog signal to time-discrete quantized samples $x(t_k)$. These samples are continuously evaluated in a logical circuit that computes the absolute difference between the current sample $x(t_k)$ and the sample of the previously reported change event $x(t_{K_{i-1}})$ that is stored in the memory x. When this logic detects a change, it creates a pulse that triggers through a different logic block the update of the memory, the activity pulse, and the output of the sample $x(t_{K_i})$. $x(t_{K_i})$ is the sample that triggered the activity.

The updates of the digital change detector are driven by the samples of the ADC and thus by the sampling clock. The system bandwidth is limited by the sampling clock. Furthermore, the sample values $x(t_k)$ are not continuous since they are discretized by the ADC. The digital system can only operate on discretized values. Thus, the temporal resolution of the digital change detector is limited by the sample rate of the ADC, and the sensitivity is limited by the quantization resolution of the ADC. However, the digital change detector provides absolute values along activity pulses and thus avoids the challenges of initialization and drift as found in analog change detectors (see Section 3.1.2.1).

3.1.2.3 Hybrid Change Detectors

This section proposes a hybrid change detector that combines elements of analog change detectors with elements of digital change detectors to circumvent the bandwidth limitations of the ADC and the differential property of the analog change detector. The hybrid change detector uses the analog change detector to take advantage of its superior change detection capabilities, which include its capability to rapidly convert analog signals to digital activity and type information (increase/decrease). Rather than directly generating up/down events, the hybrid change detector performs local digital integration through the means of an asynchronous counter (see Figure 5).

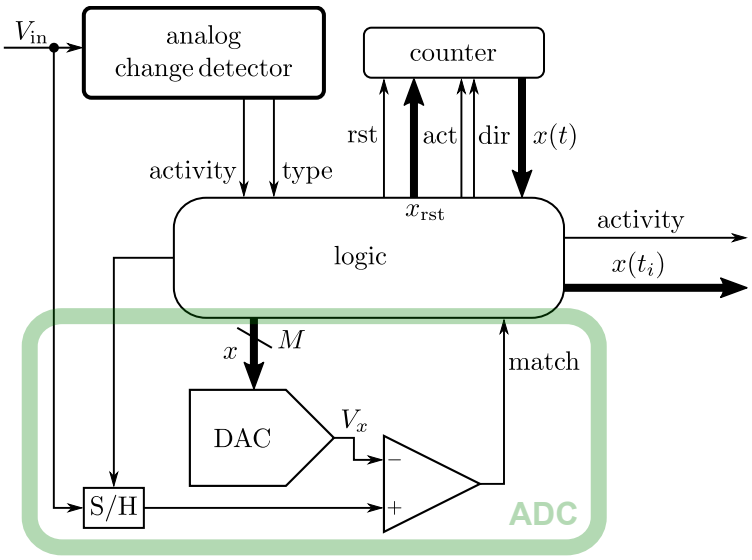


Figure 5 A hybrid change detector consists of an analog change detector and asynchronous digital circuits to locally convert activity driven differential information to offset compensated absolute values. These absolute values are quantized with the resolution of the analog change detector, that is δ, see also Figure 2b. The offset compensation logic comprises a tightly coupled successive approximation ADC (green box) that provides initial values and feedback for drift compensation. The application of low bandwidth ADCs is suitable without any immediate impacts on the system bandwidth of the hybrid change detector. A less tightly coupled but simpler Sigma-Delta ADC is also suitable at the cost of a slower and less flexible compensation response. The logic provides an activity pulse and the absolute value $x(t_i)$ of the input signal.

The analog change detector converters the input voltage V_{in} to activity pulses and differential information. Thus, novelty can drive the integration of the differential information provided by the analog change detector to absolute values. A supervising logic incorporating a low bandwidth ADC provides initial values and feedback to compensate integration offsets.

Overall, the hybrid change detector operates asynchronously and provides absolute values along activity pluses. It provides higher temporal resolution than the digital change detector and can be combined with the SoDP event protocol. The option to utilize a low bandwidth Sigma-Delta ADC has great potential to reduce the circuit complexity of the hybrid change detector in comparison with its digital counter part. It is expected that this reduced circuit complexity combined with novelty-driven activity will contribute to increased energy efficiency in hybrid change detectors.

3.1.3. Estimating Event Rates

The focus of this section is the estimation of event rates in novelty-driven systems. Therefore, this section analyses the impacts of different factors on the rate of events originating from change detectors. Furthermore, this section links novelty-driven systems to signal processing theory by presenting their relations to signal and system bandwidth. The insights gathered in this section provide the basis for the subsequent investigations that analyze the impacts of sensitivity (Section 3.1.4), conversion error (Section 3.1.5), and noise (Section 3.1.6) on the event rate. All together deliver the foundations for the correct and optimal parameterization of change detectors, that is, event generators in Section 3.1.7.

Section 2.3.3 described that novelty-driven event generators directly correlate the event rate f_e with the information rate $\mathrm{H}(X)$ of the sensor or an input signal $x(t)$:

$$f_\mathrm{e} \propto \mathrm{H}(X). \tag{3.35}$$

While the Equations (3.9) and (3.15) of the continuous and discrete change detectors respectively clearly describe the conditions when events are generated, it is not straight forward to determine the event rate of a given input signal $x(t)$. However, to correctly parameterize change detectors, the influence factors on the event rate need to be investigated. A correct parameterization determines the optimal event threshold for the best possible compromise between low event rates, high sensitivity, low encoding errors, and low susceptibility to noise.

The investigation of influence factors on the event rate of change detectors starts in Section 3.1.3.1, which presents a relationship between the slope of the input signal, the event threshold, and the event rate. Then, Section 3.1.3.2 relates the fundamentals of signal theory, namely system bandwidth, and the signal bandwidth, to the maximum signal change between events and the upper bounds for event rates.

3.1.3.1 The Fundamentals of Estimating Event Rates

This section presents the influence of the input signal $x(t)$ and the event threshold δ on the overall average event rate $\overline{f_\mathrm{e}}$ of a change detector. The resulting relationship has been first introduced in the work of [96].

The average event rate $\overline{f_\mathrm{e}}$ describes the number of events that occur per time interval. Therefore, it is defined by:

$$\overline{f_\mathrm{e}} = \frac{n}{\sum_{i=1}^{n} T_{\mathrm{e},i}} = \frac{n}{T_\mathrm{e}} \tag{3.36}$$

where $T_\mathrm{e} = t_n - t_0$ is the time that passed till the occurrence of the n-th event $\mathbf{e}_n$, or respectively, the sum of all the time intervals $T_{\mathrm{e},i}$

$$T_{\mathrm{e},i} = t_i - t_{i-1} = t(\mathbf{e}_i) - t(\mathbf{e}_{i-1}) \tag{3.37}$$

between events. The event generation rule defines a relationship between the event threshold δ, the absolute average slope $\left|\overline{\dot{x}_i}\right|$ of the input signal $x(t)$ between events, and the time $T_{\mathrm{e},i}$

The work presented in Section 3.1.3.1 was in part published in:
Bergner, F., Dean-Leon, E., Cheng, G., "Design and Realization of an Efficient Large-Area Event-Driven E-Skin". In: *Sensors* 20.7 (2020), p. 1965.
Copyright permissions: see Appendix D.

between events in Equations (3.6) and (3.12), or more generally

$$\boxed{\delta \leq \left|\overline{\dot{x}_i}\right| \; T_{\mathrm{e},i}} \,. \tag{3.38}$$

Summing up all these inequalities leads to an estimate for n events

$$n\,\delta \leq \sum_{i=1}^{n} \left|\overline{\dot{x}_i}\right| \, T_{\mathrm{e},i} \quad \Leftrightarrow \quad n \leq \frac{1}{\delta} \sum_{i=1}^{n} \left|\overline{\dot{x}_i}\right| \, T_{\mathrm{e},i}. \tag{3.39}$$

We combine Equation (3.36) and (3.39) and get

$$\overline{f_{\mathrm{e}}} \leq \frac{1}{\delta \; T_{\mathrm{e}}} \sum_{i=1}^{n} \left|\overline{\dot{x}_i}\right| \, T_{\mathrm{e},i} = \frac{1}{\delta \; T_{\mathrm{e}}} \sum_{i=1}^{n} \left| \int_{t_{i-1}}^{t_i} \dot{x}(t) \, \mathrm{d}t \right| . \tag{3.40}$$

Exploiting the triangle inequality, that is,

$$\sum_{i=1}^{n} \left| \int_{t_{i-1}}^{t_i} \dot{x}(t) \, \mathrm{d}t \right| \leq \sum_{i=1}^{n} \int_{t_{i-1}}^{t_i} \left|\dot{x}(t)\right| \, \mathrm{d}t = \int_{t_0}^{t_n} \left|\dot{x}(t)\right| \, \mathrm{d}t \tag{3.41}$$

the estimation of Equation (3.40) further simplifies to [96]

$$\overline{f_{\mathrm{e}}} \leq \frac{1}{\delta \; T_{\mathrm{e}}} \int_{t_0}^{t_n} \left|\dot{x}(t)\right| \, \mathrm{d}t. \tag{3.42}$$

With the Equation (3.5) for the average of signals, this Equation (3.42) yields

$$\boxed{\overline{f_{\mathrm{e}}} \leq \frac{1}{\delta} \, \overline{\left|\dot{x}(t)\right|}} \,. \tag{3.43}$$

The resulting estimation of the average event rate $\overline{f_{\mathrm{e}}}$ in Equation (3.43) bases on two conservative assumptions such that, in reality, $\overline{f_{\mathrm{e}}}$ can be considered as the upper bound for average event rates. First, the triangle inequality presented in Equation (3.41) expresses a conservative assumption. The average of the absolute slope $\overline{\left|\dot{x}(t)\right|}$ overestimates the sum of the absolute averages of slopes between events

$$\overline{\left|\dot{x}(t)\right|} \geq \sum_{i=1}^{n} \left|\overline{\dot{x}_i}\right| , \tag{3.44}$$

whenever the slope $\dot{x}(t)$ changes its sign between events. Second, the system bandwidth f_{B}, or respectively the sample rate f_{s}, limit the time between events $T_{\mathrm{e},i}$ (Equation (3.38)). If the average slope $\left|\overline{\dot{x}_i}\right|$ exceeds the limit defined by the bandwidth of the system f_{B}, that is

$$\left|\overline{\dot{x}_i}\right| > \delta \, f_{\mathrm{B}}, \tag{3.45}$$

then $T_{\mathrm{e},i}$ is underestimated. Thus, $\overline{f_{\mathrm{e}}}$ is then overestimated.

According to Equation (3.43), the average event rate $\overline{f_\mathrm{e}}$ inversely correlates with the threshold δ:

$$\overline{f_\mathrm{e}} \propto 1/\delta. \tag{3.46}$$

under the condition that the input profile $x(t)$ is identical.

3.1.3.2 The Limiting Factor of Bandwidth and Sample Rate

The previous section demonstrated that the event rate f_e is tightly coupled to the average slope $\overline{|\dot{x}(t)|}$ of the monitored signal $x(t)$ and the event threshold δ, see Equation (3.43). Furthermore, the sampling rate f_s limits the event rate f_e, see Equation (3.38) and (3.45). This limitation of the event rate leads to additional errors. The signal $x(t)$ could have changed by more than δ between two samples, or respectively, events. Therefore, it is worthwhile to investigate whether Band-Width Limited Systems (BWLSs) impose limits on event rates and event encoding errors. BWLSs are insofar relevant since all practical systems are bandwidth limited [71, 80, 63]. Even sampled systems relate to BWLSs via the Shannon-Nyquist Theorem. Thus, findings for BWLSs would lead to results for both, time continuous and time discrete event-driven systems.

A BWLS's behavior is comparable to a low-pass filter since signal frequencies beyond its effective bandwidth are damped to virtually zero. Figures 6, 7, and 8 respectively depict the step response, the impulse response, and the gain in the frequency domain of an ideal low-pass filter.

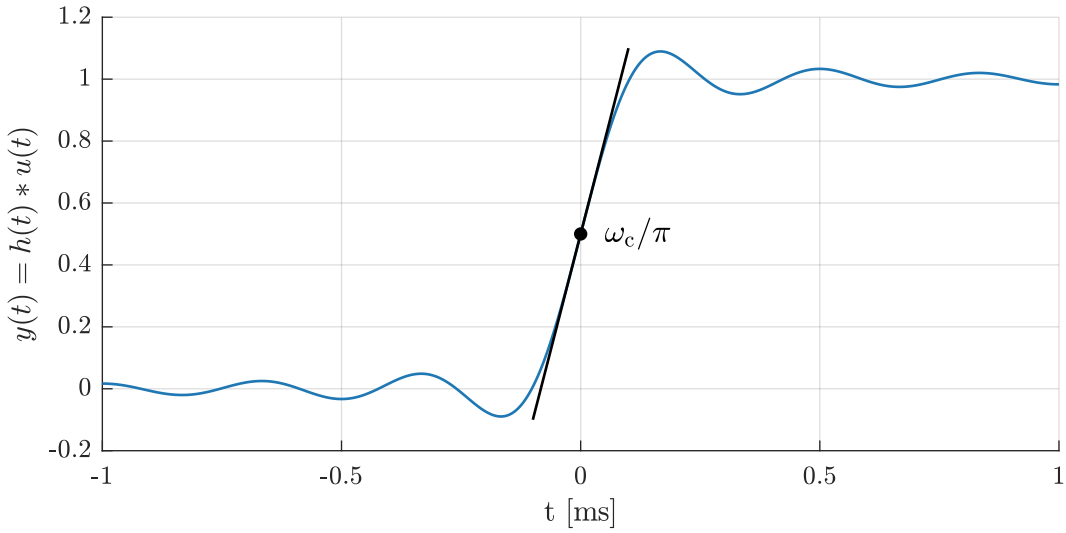


Figure 6 The step response of the ideal low pass filter with a cutoff frequency of $f_\mathrm{c} = 3\,\mathrm{kHz}$. The maximum slope is $\max(\mathrm{d}y(t)/\mathrm{d}t) = \omega_\mathrm{c}/\pi$.

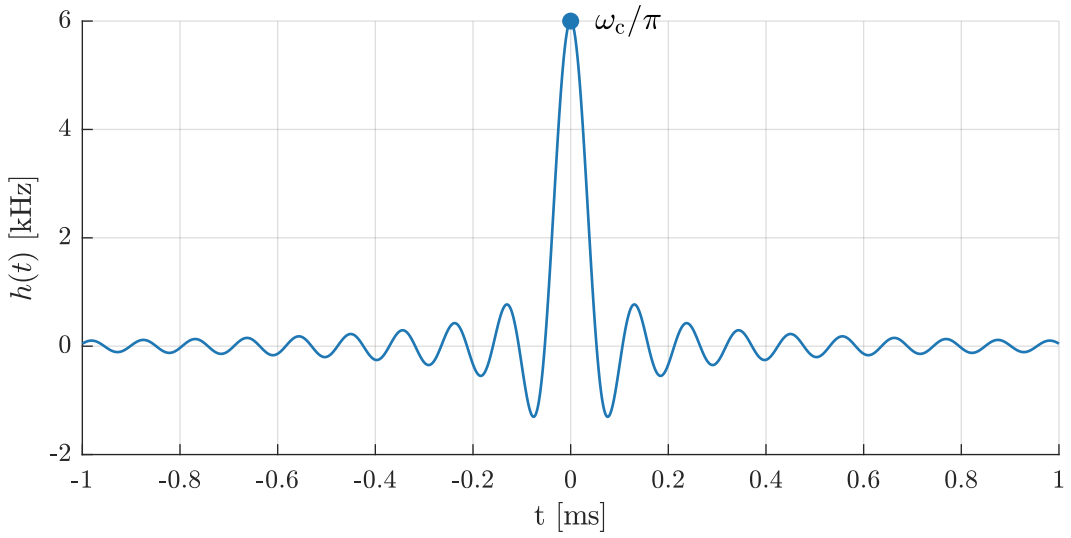


Figure 7 The impulse response of the ideal low pass filter with a cutoff frequency of $f_c = 3\,\text{kHz}$.

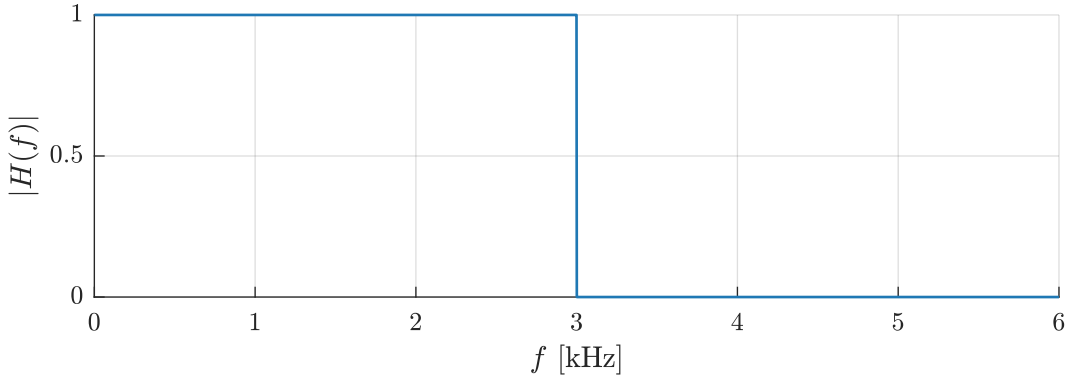


Figure 8 The gain of the ideal low pass filter with a cutoff frequency of $f_c = 3\,\text{kHz}$.

Since the step response of a filter

$$y(t) = h(t) * u(t) = \int_0^t h(t)\,\mathrm{d}t \tag{3.47}$$

just describes the integral of its impulse response, its impulse response analogically describes the derivative of its step response

$$\dot{y}(t) = \frac{\mathrm{d}}{\mathrm{d}t} y(t) = h(t) * \delta(t) = h(t). \tag{3.48}$$

Thus, the maximum slope $\max(|\dot{x}(t)|)$ of an ideal BWLS with a signal range of $x(t) \in [0, x_{\max}]$ is

$$\max(|\dot{x}(t)|) = \frac{\omega_c}{\pi}\, x_{\max} = 2\, f_c\, x_{\max} = 2\, f_B\, x_{\max} \tag{3.49}$$

where f_c is the cutoff frequency of the ideal low-pass filter, or respectively, the bandwidth f_B of the system. While this relationship already provides insights on how the system bandwidth

limits the maximum possible slope of a signal, ideal low-pass filters are non-causal and do not exist in real systems. It would be interesting to know how the relationship between system bandwidth and maximum slope changes for real systems. Figures 9, 10, and 11 compare the step response, impulse response and gain of Butterworth filters of orders one, two, and ten with the ideal low-pass filter.

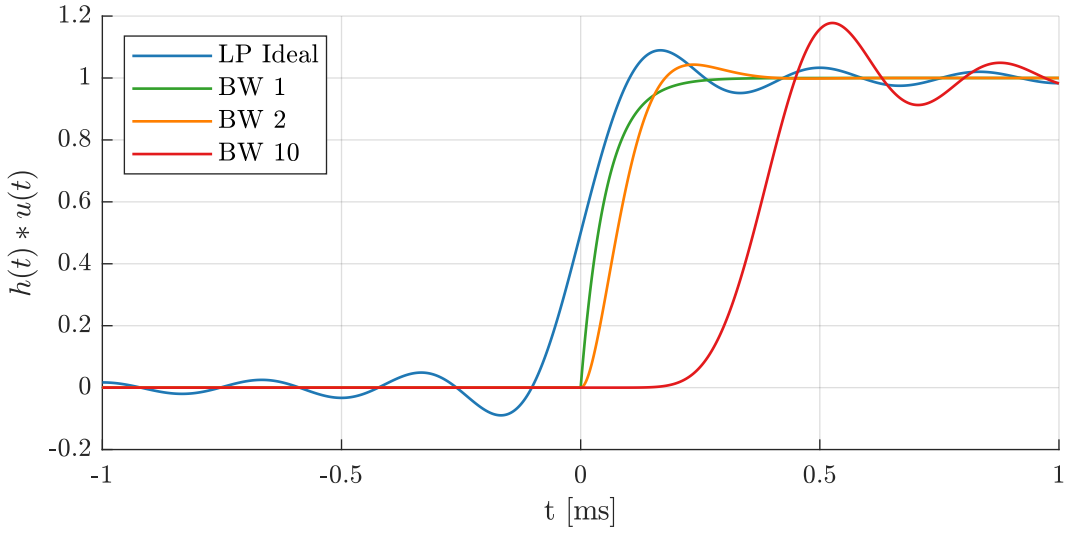


Figure 9 The step responses of the ideal low-pass filter and of Butterworth filters of order 1, 2, and 10. All filters implement a cutoff frequency of $f_c = 3\,\text{kHz}$.

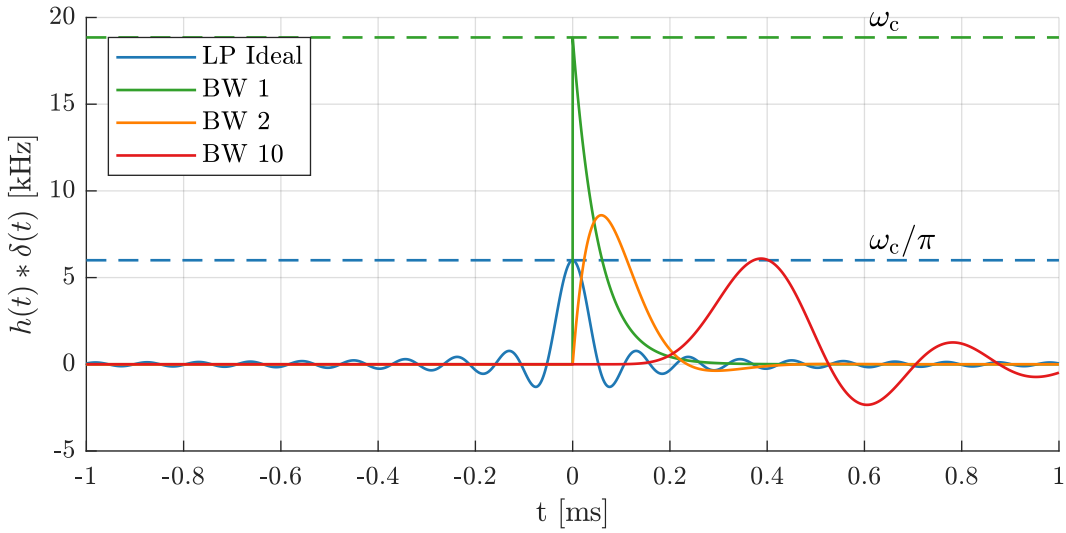


Figure 10 The impulse responses of the ideal low-pass filter and of Butterworth filters of order 1, 2, and 10. All filters implement a cutoff frequency of $f_c = 3\,\text{kHz}$.

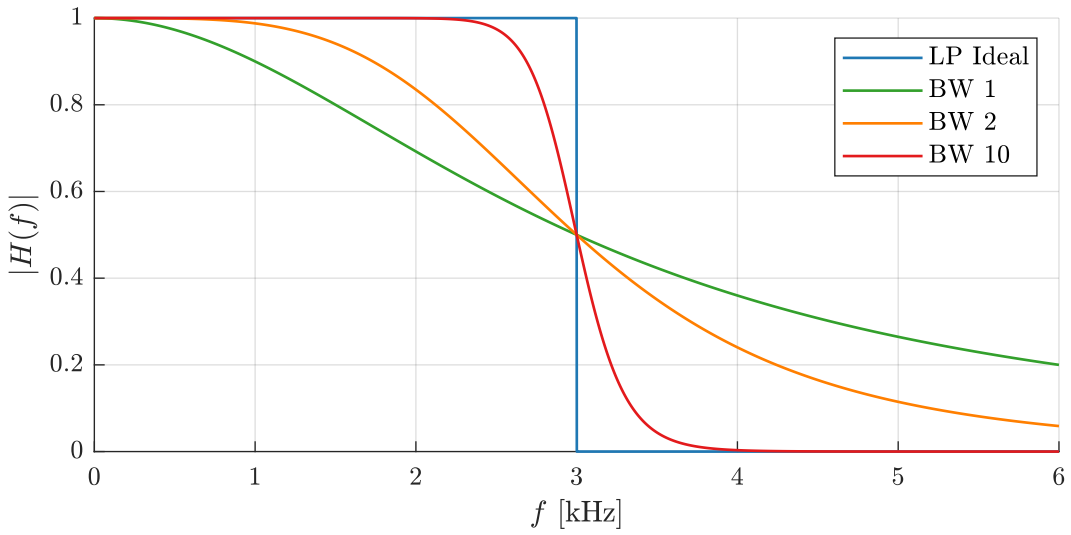


Figure 11 The gain of the ideal low-pass filter and of Butterworth filters of order 1, 2, and 10. All filters implement a cutoff frequency of $f_\mathrm{c} = 3\,\mathrm{kHz}$.

We observe that the range between the worst-case maximum slope (first-order filter) and the best-case maximum slope of the step response range from ω_c to ω_c/π, see Figure 10. The maximum slope of the step response of the second-order filter (namely $\omega_\mathrm{c}\,\mathrm{e}^{-\pi/4} \approx 1.43\,\omega_\mathrm{c}/\pi$) is already close to the best-case, while the tenth-order filter practically matches the base-case. The more realistic but less narrow filters, or respectively system bandwidths, account for the larger maximum slopes since higher frequencies are not instantly attenuated.

The relationship between system bandwidth and maximum slope enables us to derive relationships between system bandwidth, maximum event rate, resolution, and error bounds. The system bandwidth and the resulting maximum signal slope couples to the event threshold δ and the inter-event time $T_{\mathrm{e},i}$ via Equation (3.38).

The first implication is that the time between events $T_{\mathrm{e},i}$ is bounded by the system bandwidth f_B. The change detector cannot react faster than the response time of the system, that is the event rate f_e is bounded by the system bandwidth f_B

$$f_\mathrm{e} \leq f_\mathrm{B}. \tag{3.50}$$

Since the system bandwidth f_B equals the sampling rate f_s in sampled systems, the event rate is analogically bounded by

$$f_\mathrm{e} \leq f_\mathrm{s}. \tag{3.51}$$

The second implication is that the maximum observable slope $\max(|\dot{x}(t)|)$ of a signal $x(t)$ entering the change detector is determined by the bandwidth $f_{\mathrm{B,x}}$ of this signal. The upper bound of $f_{\mathrm{B,x}}$ is naturally half the system bandwidth (Shannon-Nyquist) of the change de-

tector. Thus, considering that in the worst case the average slope $|\overline{\dot{x}_i}|$ between two events equals $\max(|\dot{x}(t)|)$ and that the time between events $T_{\mathrm{e},i}$ is bounded by $f_{\mathrm{B,x}}$, see Equations (3.38) and (3.50), we get a lower bound $\delta_{\min}$ for the event threshold δ

$$\delta \geq \delta_{\min} = \frac{\max(|\dot{x}(t)|)}{f_{\mathrm{B}}}. \tag{3.52}$$

Assuming that the signal $x(t)$ is ideally bandwidth limited (ideal low pass filter) the lower bound for the absolute change $\delta_{\min}$ between events is

$$\boxed{\delta \geq \delta_{\min} = \frac{2\, f_{\mathrm{B,x}}\ (x_{\max} - x_{\min})}{f_{\mathrm{B}}}} \tag{3.53}$$

where $x_{\min}$ and $x_{\max}$ are the lower and upper bounds of the signal $x(t) \in [x_{\min}, x_{\max}]$. Equation (3.52) describes that the system bandwidth impacts the lower bound of the event threshold. When this lower bound is held, the average absolute change between events is at most δ. When the lower bound is violated, the change between events is larger than δ in the cases where the system bandwidth limits the change detector.

The lower bound for the absolute change between events, and with it, the choice of the event threshold δ impacts the sensitivity and the encoding error of the change detector. Both these properties will be discussed in the following sections. Nevertheless, the derived relationship already indicates that the system bandwidth f_{B}, the bandwidth $f_{\mathrm{B,x}}$ of the input signal $x(t)$, and its range impact both, the sensitivity and the encoding error of the change detector.

In summary, the event rate f_{e} is tightly coupled with the event threshold δ and the profile of the input signal $x(t)$. The event rate f_{e} correlates to the information rate via the absolute average slope $|\overline{\dot{x}_i}|$. Furthermore, the system bandwidth f_{B} and the signal bandwidth $f_{\mathrm{B,x}}$ define the upper bound of the event rate f_{e} and the lower bound $\delta_{\min}$ for the change of the signal between events.

3.1.4. Sensitivity of Change Detectors

The sensitivity of a sensing system defines the number of quantization levels the change of a signal includes. These quantization levels are often termed *ticks*. For example, an ADC with N_{ticks} quantization levels and a voltage range of $V(N) \in [0,\ V_{\max}]$ maps N ticks to a voltage $V(N)$ with

$$V(N) = \frac{V_{\max}}{N_{\mathrm{ticks}}}\, N. \tag{3.54}$$

Then, the sensitivity $\mathrm{S}_{\mathrm{adc}}(V)$ of the ADC is defined by

$$\mathrm{S}_{\mathrm{adc}}(V) = \left|\frac{\mathrm{d}N}{\mathrm{d}V}\right| = \frac{N_{\mathrm{ticks}}}{V_{\max}} \qquad [\text{ticks/V}] \tag{3.55}$$

and describes how many ticks encode one Volt. Since the ADC uses linear quantization, the sensitivity is constant and independent of V. In general, the sensitivity of a signal $x(N)$ is defined by

$$\mathrm{S}(x) = \left| \frac{\mathrm{d}N}{\mathrm{d}x} \right| \tag{3.56}$$

and vice versa, the resolution $\mathrm{R}(x)$ is defined by

$$\mathrm{R}(x) = \frac{1}{\mathrm{S}(x)}. \tag{3.57}$$

The resolution then describes how many units a tick encodes. To detect small changes of $x(t)$, the sensitivity should be maximized and the resolution be minimized.

For change detectors, the event threshold δ defines the change of the signal $x(t)$ between two events, see Equations (3.9) and (3.15). Assuming that the quantization resolution of a discrete change detector (e.g. the resolution of the employed ADC) is lower than δ, then the sensitivity of the change detector is, in general, dominated by δ.

If the input signal $x(t)$ of a change detector is in the range $x(t) \in [x_{\min}, x_{\max}]$ and we target a resolution $\mathrm{R}(x) = R_{\mathrm{x}}$, then the event threshold δ is defined by

$$\boxed{R_{\mathrm{x}} = \frac{x_{\max} - x_{\min}}{N_{\mathrm{ticks}}} = \frac{N_{\mathrm{ticks}}\,\delta}{N_{\mathrm{ticks}}} = \delta}\,. \tag{3.58}$$

Since the sensitivity directly depends on the event threshold δ, maximizing the sensitivity competes with minimizing the event rate f_{e}. Furthermore, the best possible sensitivity is bounded by $\delta_{\min}$ and thus by the system bandwidth f_{B} and the signal bandwidth $f_{\mathrm{B,x}}$. In the following section we will additionally see that the sensitivity is tightly coupled with the encoding error.

3.1.5. Conversion Error of Change Detectors

The maximum error between the signal $x(t)$ monitored by the change detector and the signal encoded by the events is set by the maximum difference that can occur between two events. The update condition of the change detector, Equations (3.9) and (3.15), describes this difference. A novel event containing the current value of the signal $x(t)$ is not generated until the difference of $x(t)$ to the value of the last event $x(t(\mathbf{e}_{i-1}))$ is not at least δ. Thus, the conversion error ϵ directly relates to the event threshold δ. The error ϵ between $x(t)$ and the signal encoded by events is at most

$$\epsilon_{\max} = \delta. \tag{3.59}$$

However, this relation only holds if $\delta \geq \delta_{\min}$. This lower bound is set by Equation (3.52) through the relationship between the maximum average absolute slope (limited by the bandwidth of the signal) and the system bandwidth f_{B}. If the signal bandwidth $f_{\mathrm{B,x}}$ of $x(t)$ is too large in comparison to the system bandwidth f_{B}, then the condition $\delta \geq \delta_{\min}$ might no longer be fulfilled. Then, for $\delta < \delta_{\min}$ the error $\epsilon_{\max}$ will be larger than δ whenever

$$\left|\overline{\dot{x}_i}\right| > \delta\, f_{\mathrm{B}} \tag{3.60}$$

that is, when the absolute average slope of $x(t)$ is larger than the product of event threshold and the system bandwidth of the event generator.

Consequently, the sensitivity and the encoding error are both directly determined by the event threshold δ. A smaller δ results in a higher sensitivity S and a smaller encoding error ϵ, but causes higher event rates f_{e}. The direct dependency of the error on δ only holds for $\delta \geq \delta_{\min}$. That is a fact that has to be considered when decreasing δ to improve the sensitivity and to reduce the encoding error. In the case that $\delta_{\min}$ severely impacts sensitivity and the encoding error, the adjustment of $f_{\mathrm{B,x}}$ can be considered or, if possible, the process or the sampling rate could be improved to increase the system bandwidth f_{B}.

3.1.6. The Influence of Noise on the Event Rate

This section analyzes the impact of noise on the event rate of change detectors. Ideally, a change detector only creates events when it detects sufficient change in the input signal $x(t)$. Unfortunately, any input signal is superimposed with noise. This noise also generates changes that could trigger the generation of events. Events generated by noise are undesirable and a good parameterization of the change detector should minimize their occurrence. Therefore, this section investigates the dependency of the change detector's parameters on the properties of noise.

Noise $z(t)$ superimposes the actual stimulus $s(t)$ such that the input signal $x(t)$ of a change detector composes of:

$$x(t) = s(t) + z(t). \tag{3.61}$$

Thus, noise impacts the event generation, and some events will be caused by noise rather than by the change of the stimulus. Assuming Additive White Gaussian Noise (AWGN), then $z(t)$ is white noise with a normal distribution, an expected value μ_{N} of zero, and a variance of $\mathrm{Var}(Z) = \sigma_{\mathrm{N}}^2$, thus

$$Z \sim \mathcal{N}(0, \sigma_{\mathrm{N}}^2) \tag{3.62}$$

The work presented in Section 3.1.6 was in part published in:
Bergner, F., Dean-Leon, E., Cheng, G., "Design and Realization of an Efficient Large-Area Event-Driven E-Skin". In: *Sensors* 20.7 (2020), p. 1965.

In this case, since $z(t)$ is normally distributed, 68% of its values z_i (evaluations of its random variable Z) would deviate from zero at most by σ_N, 95% at most by $2\,\sigma_\mathrm{N}$, and 99.7% at most $3\,\sigma_\mathrm{N}$. These deviations are changes that the event generator converts to events. Thus, the standard deviation σ_N of the additive noise impacts the selection of the event threshold δ (Figure 12). For example, an event threshold $\delta \geq 3\,\sigma_\mathrm{N}$ will reduce the impact of noise by 99.7%, since 99.7% of the changes in $x(t)$ caused by noise will not surpass the threshold δ.

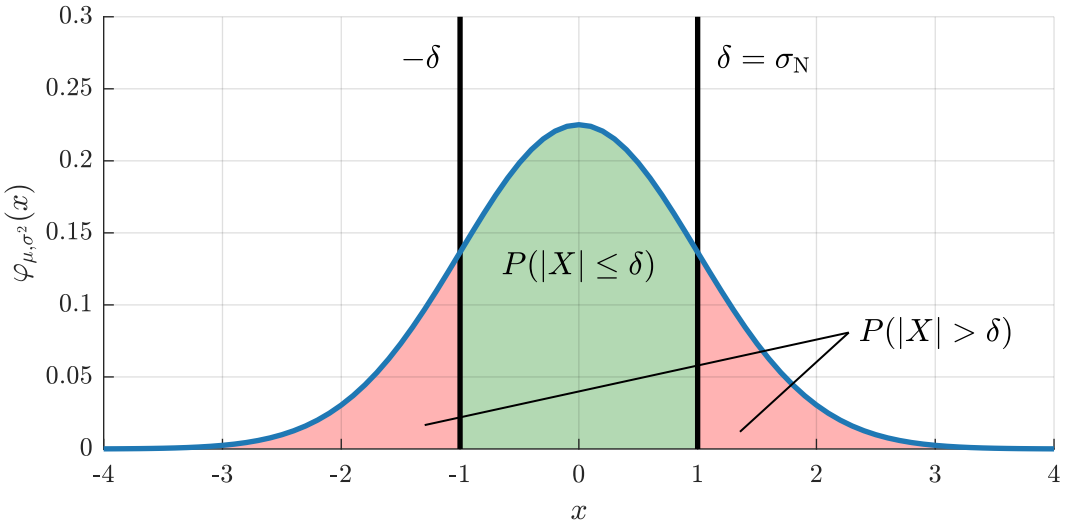


Figure 12 The probability density function $\varphi(x)$ for the normal distribution $X \sim \mathcal{N}(0,1)$ with $\sigma_\mathrm{N} = 1$. In this example, the threshold δ is set to σ_N. Then, $P(|X| \leq \delta)$ = 68 % of the noise would be canceled (green area), and $P(|X| > \delta)$ = 32 % would trigger events (red area).

The average rate of events $\overline{f_{\mathrm{e,N}}}$ solely caused by noise (the stimulus $s(t)$ is assumed to be constant) is then defined by

$$\overline{f_{\mathrm{e,N}}}(f_\mathrm{B}) = \begin{cases} \left[1 - \operatorname{erf}\left(0.5\sqrt{2}\right)\right]\, f_\mathrm{B} & \textbf{for} \quad \delta = \sigma_\mathrm{N} \\ \left[1 - \operatorname{erf}\left(\sqrt{2}\right)\right]\, f_\mathrm{B} & \textbf{for} \quad \delta = 2\,\sigma_\mathrm{N} \\ \left[1 - \operatorname{erf}\left(\frac{3}{2}\sqrt{2}\right)\right]\, f_\mathrm{B} & \textbf{for} \quad \delta = 3\,\sigma_\mathrm{N} \end{cases} \tag{3.63}$$

or approximately by

$$\overline{f_{\mathrm{e,N}}}(f_\mathrm{B}) \approx \begin{cases} 0.31731\, f_\mathrm{B} & \textbf{for} \quad \delta = \sigma_\mathrm{N} \\ 0.045500\, f_\mathrm{B} & \textbf{for} \quad \delta = 2\,\sigma_\mathrm{N} \\ 0.0026998\, f_\mathrm{B} & \textbf{for} \quad \delta = 3\,\sigma_\mathrm{N} \end{cases}, \tag{3.64}$$

where f_B is the system bandwidth of the change detector. For the time discrete change detector, f_B is the sampling rate f_s. The function $\operatorname{erf}(\cdot)$ is the standard Gauss error function defined by

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, \mathrm{d}t. \tag{3.65}$$

Consequently, the noise in the monitored signal $x(t)$ sets a lower bound for the event threshold δ since noise should ideally create only very few events. This lower bound is defined by the properties of the additive noise, that is σ_{N}. Furthermore, the event rate caused by noise scales linearly with the system bandwidth f_{B}. Thus, a faster system will generate more events caused by noise.

3.1.7. Parameterizing Event Generators

The previous sections presented different properties such as event rate, sensitivity, encoding error, and noise resilience and related them to the parameters and system properties of change detectors. All these properties are relevant and impact the performance of the event generators that employ change detectors for detecting novelty. Event generators should be parameterized and designed such that the event rate stays as low as possible (ideally the event generators do not generate any events when the monitored input signal is constant), the sensitivity is maximized, the encoding error minimized, and the impact of noise is neglectable. These goals compete with each other since parameters that are optimal for low event rates hinge on sensitivity and encoding error. Furthermore, noise enforces a trade-off between sensitivity and events solely caused by noise. The most important parameter for determining event generators is the event threshold δ. The event threshold not only determines how much novelty is required to trigger the next event, it is also tightly coupled to sensitivity, encoding errors, and noise. The system parameters signal bandwidth, system bandwidth (or respectively sample rate), indirectly impact the event threshold through imposing bounds on the maximum event rate and the minimum event threshold, see Section 3.1.3.2.

The analysis of all these factors towards determining the best compromise leads to the following guidelines for tuning the event threshold δ:

1. When the encoding error ϵ should be lower than $\epsilon_{\max}$, then choose $\delta < \epsilon_{\max}$
2. When the error ϵ has to be bounded by $\epsilon_{\max}$, then choose $\delta_{\min} \leq \delta < \epsilon_{\max}$
3. When the sensitivity requires to detect changes down to a minimal change $\Delta_{\min}$, then choose $\delta < \Delta_{\min}$
4. When the idle event rate $f_{\mathrm{e,idle}}$, that is, the event rate for $x(t) = \mathrm{const.}$, or respectively, the noise event rate $\overline{f_{\mathrm{e,N}}}$ has to be smaller than 1% of f_{B}, then choose $\delta > 3\,\sigma_{\mathrm{N}}$
5. When the overall average event rate $\overline{f_{\mathrm{e}}}$ is too high, then
   a. Increase δ at the cost of reducing the sensitivity and increasing the encoding error

---

The work presented in Section 3.1.7 was in part published in:
Bergner, F., Dean-Leon, E., Cheng, G., "Design and Realization of an Efficient Large-Area Event-Driven E-Skin". In: *Sensors* 20.7 (2020), p. 1965.

b. Decrease f_{B} at the cost of reducing the temporal precision

6. Similar to the signal-to-noise ratio (SNR), the σ_{N} of the noise source defines the performance limits of the event generator, and thus, has to be kept as small as possible.

3.2. Event Representation and Communication Protocols

After the presentation of the generation of events in the previous Section 3.1, this section focuses on the representation of events (Section 3.2.1) and protocols for their communication (Section 3.2.2).

3.2.1. Event Representation

The event generators discussed in the previous section focused on novelty detection in monitored signals or sensory stimuli. They employ change detectors that trigger activity when novelty has been detected. Activity alone can not convey or represent information. Codes have to be employed. Inspired by the neural codes found in nature (see Chapter 2), technical systems can utilize different encoding schemes to convey information through events. In the following, we first discuss the different kinds of information events need to be able to encode before we detail some of the prominent event encoding schemes applied in technical systems.

In general, events need to encode

- the time of occurrence,
- the source/location,
- the type, and
- the magnitude

of information. While the activity, or more precisely, the presence of an event already encodes the time of occurrence of information, its source, type, and magnitude have to be encoded by different means.

A technical system can represent an event by an *activity token*. In its simplest form, an activity token is just a pulse in a wire, similar to an action potential in a nerve fiber. The pulse then represents the event, and the presence and absence of the pulse encodes the presence and absence of the event. Similarly, the occurrence of the pulse in the wire encodes the occurrence time of the event.

Since technical systems can implement much higher transmission rates per wire with up to several Gbit/s in comparison to the approximately 1 kbit/s per nerve fiber in biology, technical systems may time-multiplex activity tokens to share a common wire. This reduces the number of wires without affecting the performance of the system, a very desirable feature since wires have many penalties in technical systems and are hard to realize when their number increases.

When several information sources feed activity tokens into the same wire, then these tokens have to be distinguishable, they have to encode their source of origin. This can be achieved in two ways: 1) Rather than by a one-bit token, the event is represented by a multi-bit serial datagram, or 2) The activity token is represented by a bit code in several parallel wires. The serial datagram for an activity token of option 1) requires only one wire but occupies the shared wire for N different sources for $M = \lceil \log_2(N) \rceil$ bit intervals. The parallel activity token of option 2) occupies each of its parallel wires only for the time interval of one bit, but requires for N different sources $M = \lceil \log_2(N) \rceil$ wires. Either way, if the bit rate of a wire is by a factor F higher than the temporal resolution required, then the wire could be shared by F/M sources. Each source is then identified by an M-bit identifier, that is the source ID.

Additional information, for instance, type and magnitude, can similarly be encoded into the activity tokens of the events with the penalty of demanding additional transmission capacity, either through occupying the shared wire for a longer time with longer serial datagrams (option 1) or through additional wires in the parallel bus (option 2).

Neural codes in the brain encode information by *context*, which is spatial-temporal patterns. Similar principles (with emphasis on temporal patterns since spacial information is already encoded by source IDs) can be exploited in technical systems to encode magnitude information. Therefore, two or more events of the same information source together encode magnitude information.

For instance, on and off events $\mathbf{e}_{\uparrow}$ and $\mathbf{e}_{\downarrow}$ could encode that the monitored signal increased or decreased by δ. Then, the activity token of the event only needs one additional bit to encode the type of the event. Thus, if a system uses M_{s}-bit for source IDs and M_{v}-bit for magnitudes, then on/off events have a higher encoding efficiency when

$$M_{\mathrm{v}} > 1 \tag{3.66}$$

which is always the case. To reconstruct the magnitude, the event decoder increases/decreases the last memorized magnitude by δ according to the type of the event. Thus, encoding/decoding magnitudes by on/off events have a low complexity while the gain of encoding efficiency is substantial. The downsides of the on/off encoding scheme are initialization and drift. The event decoder only receives differences. Thus, if the absolute value is not known, e.g. cannot be assumed to be zero when having been inactive for a longer time, then the magnitude can not be reset and the magnitudes reported by the event decoder will show a constant offset with respect to the real signal. The reconstructed signal of the event decoder also drifts. These drifts are caused by the loss of events or when the encoding error is not δ, for example, when the event generator gets bounded by its system bandwidth whenever the monitored signal's slope is too large or the event threshold is too small, see Section 3.1.5. The on/off event encoding scheme has been successfully applied in the event-driven sensing systems of [86, 28].

Rather than encoding the signal magnitude into bits, the magnitude can be also encoded as time. Therefore, the event generator creates two events for each detected change. The low event $\mathbf{e}_{\mathrm{L}}$ occurs on the detection of the change and thus encodes the time of occurrence and the information source. The high event $\mathbf{e}_{\mathrm{H}}$ occurs after the low event such that the time

$$T_{\mathrm{LH},i} = t(\mathbf{e}_{\mathrm{H},i}) - t(\mathbf{e}_{\mathrm{L},i}) \tag{3.67}$$

between low and high events correlates to the magnitude $x(t_i)$ at $t_i = t(\mathbf{e}_{\mathrm{L},i})$

$$x(t_i) \propto T_{\mathrm{LH},i}. \tag{3.68}$$

The encoding efficiency of low/high events is better than encoding magnitudes to single activity tokens when

$$2\ (M_{\mathrm{s}} + 1) < M_{\mathrm{s}} + M_{\mathrm{v}} \tag{3.69}$$

which simplifies to

$$M_{\mathrm{s}} + 2 < M_{\mathrm{v}}. \tag{3.70}$$

The encoding efficiency of low/high events always surpasses events encoding the magnitude in their activity tokens when the event generator is time continuous and range continuous. In this case, given that the events are conveyed with a high enough temporal precision, low/high events encode magnitudes with approximately continuous precision. A comparable precision can not be reached by magnitudes encoded with M_{v}-bit. The downsides of the low/high event encoding scheme are the complex decoding and the high demands on temporal precision. The low/high event encoding scheme has been successfully applied in the event-driven vision sensing systems of [122].

3.2.2. Communication Protocols for Events

The previous section depicted the encoding of information in events and their representation in activity tokens. This section focuses now on communication protocols to convey these activity tokens, that is, the events. Communication protocols for events have to comply with two key aspects to ensure the efficient communication of events and the event-driven mechanism in event-driven systems. The protocol has to ensure

1. the efficient, fast, and non-complex arbitration of events onto shared communication buses, and
2. the asynchronous conveyance of information.

As discussed in the previous Section 3.2.1, the technical system requires an arbitration to fully take advantage of faster communication capabilities. Both the fast arbitration and the asynchronous communication ensure that the communication protocol preserves the occurrence

time of the events. With that, the communication protocol preserves the time information encoded in the events, and it ensures that activity drives the communication and can trigger the handling of information in the following stages of the event-driven information handling system.

Communication protocols for events can achieve the required characteristics in two different ways. First, the communication protocol can be fully realized in specifically customized asynchronous hardware with a hardware arbiter and an asynchronous communication bus for events. Then, the activity is arbitrated through a handshaking protocol. Second, the communication protocol can be realized with standardized communication buses and asynchronous software protocols. The former approach appeals with superior performance but often hinges on flexibility and complex arbitration. The latter approach wins over with flexibility, standardized, and hardware-independent arbitration at the cost of reduced performance and less timing precision.

The following two sections present the two most prominent communication protocols for events and detail their properties regarding arbitration and asynchronous information conveyance. Both protocols have been introduced in Chapter 2 and the focus lies now on the aspects relevant for this section.

3.2.2.1 Address-Event-Representation (AER) Protocol

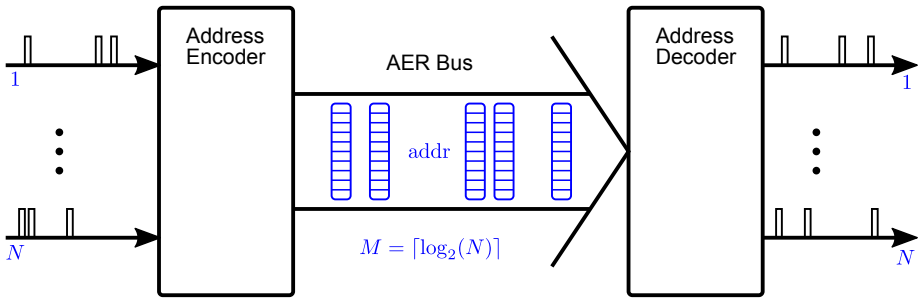
The AER (see Figure 13) represents events by activity tokens on a parallel and more recently also a serial asynchronous hardware bus [90, 22, 126].

Figure 13a provides a conceptual overview of AER. N event sources $1, \ldots, N$, e.g. sensory event generators, produce events. Each of these sources has an exclusive connection to the address encoder. The address encoder arbitrates the activity of these N connections employing a fair arbitration scheme (e.g. round-robin) and time-multiplexing such that the events of N sources can be represented by events on a common asynchronous, parallel AER bus. The events of N sources can be represented by M-bit addresses. Thus, the AER can reduce the number of wires by a factor N/M. The address decoder de-multiplexes AER events back to events in connections $1, \ldots, N$. After that, just as before, the connection itself encodes the source of the events.

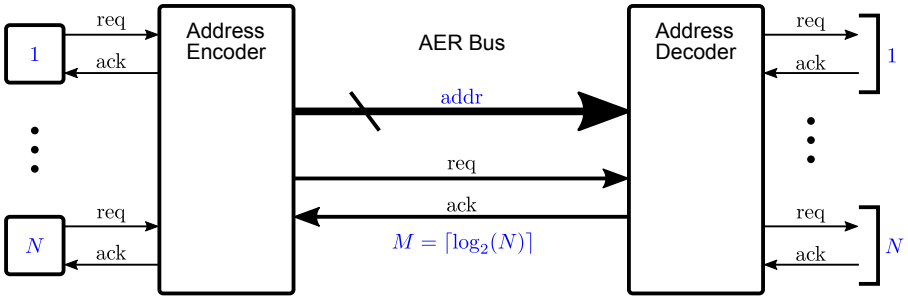
Figure 13b concretizes the arbitration and its self-timed handshaking scheme. The whole protocol is driven by events. The event source decides when an event is sent. Thus, the information is not synchronized to any clock or requested by the receiver. The conveyance of an event follows a four-phase handshaking cycle as depicted in Figure 13c. First, a source requests the arbitration of an event through a request line (REQ). Then, the arbiter (address encoder/decoder) acknowledges the request through the acknowledge line (ACK). Finally, the event source releases the request after which the arbiter releases the acknowledgement,

and the cycle finishes. One of these cycles represents the timeline/pulse, or respectively, the activity token of an event, while the address bus encodes the source of the event.

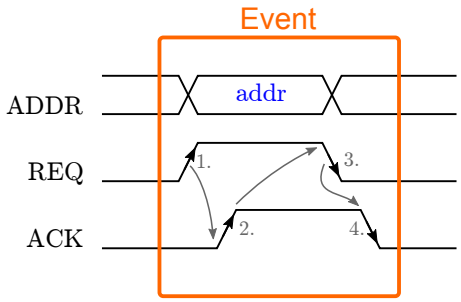
Figure 13d depicts the protocol for serial AER replacing the parallel bus between address encoder and decoder [126]. In serial AER the event is serialized into an event datagram using only one pair of differential wires (LVDS). The datagram uses a start bit S, M address bits, and a parity bit P. In contrast to parallel AER, the datagram itself represents the activity token through its format. The start and end of the token are defined by the start and end of the datagram.



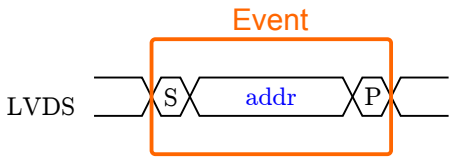
(a) Conceptional overview. The address encoder maps N inputs to addresses of length M. The address decoder maps the addresses back to N outputs.



(b) Self-Timed transmission. The source requests to send and the receiver acknowledges the request.



(c) Parallel AER. The address is transmitted on a parallel bus. The request and acknowledge signals define the validity of the activity token (event).



(d) Serial AER. The address is serialized into a datagram with a start bit and a parity bit.

Figure 13 The Address-Event-Representation Protocol. All figures are adopted from [90, 22, 126].

The AER protocol realizes fast and efficient arbitration of events to a common communication bus. The communication bandwidth is only limited by the bandwidth of the bus and

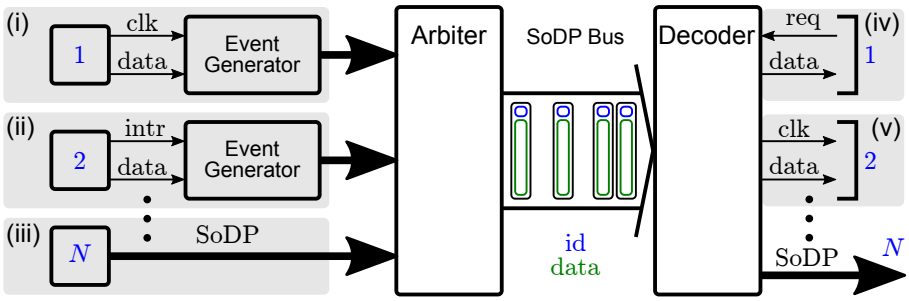
the handshaking cycle time, or respectively, the datagram length. Events are represented and communicated asynchronously. Thus, the protocol provides a high temporal resolution. However, the arbitration is complex and not flexible. The protocol requires additional communication lines, and event sources and sinks cannot be easily added or removed. Furthermore, the AER protocol relies on customized hardware which additionally hinges on flexibility.

3.2.2.2 Send-on-Delta Principle (SoDP) Protocol

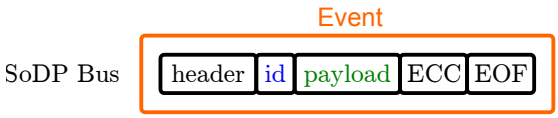
The SoDP is a protocol for communicating events that are completely hardware independent [114, 97, 99]. The SoDP represents events by packets on an asynchronous communication interface (see Figure 14). The arbitration is handled by the protocol of the communication interface.

Figure 14a provides an overview of the various ways to establish event-driven communication channels between information sources and sinks through links employing the SoDP protocol. If multiple sources share a common communication interface, then an arbiter implements a fair sharing of the bus between the information sources. Packets representing events and containing their information, thus termed event packets, are queued, scheduled (e.g. round-robin), and finally placed on the common bus. Similar to the events in AER, the event packets are self-timed and are produced by the event sources (e.g. event generators) and thus drive the communication. The occurrence of an event packet encodes the occurrence time of the event it represents. The labels (i), (ii), (iii) in Figure 14a depict different event sources $1, \ldots, N$. An event source could be a clock-driven information source connected to an event generator that produces event packets (i), an information-driven sensor with an interrupt line connected to an event generator (ii), or event-driven sensors or algorithms that produce event packets (iii). The SoDP decoder de-multiplexes the shared bus to single-ended SoDP channels or to sinks that are clock-driven (v) or request information at their own rate (iv).

The activity token of an event packet is represented by its datagram, see Figure 14b. The start/end of the datagram represents the borders of the event's activity token. The event packet at least encodes the ID of the information source and usually also the magnitude and the type of information into its payload. The frame of the event packet datagram is usually defined by a header to mark the start of the datagram, the payload containing the information to convey, the Error Correction Code (ECC) to increase the robustness of the payload against errors, and the End of Frame (EOF) to mark its end.



(a) Conceptional overview. The arbiter maps N inputs to a common bus. The decoder maps the datagrams back to N outputs.



(b) Representation of an SoDP event as a datagram containing a header, a source id, a payload, an error correction code, and an end-of-packet token.

Figure 14 The Send-on-Delta Principle: (a) Overview, (b) SoDP Event Datagram.

The SoDP protocol realizes fast and non-complex arbitration exploiting standard communication protocols for asynchronous interfaces. Its hardware independency allows for its great flexibility. Events are represented and communicated asynchronously enabling the SoDP protocol to realize event-driven communication, even between non-event-driven sources and sinks. This flexibility has to be paid by a reduced encoding efficiency. The SoDP protocol relies on existing non-specialized communication interfaces and protocols (e.g. User Datagram Protocol (UDP)). Thus, the temporal resolution of the SoDP is lower than in the case of AER and is usually not sufficient to encode information in the context between events, for instance, in the timing between events. Magnitude, type, and source ID information are stored in the payload of the event packets, increasing the number of bits per event.

3.3. Event-Driven Information Handling

The previous two sections, Sections 3.1 and 3.2, focused on generating event-driven information, and on representing and conveying event-driven information. However, complex systems, artificial or biological, additionally process information and generate actions to achieve desired system behaviors. The complete information flow in a perception-action loop [52] or a system control loop, that is, acquiring, transmitting, processing, and acting on information [72], describes the process of *handling information*.

This section now focuses on the missing part of processing/acting and introduces the theory and dominant principles for creating systems in which novel information represented by events drives computations and reactions. These principles contrast to clock-driven computations and reactions where updates are dictated by a system clock. Section 3.3.1 summarizes these principles while Section 3.3.2 derives and evaluates a model that estimates the resulting efficiency gain of event-driven computations and reactions.

3.3.1. Event Handling – Activity on Demand

To achieve a complete event-driven system, that is an event-driven information handling system, processing and reactions need to be driven by events. From the information theory point of view, algorithms and computations realizing reactional behaviors actually should only have to compute updates and trigger actuation when novel information is available. This novel information could, for instance, be a changing system state (internal information) or sensory feedback (external information). In this context, a system, that is neither changing its internal state (e.g. damping a motion until it gets stationary), nor receiving novel sensory inputs, is stationary/idle. A stationary system does not need to process information and is thus inactive.

A system in a clock-driven setup can be stationary but still, compute updates with an information rate of zero. The clock dictates the computation of updates whether they are strictly necessary or not. That is, a stationary clock-driven system demands network bandwidth and processing power without computing a new result/reaction.

Contrarily, an event-driven system is only active when it is not stationary. The updates of its underlying algorithms are triggered by events that may originate externally (sensory feedback) or internally (internal state changes). Thus, event-driven processing and actions require implementations that can realize *activity on demand*. The computations and actions should rest/sleep until novel information requires the computation of an update.

From these insights, we expect that event-driven information processing consumes less power and reduces computation demands, because the event-driven system is, on average, less active and executes fewer computations. Similar to the fact that the event rate linearly relates to the transmission bandwidth occupied in communications, we expect that the event rate also

linearly relates to demands on computational power, that is

$$b_{\mathrm{c}} \propto f_{\mathrm{p}} = f_{\mathrm{e}} \tag{3.71}$$

where b_{c} is the transmission bandwidth in bit/s, f_{p} the event packet rate in packet/s, and f_{e} the event rate in event/s, and

$$f_{\mathrm{instr}} \propto f_{\mathrm{p}} = f_{\mathrm{e}} \tag{3.72}$$

where f_{instr} is the computational load in Instructions Per Second (IPS).

Chapter 2 introduced related work that addresses the implementation of the discussed event-driven communication and processing principles. Chapter 4 will introduce the implementation of event-driven information handling in Large-Area Skin Systems (LASSs), and particularly how event-driven systems can be realized on standard computing systems (Section 4.2). Both, Chapters 4 and 5 will validate the effectiveness of event-driven information handling.

3.3.2. Event Handling – Processing

Following the discussion of Section 3.3.1, it seems plausible that the processing demands in terms of CPU usage or computation time u increase linearly with the number of packets (events)

$$u \propto f_{\mathrm{p}}. \tag{3.73}$$

Event-Driven systems would already profit from such a directly proportional relationship since their reduction of the transmission rate would directly and proportionally impact their processing demands in information handling. That is, a reduction of the transmission rate would result in an equal reduction of computational load.

However, the experimentation in Chapter 4 indicates that the relation between CPU usage and packet rate (directly correlates to transmission rate) is better than linear, that it is approximately logarithmic. This finding would imply that the investigated processing algorithm is more efficient when processing more information per instance of time.

To better understand this behavior, this thesis introduces a realistic CPU usage model that accounts for the queuing of information, the inter-scheduling delay of threads, the setup time for resuming threads, and the processing time per packet. If this model fits well with the measurements of the real system, then the increase of efficiency for larger packet rates is accounted for by the virtual reduction of the setup time per packet through queuing.

3.3.2.1 CPU Usage Model

The model assumes that a thread processing a packet causes two types of CPU usage. First, a computation time t_s is spent to resume the thread. Then, a computation time t_p is spent to process the packet. While the setup time t_s is spent once, when the thread reenters the running state, the packet processing time t_p is spent for each processed packet. Thus, when each packet is processed immediately before the next packet arrives, then the CPU usage per packet is defined by

$$\frac{\mathrm{d}u}{\mathrm{d}p} = t_\mathrm{s} + t_\mathrm{p} \tag{3.74}$$

where p denotes the number of packets. During this condition, the model employs a queue with a length of one.

The model furthermore assumes that the scheduler does not immediately resume a thread that has just been suspended (fair scheduling, e.g. round-robin). This assumption is valid since the scheduler has to fairly share computation time between threads. This turnaround/-idle time t_I, until a thread is rescheduled, in combination with t_s and t_p, defines the packet rate $f_\mathrm{p,b}$ beyond which the queue length q is, in average, larger than one:

$$f_\mathrm{p,b} = \frac{1}{t_\mathrm{s} + t_\mathrm{p} + t_\mathrm{I}} \tag{3.75}$$

with

$$\begin{aligned} q &\leq 1 \quad \text{for} \quad f_\mathrm{p} \leq f_\mathrm{p,b} \quad \text{and} \\ q &> 1 \quad \text{for} \quad f_\mathrm{p} > f_\mathrm{p,b}. \end{aligned} \tag{3.76}$$

As long as the thread processes more packets per second than packets arrive, the queue length stabilizes such that the packet rate f_p equals the packet processing rate. This condition leads to the following relationship

$$t_\mathrm{s} + t_\mathrm{p}\, q + t_\mathrm{I} = q\, \frac{1}{f_\mathrm{p}}. \tag{3.77}$$

That is, the time that passed between resuming the process twice equals the time it took to accumulate q packets in the queue when packets arrive with a rate of f_p. This relationship leads to a formula for the average queue length with a dependency on the packet rate

$$\boxed{q(f_\mathrm{p}) = \frac{t_\mathrm{s} + t_\mathrm{I}}{1/f_\mathrm{p} + t_\mathrm{p}}}\,. \tag{3.78}$$

Furthermore, the CPU usage per packet for queue lengths greater than one can be computed with

$$\frac{\mathrm{d}u}{\mathrm{d}p} = t_\mathrm{s} + t_\mathrm{p}\, q = q\,\frac{1}{f_\mathrm{p}} - t_\mathrm{I} \tag{3.79}$$

which with Equation (3.78) leads to

$$\frac{\mathrm{d}u}{\mathrm{d}p} = \frac{1}{f_\mathrm{p}}\left[1 - \frac{t_\mathrm{I}}{t_\mathrm{s}+t_\mathrm{I}}\left(1 - t_\mathrm{p}\right)\right]. \tag{3.80}$$

Thus, with the results of Equations (3.74) and (3.80) we can calculate the CPU usage per second with the dependency on the packet rate

$$\boxed{u(f_\mathrm{p}) = \begin{cases} \left(t_\mathrm{s} + t_\mathrm{p}\right) f_\mathrm{p} & \textbf{if} \quad f_\mathrm{p} \leq f_\mathrm{p,b} \\ 1 - \frac{t_\mathrm{I}}{t_\mathrm{s}+t_\mathrm{I}}\left(1 - t_\mathrm{p} f_\mathrm{p}\right) & \textbf{if} \quad f_\mathrm{p} > f_\mathrm{p,b} \end{cases}}. \tag{3.81}$$

The packet rate $f_\mathrm{p,s}$, where the system will saturate with $u(f_\mathrm{p,s}) = 1$, is defined by

$$f_\mathrm{p,s} = \frac{1}{t_\mathrm{p}} \tag{3.82}$$

and is thus only dependent on the time needed to process one packet.

Figures 15, 16, and 17 depict the different effects of the parameters of the CPU usage model

$$\mathbf{p}_\mathrm{u} = \begin{pmatrix} t_\mathrm{s} & t_\mathrm{p} & t_\mathrm{I} \end{pmatrix}^\top \tag{3.83}$$

where we assume

$$\mathbf{p}_\mathrm{u} = \begin{pmatrix} 16 & 16 & 150 \end{pmatrix}^\top \text{µs}. \tag{3.84}$$

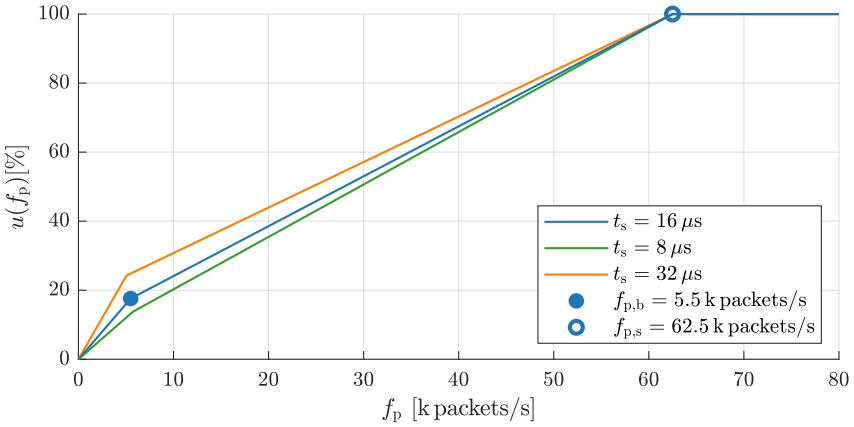


Figure 15 Decreasing/Increasing t_s only slightly impacts $f_\mathrm{p,b}$ but significantly influences the slope of the CPU usage for $f_\mathrm{p} \leq f_\mathrm{p,b}$. A larger t_s induces a larger slope. The model has been parameterized with $\mathbf{p}_\mathrm{u}$ as defined in (3.84).

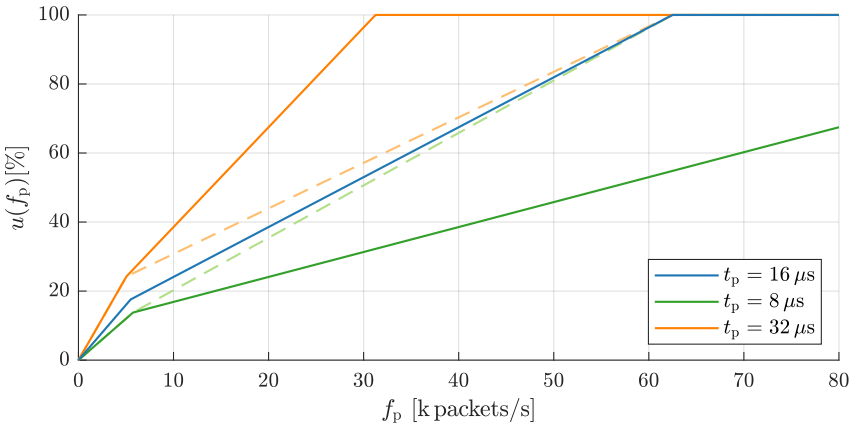


Figure 16 Similar to t_s, see Figure 15, decreasing/increasing t_p only slightly impacts $f_\mathrm{p,b}$ but significantly influences the slope of the CPU usage. Additionally, t_p severely impacts $f_\mathrm{p,s}$. A larger t_p decreases $f_\mathrm{p,s}$. The model has been parameterized with $\mathbf{p}_\mathrm{u}$ as defined in (3.84).

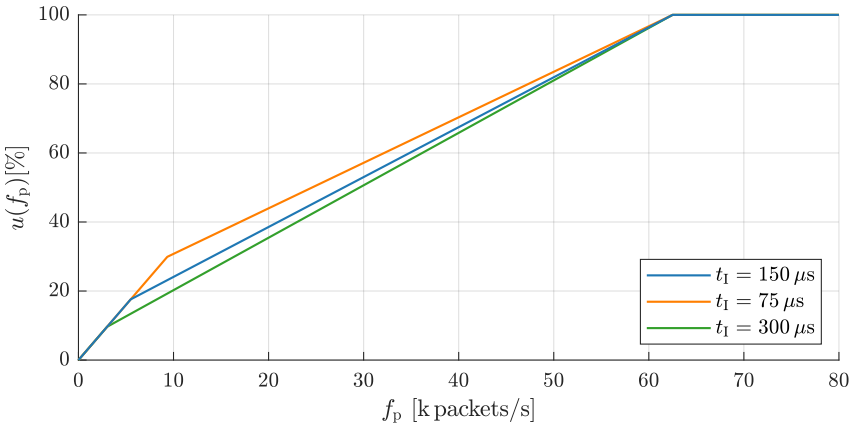


Figure 17 Decreasing/Increasing t_I mainly impacts $f_\mathrm{p,b}$. A larger t_I decreases $f_\mathrm{p,b}$ that is, packets are queued earlier. The model has been parameterized with $\mathbf{p}_\mathrm{u}$ as defined in (3.84).

Figure 18 visualizes the queue length of the model. Here, we set the limit of the queue $q_{\max}$ to 500 packets. According to the model, q would reach infinity at $f_{\mathrm{p}} = f_{\mathrm{p,s}}$. The presented model is only valid until the maximum queue length $q_{\max}$ is reached. After that, the relation assumed in Equation (3.77) is no longer valid. In reality, the accuracy of the model will deteriorate even before $q_{\max}$ is reached since other effects that are not modeled will gain influence and dominate the behavior. For example, the scheduler could increase the priority of the process because it has a high demand for computational power, or the CPU could boost the execution for short intervals. The scheduler could even execute the process without performing further context switches. These effects would delay the saturation of the queue and push the measured CPU usage below the expected values predicted by the model.

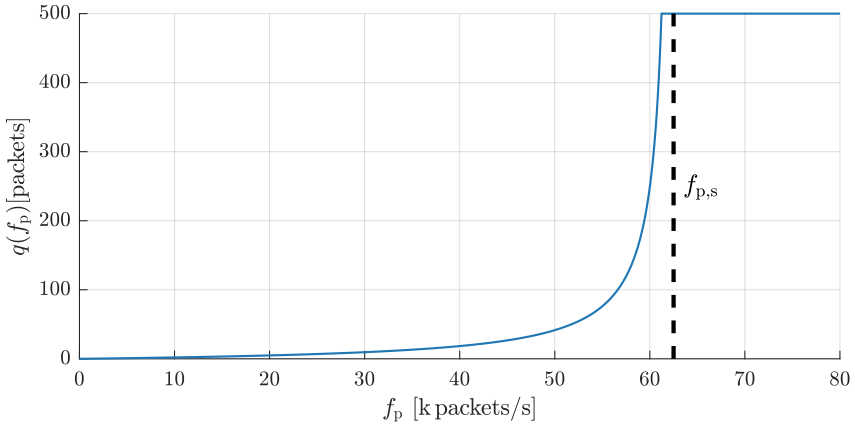


Figure 18 Queue length q of the model in number of packets. The queue length is limited to the maximum queue length $q_{\max}$. The model has been parameterized with $\mathbf{p}_{\mathrm{u}}$ as defined in (3.84).

3.3.2.2 CPU Usage Model – Validation

The presented CPU usage model can be validated with the results that have been obtained in experiments with a real event-driven Large-Area Skin System (LASS), see Chapter 4. The measured CPU usage for various packet rates can be drawn on to fit the CPU usage model, see Figures 19 and 20. The parameters $\mathbf{p}_{\mathrm{u}}$ of the models have been obtained by employing non-linear least square fitting. For the clock-driven system the results are

$$\mathbf{p}_{\mathrm{u,d}} = \begin{pmatrix} 15.6 & 14.1 & 156 \end{pmatrix}^{\top} \text{µs} \tag{3.85}$$

and for the event-driven system

$$\mathbf{p}_{\mathrm{u,e}} = \begin{pmatrix} 11.2 & 12.4 & 142 \end{pmatrix}^{\top} \text{µs}. \tag{3.86}$$

The parameters lie within reasonable ranges that could be expected in normal computer systems and the differences between the clock-driven and the event-driven operation modes are small. The model indicates that the event-driven system has a slightly better performance

than its clock-driven counterpart. The results reliably indicate that the model describes the phenomenons of the real system sufficiently well.

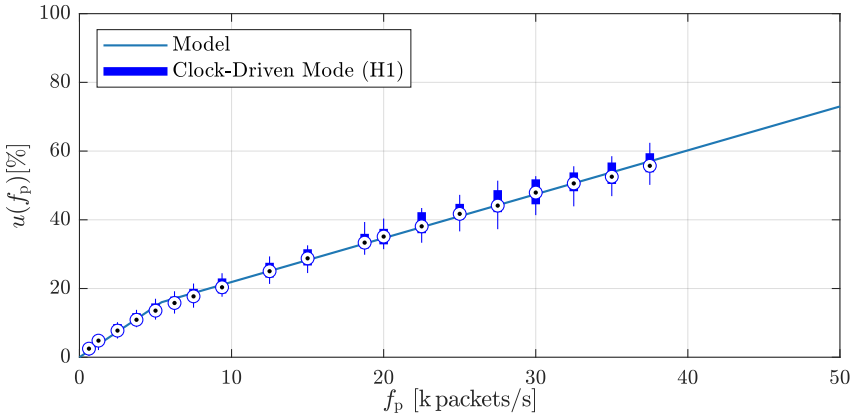


Figure 19 The model fits the statistical results for the skin driver program running in clock-driven mode on the robot H1. More information on the experiment can be found in Section 4.6.

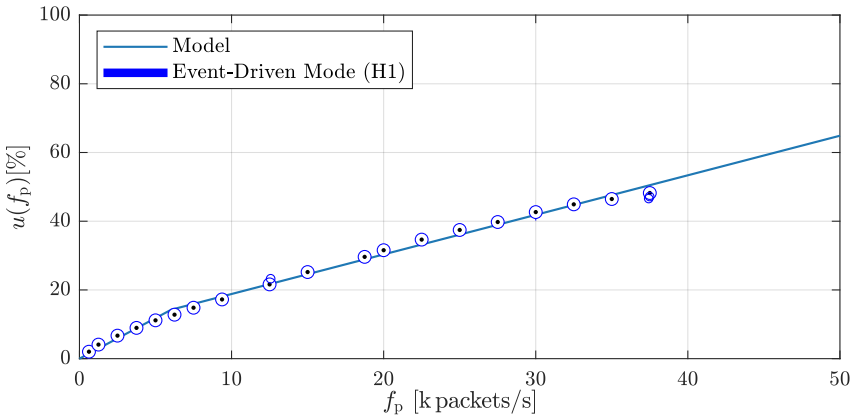


Figure 20 The model fits the statistical results for the skin driver program running in event-driven mode on the robot H1. More information on the experiment can be found in Section 4.6.

3.3.2.3 CPU Usage Model – Limitations

Naturally, the proposed CPU usage model can only account for the dominant effects within specific operational conditions of a very complex and dynamic system with many stochastic disturbances. The model describes the system behavior well up to a CPU usage level of around 80%. Above this level, effects that are not modeled, such as queue saturation, packet loss, jitter, temperature, scheduling accuracy, and the competition between threads, will gain influence and deteriorate the accuracy of the model.

In summary, the presented CPU usage model describes the relationship between CPU usage and packet rate with good accuracy in real systems for a wide range of packet rates. The model fits well to support the evaluations of real experimental measurements (in Chapter 4)

and provides insights to explain the counter-intuitive effect that processing packets and thus events gets more efficient for larger packet/event rates.

3.4. Summary

This chapter outlined the working principles of change detectors that are used in event-generators to detect novelty and generate events. The theoretical analysis of these principles led to insights into how system parameters, such as the event threshold, and the signal and system bandwidth, influence and limit the performance of event generators. This resulted in guidelines for the effective parameterization of change detectors in event generators.

Event generators consist of a change detector, an event encoder, and an event transmitter. The theory of event representation and communication, which centered around activity tokens and context, and communication protocols for activity tokens, provide the background for designing effective event encoders and transmitters, completing the fundamentals for event generators.

The presented principle of activity on demand, which is the core of event-driven information handling, explained the potential efficiency gain in event-driven systems. Network bandwidth and CPU usage emerged as promising indicators for their evaluation. Additionally, the introduction of a CPU usage model for event-driven systems in standard computer systems indicated the expected behavior between the number of events per second and the CPU usage demanded in event-driven information processing.

This chapter set the stage for designing an efficient novelty-driven approach for LASSs that is the objective of this thesis, and for the subsequent comprehensive evaluation of realized LASSs that empirically proves the efficiency and effectiveness of this thesis' approach.

4. Realizing an Event-Driven Large-Area Skin System

This chapter demonstrates that the initial idea of tailoring the novelty-driven approach to Large-Area Skin Systems (LASSs) is feasible and leads to the realization of efficient LASSs, which is the central theme of this thesis.

This chapter first draws on Chapters 2 and 3 to tailor the novelty-driven approach to a design that is suitable for LASSs (Section 4.1 and 4.2). Chapter 2 delivers the basis for the design in terms of design constraints for LASSs and a suitable implementation of an Event-Driven System (EDS). Chapter 3 provides the theoretical background for efficient designs.

Then, this chapter validates the feasibility and correctness of the determined design by realizing it in a real LASS (Section 4.3). The realized e-skin incorporates a very large number of multi-modal tactile sensors (more than 10 000 sensors).

In the last step, this chapter empirically evaluates and assesses the design and realization of the LASS (Sections 4.4 to 4.7). The results demonstrate the efficiency of this thesis' novelty-driven approach for LASSs with a significant boost in performance. Event-driven LASSs are feasible, efficient, and, for the first time, enable the effective handling of tactile information in a large-area e-skin with more than 10 000 sensors.

The results of this chapter contribute a significant part to the objective of realizing efficient large-area e-skin with the capability to provide the tactile feedback of large-area contacts in physical interaction. The validation of the realized LASS's effectiveness for large-area feedback in large-scale applications follows in Chapter 5.

4.1. Event-Driven Sensing for Efficient Large Area Skin

This section presents the designs to realize event-driven sensing (as outlined in Section 3.1) for e-skin systems. Therefore, this section first introduces the approach and the expected efficiency gain (Section 4.1.1). Then, it presents further details for designing event generation in the exchangeable modules (elements with multi-modal sensors and local processing capabilities) of modular e-skin systems (Section 4.1.2).

4.1.1. Event-Driven Sensing in Large-Area Skin

This section focuses on the actual realization of LASSs considering the collected insights of Chapters 2 and 3. As pointed out in Chapter 2, *modularity*, *self-organization*, and *bio-inspired event-driven information handling* may significantly contribute to tackling the challenges of LASSs, and Chapter 3 has delivered the theory behind the event-driven approach. This chapter incorporates both results in the designs it devises.

4.1.1.1 Realizing Event-Driven Sensing in Large-Area Skin

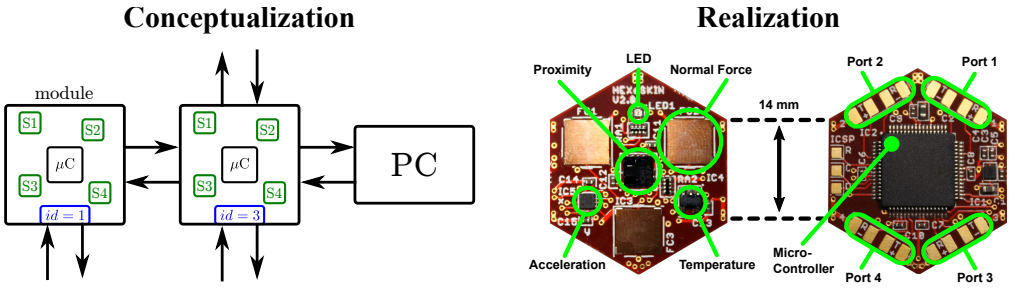


Figure 21 Conceptualization: Each sensor module/skin cell employs different sensors (e.g., S1, . . . , S4) and a microcontroller (μC) for local processing capabilities. The modules are inter-connected, form a network of modules, and provide a bidirectional communication path between each module and the information handling system, for example, a computer (PC). Realization: One skin cell of an e-skin system [109], more details in Section 4.3.

Self-organizing and modular e-skin systems tackle most of the challenges of LASSs. Therefore, a LASS should group sensors into smart modules. Within this thesis, a module is referred to as a *smart module* when the module provides local processing capabilities and when a group of modules can distributedly organize themselves into a robust network of communicating modules, see Section 2.2.3 and Figure 21. The local processing capabilities (e.g., microcontrollers) of these smart modules implement the self-organizing network capabilities in the network of modules. The realization of such a modular and self-organizing skin system then only lacks the information handling efficiency to effectively scale up to LASSs.

The work presented in Section 4.1 was in part published in:
Bergner, F., Dean-Leon, E., Cheng, G., “Design and Realization of an Efficient Large-Area Event-Driven E-Skin”. In: *Sensors* 20.7 (2020), p. 1965.
Copyright permissions: see Appendix D.

Introducing event-driven sensing and information handling to a modular and self-organizing e-skin system can extensively improve its efficiency. Therefore, Section 2.4 determined the Send-on-Delta Principle (SoDP) as the event representation and communication protocol that suits best for this purpose. The realization of this event-driven e-skin requires event-driven sensing at the level of the modules, and event-driven communication between modules and higher-level information handling. By exploiting their local processing capabilities, the modules can realize both requirements and implement event generators following the principles of the event-driven approach (as presented in Section 3.1).

Such a modular event generator, that is, a event generator in the module of an e-skin, produces SoDP events that are packets in the communication network of the modules. The self-organizing network of the modular e-skin can consequently convey these event packets similar to the packets of a Clock-Driven System (CDS), as long as the network allows asynchronous communication. Most standard communication systems, for instance, RS232/-Universal Asynchronous Receiver Transmitter (UART) or Ethernet/User Datagram Protocol (UDP), fulfill this requirement. Section 4.2 will further elaborate on the realization of efficient event-driven information handling in the higher layers.

4.1.1.2 Event-Driven Sensing increases the Efficiency of Large-Area Skin

Event-driven sensing will significantly reduce the networking and processing load in e-skin in comparison to clock-driven e-skin.

A clock-driven e-skin continuously induces high information handling loads. These loads L_d scale linearly with the sampling frequency f_s and the number of sensors n_s:

$$L_\mathrm{d} \propto f_\mathrm{s}\, n_\mathrm{s}. \tag{4.1}$$

Contrarily, event-driven e-skins only require transmission bandwidth and processing power when they are active, that is, when they handle events. These loads depend on the event rate f_e

$$L_\mathrm{e} \propto f_\mathrm{e} \tag{4.2}$$

and can be approximated by the number of activated sensors n_a that register novel information

$$L_\mathrm{e} \propto\!\!\!\sim\, n_\mathrm{a} \tag{4.3}$$

or more concretely by the shape of the stimuli $x_a(t)$ these sensors register:

$$L_\mathrm{e} \propto \sum_{a}^{n_\mathrm{a}} \overline{|\dot{x}_a(t)|}. \tag{4.4}$$

An ideal event-driven e-skin system would not create and handle events when its sensors do not register stimuli with novel information. Therefore, its event rate f_e would be zero. In the worst case, all sensors are activated and stimulated with maximum information rate. The event rate f_e then reaches the cumulative sample rate of a CDS

$$f_\mathrm{e} = f_\mathrm{s}\, n_\mathrm{s}, \tag{4.5}$$

when utilizing discrete event generators in compound architectures, for example, SoDP. The event rate could even surpass that limit in systems employing neuromorphic event-driven sensors, since these sensors usually achieve higher bandwidths than clock-driven sensors [86, 120]. However, for e-skin systems, especially LASSs, the worst case is very improbable. Touch stimuli are usually localized at spots on the body, and it is hard to stimulate the whole body with a non-constant profile for longer periods. Even covering the LASS of a robot with a cloth and moving this cloth to create additional stimuli can by far not reach the excitation levels necessary to deteriorate the performance of the EDS to that of a CDS [20] (Section 4.6). Consequently, the event rate of an event-driven e-skin

$$f_\mathrm{e} \ll f_\mathrm{s}\, n_\mathrm{s} \tag{4.6}$$

is on average much lower than the cumulative sample rate of a clock-driven e-skin.

In summary, realizing event-driven information handling in LASSs will definitely improve their information handling efficiency and will thus contribute to render these systems feasible to operate in real-world applications. This postulated effectiveness will be further investigated and validated in Sections 4.4, 4.5, and 4.6, and in Chapter 5.

4.1.2. Modular Event Generation for Large-Area Skin

Modular event generation requires the design of event generators in the smart modules of a LASS (Section 4.1.1). Therefore, a modular event generator realizes a time-discrete change detector for each of the module's sensors, see Figure 22, to convert clock-driven sensor samples to events.

The event generator of a module (Figure 22) samples a sensor and compares the current sensor value $x(t_k)$ with the sensor value $x(t_{\mathrm{K}_{i-1}})$ of the previous event $\mathbf{e}_{i-1}$. Whenever the absolute difference between the current sensor value and the value of the last event is larger than δ, then the change detector triggers the generation of a new event $\mathbf{e}_i$ (Equation (3.14)).

The event generator represents events according to the SoDP. The event packet generator takes these events and creates and transmits an *event packet* for each event. Event packets contain an ID to identify the sensor and the sensor value $x(t_{\mathrm{K}_i})$ that triggered the creation of the event/event packet. Modules that employ more than one sensor need to implement an event generator for each sensor l but can share the packet generator.

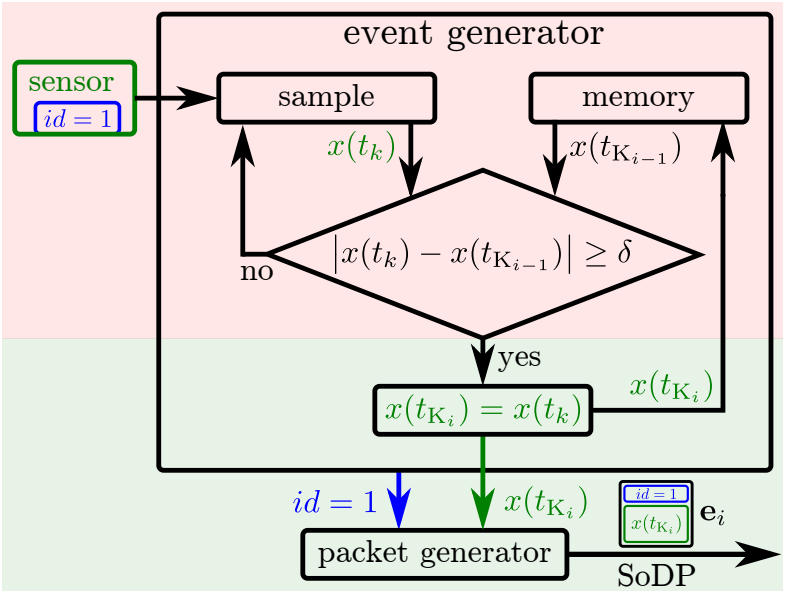


Figure 22 A networked module/skin cell with local processing capabilities (see Figure 21), can realize an event generator and an event packet generator representing events according to the SoDP. A sensor with an ID is sampled and its value $x(t_k)$ is compared with the value $x(t_{K_{i-1}})$ of the previous event $\mathbf{e}_{i-1}$. When the absolute difference is below δ, then the next sample is acquired, otherwise, a new event $\mathbf{e}_i$ is generated with a sensor value $x(t_{K_i})$. The event packet generator creates an event packet containing the ID of the sensor and the sensor value $x(t_{K_i})$. The red background indicates components that are strictly clock-driven and the green background components that are event-driven.

The spatial distribution of the sensors in a module is close and the temporal resolution of the clock-driven sensors is limited by their sampling rate. Thus, the probability that several sensors (of different modalities) in a module are excited at the same time is high. This probability leads to an optimal number of sensors per packet generator that eventually reduces the overhead of event packets. This optimal number will be further analyzed in the experimental validation of Section 4.4.6.

Events occurring at the same time can share one common event packet, see Figure 23. Sharing the event packet reduces the overhead of the packet frame that is then only required once instead of for each event. An event packet containing several events of a module encodes the information as depicted on the right side of Figure 23. That is, the payload of the event packet is $\mathbf{e}_i = (ID,\ mask,\ x_1,\ \ldots,\ x_l)$.

Figure 23 depicts event and packet generators within one module, realizing a modular event generator with sensors sharing event packets. Event packets of a module identify the module by a module ID, and sensors by a mask. The sensor values $x_l(t_{K_{i_l}})$ follow the mask field in the packet.

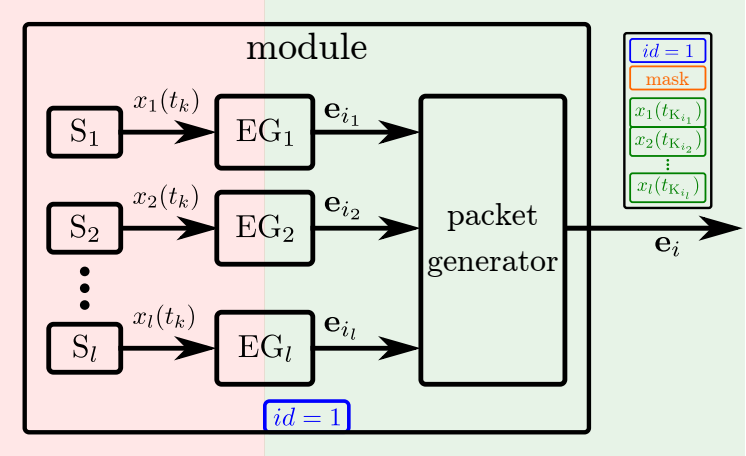


Figure 23 Modules/skin cells employing several sensors need one event generator EG_l per sensor S_l. To reduce the overhead of sending an event packet for each sensor of a module, the module can stack events $\mathbf{e}_{i_l}$ occurring at the same time $t_{K_{i_1}} = \cdots = t_{K_{i_l}}$ into one event packet $\mathbf{e}_i$. This event packet contains then the module ID to identify the module, a mask to identify the sensor, and the sensor values $x_l(t_{K_{i_l}})$ of the events $\mathbf{e}_{i_l}$.

The event packets of a modular event generator change in size according to the number of events occurring at the same time. While network protocols usually support packets with dynamic sizes, some may require a predefined fixed packet size. In the case of a fixed packet size, a good trade-off has to determine the optimal packet size that minimizes overhead. This trade-off is a compromise between the overhead of sending several small packets when not all events fit into one packet and the overhead of sending partially empty packets when only one event occurs. Section 4.4.6 will further analyze this trade-off.

In the following, the implementation presented in Section 4.3.3 will validate the presented design for modular event generation, and Sections 4.4, 4.5 and 4.6 will evaluate its effectiveness.

4.2. Event-Driven Information Handling for Large-Area Skin

This section presents the designs for realizing event-driven information handling for LASSs on standard computing systems without the need for specialized hardware. These designs allow for very flexible event-driven implementations in complex systems. These designs thus significantly contribute to this thesis' objective to realize efficient large-area tactile feedback in complex systems allowing machines to implement, for the first time, effective physical interactions with large-area contacts.

Section 4.2.1 details the realization of asynchronous computation on-demand in standard computing systems, a key element of event-driven information handling. Then, Section 4.2.2 presents the design of effective interfaces to efficiently exchange information between event-driven LASSs and clock-driven information consumers.

4.2.1. Event-Driven Information Handling on Standard Computing Systems

An event-driven LASS not only relies on event-driven sensing in its deployed e-skin, see Section 4.1, it also heavily relies on the handling of its information at higher processing layers, see Figure 24. Specialized bio-inspired event-driven computing systems with massively parallel computing capabilities are emerging [51, 67, 95, 123, 110, 38, 113], but require special hardware. While these systems provide very good performance, the bridging to CDSs, for example when connecting an event-driven system to a standard robot platform, requires complex algorithms and specialized hardware to decode events and provide information to the CDS. Therefore, this thesis proposes to employ the SoDP on standard computing systems to provide a more flexible and practical realization of event-driven information handling for LASSs – eliminating the needs for specialized hardware.

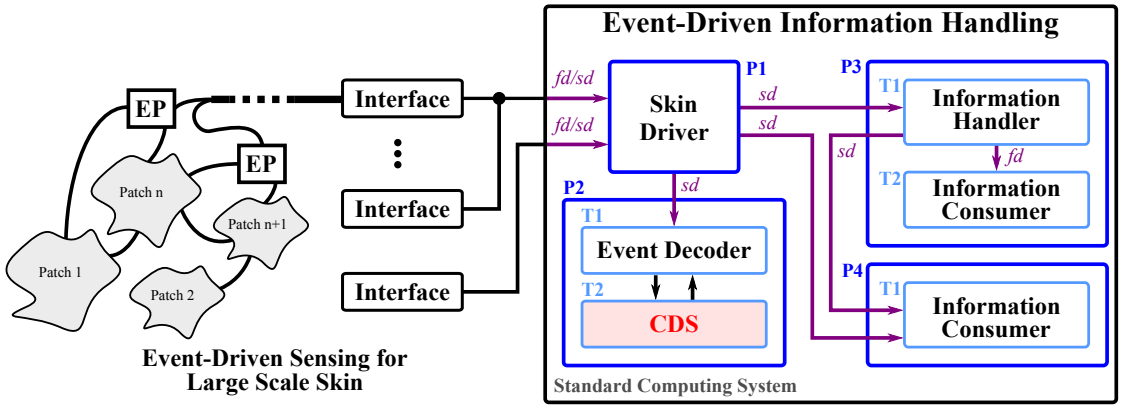


Figure 24 The Event-driven information handlers exploit the Signaled-Wakeup principle (Appendix B). The *fd* are file descriptors, while the *sd* are socket descriptors. The processes are denoted with P and the threads with T. The Process P2 contains a CDS and employs an event decoder to bridge from the event-driven to a clock-driven system. The endpoints (EP) between skin patches and interfaces refer to the realization of extended modularity in the interfaces (Appendix A, Figure 80).

The work presented in Section 4.2 was in part published in:
Bergner, F., Dean-Leon, E., Cheng, G., "Design and Realization of an Efficient Large-Area Event-Driven E-Skin". In: *Sensors* 20.7 (2020), p. 1965.
Copyright permissions: see Appendix D.

Standard computing systems utilize multi-core processors and multi-thread capable Operating Systems (OSs). These OSs schedule concurrent tasks by pausing the execution of a task (preemption) and resuming the execution of another task. In this way, tasks can share computation time and, as long as the switching between tasks is faster than their required reaction time, these operating systems effectively realize processing that is virtually concurrent and asynchronous. Multi-core systems improve concurrency since several computations can physically take place at the same time. OSs handle the fair splitting of computation time (timeslices) between tasks (processes and threads), ensuring that tasks respond as requested. To increase the efficiency of scheduling, tasks can yield their computation time and ask the operating system to resume on the occurrence of a signal. The yielding and resuming on the occurrence of a signal [75] is referred to as Signaled-Wakeup Principle (SWP), see Appendix B. This principle, combined with multi-threaded programs, allows us to realize an event-driven information handling system on standard computing systems. The implementation presented in this thesis is realized on a Linux standard OS (Section 4.3). The implementation could further profit from a real-time kernel with reduced latencies. The principles for realizing event-driven information handling on standard computing systems are available on many operating systems (e.g., Windows, Mac OS), thus, the implementation is not strictly limited to the Linux OS.

The resulting event-driven system consists of event generators and event consumers, see Figure 24. Event generators can be located inside and outside the computing system. Section 4.2.1.1 discusses their connection to the signals of an OS and Section 4.2.1.2 presents the design of event consumers.

4.2.1.1 Connecting Event Generators

Event generators have to trigger the signals (e.g., file/socket descriptors) of the operating system to exploit the Signaled-Wakeup Principle (SWP) for realizing event-driven information handling in standard computing systems. Then, the OS can wakeup and resume the computation of event consumers when new events arrive.

Event generators and event consumers may reside on different hardware, for example, the event generators of a large-area event-driven skin reside in the modules of the e-skin system and thus outside the computing system, see Figure 24. These external SoDP event generators, see Section 4.1.2, create event packets that are immediately sent to the communication network and forwarded to the information handling system. Most operating systems connect the arrival of network packets directly to signals (*fd*/*sd*), see Figure 24, such that the events of an external event generator eventually trigger the signals of the operating system.

Event generators residing inside the computing system may connect to the signals of the operating system (*fd*/*sd*), and thus connect to event consumers, in two manners. Threads that share the same process can use file descriptors (*fd*) to connect to event generators, see

process P3 in Figure 24. There, thread T1 (event generator) is connected to thread T2 (event consumer) via the file descriptor *fd*. Threads that are attributed to different processes can connect and share information through the local virtual network, see processes P3 and P4 in Figure 24. There, thread T1 (event generator) of process P3 is connected to thread T1 (event consumer) of process P4 via the socket descriptor *sd*.

4.2.1.2 Event Consumers—Event-Driven Programs

Event consumers are event-driven programs that handle information on the arrival of new events. Therefore, event consumers have to wait for events and stay inactive until events arrived. During their inactivity, event consumers pause their computations and yield the time-slices of all their threads. They then resume on the arrival of events. The yielding and resuming of threads are most efficient when triggered by the signals of the operating system. Therefore, event consumers have to connect to these signals (see Section 4.2.1.1) and utilize the SWP to wait until a signal is triggered. As a result, the event consumers become only active and handle information when novel information arrives. After handling the novel information, the event consumers may, as a result, create new events that wakeup other event consumers. Thread 1 of the process P3 in Figure 24 depicts such an event consumer.

Event dispatchers can ease the realization of event consumers. Event dispatchers wait for the activity of OS signals and then call the appropriate callback functions (event handlers) associated with the activity of an OS signal (event), see Figure 25. In this setup, the handling of events takes place in callback functions. Event dispatchers can additionally implement event queues to increase the processing efficiency of agglomerated events, that are, events that occurred almost at the same time.

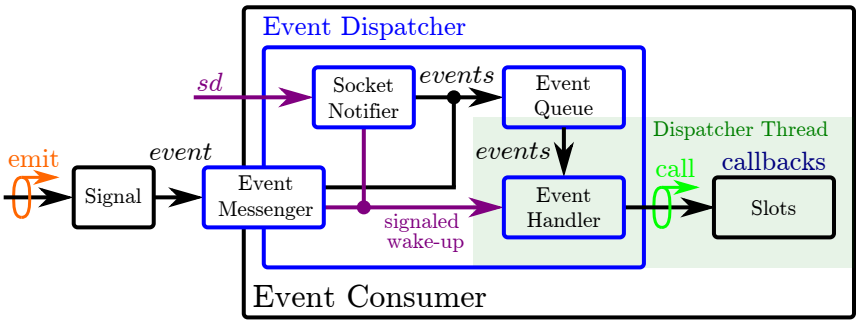


Figure 25 An event dispatcher combined with the signal-slot principle facilitates the connection of event generators and consumers within processes. An event handler implementing the Signaled-Wakeup Principle (SWP) dispatches events on their occurrence and drives the processing of events in the event consumer. This architecture allows multi-threaded signal-slot connections and eases the development of event generators and consumers.

4.2.2. Event Decoding

Event decoders realize the interface between the event-driven information handling system of a LASS and CDSs. At first glance, decoding events is not reasonable at all. However, event decoding can be seen as a compromise to utilize clock-driven algorithms that have

not been adapted yet for event-driven applications. Especially real-time low-level control for actuation in robots is often provided by third parties and cannot be easily modified. In general, guaranteeing the stability of the controllers depends on the theory that assumes that the Nyquist-Shannon sampling theorem is fulfilled at all times. This issue is addressed in event-driven control [99] that for itself is an emerging new research field. Nevertheless, up to the near future, EDSs will need to be combined with CDSs, and interfaces between both systems will be required. The interface from CDSs to EDSs is the event generator discussed in Section 4.1. The interface from EDSs to CDSs is the event decoder that will be detailed in this section.

4.2.2.1 Realizing Event Decoders

The realization of an event decoder has to provide a synchronous interface for accessing information that is updated on the arrival of new events, see Figure 26. This thesis proposes to realize the domain crossing from event-driven to CDSs by providing two completely different interfaces to a shared memory block. The event-driven interface employs the decoder to decode events to keys and values, then utilizes a fast mapping mechanism (Section 4.2.2.3) to lookup the memory index of a key. The key, the event decoder decodes of an event, depends on the implementation and the type of the event. The key could be the event ID, identifying the sensor where the event originated, or the module ID, identifying the module where the event emerged. The decoder could also combine the module ID with the sensor mask and compute the global ID of the sensor, see Figure 23. After mapping the key to an index, the event decoder stores the decoded value in the correct memory location (see Figure 26). Section 4.2.2.3 details the selection of a fast mapping mechanism for event decoders. The clock-driven interface, on the other hand, provides access to the shared memory block. Since the event decoder stores the information in a contiguous memory block, clock-driven algorithms can access and re-sample information at any time and loop through all memory locations with low performance penalties.

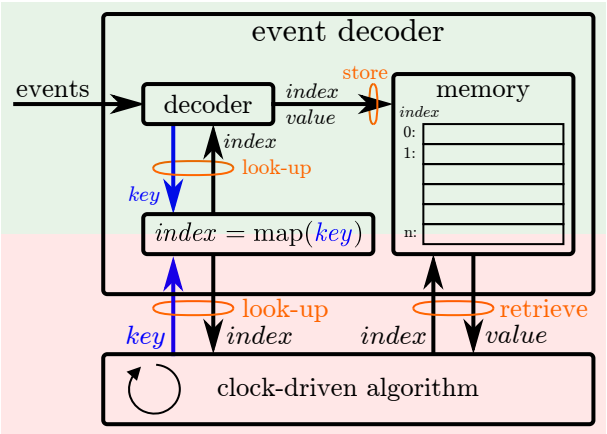


Figure 26 The event decoder provides an interface to transfer information from the event-driven domain (green background) to the clock-driven domain (red background).

Section 4.3.4 follows the presented design of event decoders in the realization of an event-driven LASS, and Chapter 5 demonstrates the effectiveness of this event decoder in systems that combine an event-driven LASS with clock-driven control algorithms.

4.2.2.2 Efficiency of Hybrid Event-Driven Systems

Hybrid event-driven systems containing clock-driven and event-driven components outperform pure CDSs. The overhead of the event decoder is comparable to the packet decoding in pure CDSs, which also requires the mapping from module IDs to indexes. Actually, on average, the event decoder outperforms the packet decoder of CDSs since its decoding and mapping resides in the event-driven domain, which is less active than the clock-driven domain. Consequently, even if all event consumers are clock-driven, an event-driven e-skin system outperforms a pure CDS, as demonstrated in our evaluation in Section 5.1.

4.2.2.3 Fast Mapping Mechanisms

The fast mapping between event IDs and memory indexes can be realized with associative arrays. Associative arrays realize a fast mapping of keys (IDs) to values (indexes) and can employ different mapping techniques [94, 76] such as direct addressing, self-balancing linear search trees, and hash tables, see Table 2. These mapping techniques exhibit different advantages and disadvantages, which are listed in Table 2.

Data Structure	**Avg. Search**	**WC. Search**	**Memory**
Direct Addressing	$O(1)$	$O(1)$	2^m
Self-Balancing Binary Search Tree	$O(\log n)$	$O(\log n)$	n
Hash Table	$O(1)$	$O(n)$	n

Table 2 The average and worst case (WC) search complexity, and the memory requirements for n elements with m bit keys.

Direct addressing is the fastest possible mapping at the cost of high memory demands. Since the address space for event keys is huge (event keys are usually grouped in meaningful subspaces) but only sparsely occupied, direct addressing is not a suitable option for LASSs.

Self-balancing binary search trees (SBBSTs) require much less memory space than direct addressing and offer a constant lookup complexity that depends on the height of the tree, see Table 2. Consequently, SBBSTs provide fast mapping for small sets of keys (less than 1000). A lookup in a map with 1000 keys, for example, has only the complexity of 11 comparisons.

Hash tables provide, on average, a better lookup complexity than SBBSTs. However, finding a good hash function is essential to avoid collisions, that is, the hash function maps two or more keys to the same bucket (set of indexes). In the worst case, all keys are mapped to the same bucket and the lookup complexity collapses to the performance of sequential searching, see Table 2. Thus, the definition of a good hash function depends on the distribution of keys, and if this distribution is known in advance, then the worst case can be avoided. Hash tables provide a good mapping performance for large key sets but are less suitable for small key sets, where the overhead of the hash function and the accessing of scattered memory significantly reduces its lookup performance.

In summary, event decoders in LASSs with up to 1000 modules/keys should employ SBBSTs, since the minor lookup complexity advantage of hash tables cannot justify the disadvantage of finding a good hash function and the penalty of scattered memory. Nevertheless, event decoders in LASSs with ten-thousands of modules will significantly profit from hash tables.

4.3. A Large-Area Event-Driven E-Skin System

This section introduces the e-skin system that was developed at the Institute for Cognitive Systems (ICS), Technical University of Munich (TUM). This thesis utilizes this e-skin as a basis for realizing its efficient event-driven designs for LASSs.

Since starting its development in the year 2010, the e-skin project evolved from developing the hardware of the skin modules, that is, the skin cells [101], to an e-skin system that encloses a self-organizing multi-modal sensing network of skin cells [101, 106, 109], an algorithm to self-calibrate the locations of its sensors [102], and multiple robot calibration algorithms for sensory-motor mappings [104, 107, 108, 109]. Even before introducing the event-driven approach [21, 16], this e-skin system proved its effectiveness in complex applications with real-time robot control for human-robot interaction [109, 45, 40].

Despite this e-skin system's unique capability to self-organize, which results in robustness, flexibility, self-localization, and scalability, the system lacks a systematic and scalable approach to handle a large amount of tactile information – an implication that the upscaling of the e-skin sensing area causes. This lack still limits the e-skin's capabilities for large-area feedback. This lack also explains why, in general, tactile feedback of LASSs is often limited to interactions with body parts rather than whole bodies, or limited in spatial or temporal resolution.

This thesis' objective is to overcome this issue with the realization of an efficient large-area e-skin by introducing the event-driven approach to e-skin systems. Therefore, this section presents the embedding of the event-driven approach (Chapter 3, and Sections 4.1 and 4.2) into this e-skin system. This integration will elevate this e-skin's capabilities from applications with a couple of hundreds of skin cells to thousands of skin cells. The initial modular e-skin system provides the ideal basis for realizing the event-driven designs presented in Sections 4.1 and 4.2.

The following sections first present the initial e-skin system, continue with enhancements to improve scalability, and then present the integration of the event-driven approach. Afterward, Section 4.3.4 introduces the e-skin systems that this thesis utilizes for validation, evaluation, and efficiency assessments. Then, Section 4.3.5 introduces the performance indicators this thesis employs for evaluations.

4.3.1. A Modularized Multi-Modal E-Skin

The core of the e-skin system [101, 106, 109] is its skin cells. They are the smart modules and form the basis of the LASS, see Section 4.1.1.1. The following sections present these skin cells, their sensing capabilities, and how they exploit modularity to achieve robustness and flexibility.

4.3.1.1 A Skin Cell – Smart Sensing Modules

A skin cell is the smallest element of the e-skin system. Its first version has been introduced in [101]. Since then, the hardware has been revised to the version presented in Figure 27, [106, 109]. The hardware version has not changed since [106, 109] but the firmware of the skin cell has been continuously improved, optimized, and modularized.

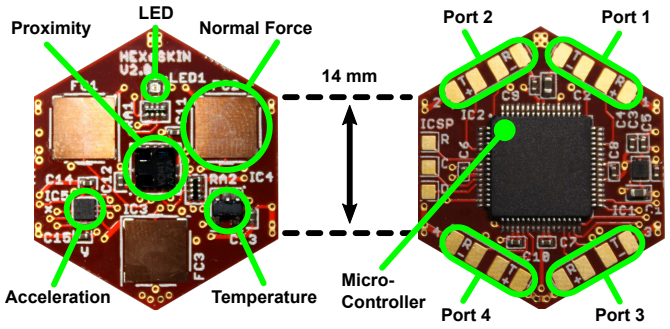


Figure 27 One skin cell of the e-skin system [106, 109]. The hardware has not changed since [106]. The red skin cells are of the last production cycle and have been deployed in large-area, see Figure 30.

A skin cell is hexagonally shaped. The hexagonal shape [105] serves mainly two purposes: First, in contrast to triangles, hexagons border to neighbors always by an edge and not by corners. Wired connections are only possible via edges. Thus a hexagon can connect to all neighbors, allowing for meshed connections rather than connection trees. Second, similar to triangles, hexagons can tessellate surfaces, an important feature since an assembly of skin cells should be able to cover surfaces without gaps. Additionally, the regular pattern of skin cells on surfaces supports the self-localization of skin cells on surfaces within the 3D surface reconstruction algorithm of [102].

The skin cell is the smart sensing module of the e-skin system. All skin cells are identical. They employ the same set of sensors and have the same set of capabilities. The top side of a skin cell mainly allocates its sensors: Three capacitive normal force sensors, one optical proximity sensor, a three-axis accelerometer, and two temperature sensors, see Figure 27. The bottom side of a skin cell accommodates a programmable microcontroller with a 16-bit Reduced Instruction Set Computer (RISC) architecture, and four serial communication ports (UART). The microcontroller provides local processing capabilities that combined with modularity and decentralized algorithms [101, 109] realize smart modules as referred to in Section 4.1.1.1.

4.3.1.2 Multi-Modal Sensing – Complex Tactile Information

Each skin cell integrates the same set of sensors perceiving cutaneous (tactile) stimuli from the environment. A skin cell of the e-skin system is rather a sensor platform than a sensor

or a collection of sensors. A skin cell is not limited to the sensors introduced in [101, 106, 109] and newer hardware revisions might use different sets of sensors further enhancing the sensing capabilities of the e-skin system.

The sensors of the skin cell are used to sense different modalities, they sense different physical measures of contacts, similar to the specialized receptors in nature (Section 2.1). The 3D accelerometer senses vibrations and net-linear accelerations, the three force sensors measure normal contact forces, the proximity sensor measures distances to close objects, and the temperature sensors measure thermal contact properties. In sum, the sensors on the skin cells provide *multi-modal contact* and *pre-contact information*.

The sensors of the skin cell are clock-driven sensors. Their sampling rate can be collectively changed to 0 Hz, 62.5 Hz, 125 Hz, and 250 Hz. A sampling rate of 0 Hz refers to an *idle* skin cell, that is, a skin cell that does not send any information, neither data packets in the clock-driven, nor event packets in the event-driven operation mode.

4.3.1.3 Modularization – Robustness & Flexibility

The modular e-skin, with its skin cells, realizes a robust and flexible system. It achieves robustness through connection redundancy and self-organization [109]. Each skin cell is connected to up to four neighbor cells and together they build up a meshed network of skin cells with multiple redundant paths for communication and power. If one connection fails, then a redundant connection can take over. A self-organizing algorithm finds connections and communication paths between skin cells when the e-skin system is started and initialized [109]. Communication paths (communication trees) are automatically built such that each skin cell receives information (downstream) from the centralized coordinator, usually a computer, and can send information back (upstream). Since these communication paths are built during every startup cycle of the system, the system automatically avoids broken connections. More recently, a dynamic routing algorithm [7, 8] enhancing the self-organization of the skin cells was introduced. This algorithm balances the communication trees, and re-routes communication paths on the occurrence of connection failures on-line, in real-time, even when the system is already in operation.

The e-skin system achieves flexibility through the combination of modularity and self-organization. Modularity allows attaching, removing skin cells, or changing the structure of the network of skin cells by adding or removing wires with little effort. The self-organization of the network of skin cells then automatically finds new communication paths. Furthermore, the in-system re-programmability of the skin cells, when they are already assembled to large networks, enables great development flexibility. The ability to change the firmware of the skin cells allows the exploitation of their local processing capabilities for specific needs, e.g. implementing event generators (Section 4.3.3) or distributively computed algorithms (Section 5.2).

4.3.2. Towards Large-Area E-Skin – From Cells to Patches to Networks

The previous section introduced the modular skin cells and explained that connected skin cells form redundant, meshed, and self-organized networks. In the following, the structural organization of large networks of skin cells and their links to generic high-speed networks of computer systems is presented.

4.3.2.1 Skin Patches – Directly Connected Skin Cells

A *skin patch* is formed of skin cells that are tessellated into hexagonal grids, directly connected via Flex-Printed Circuit Board (PCB) connectors, and covered with a silicone protection layer, see Figure 28.

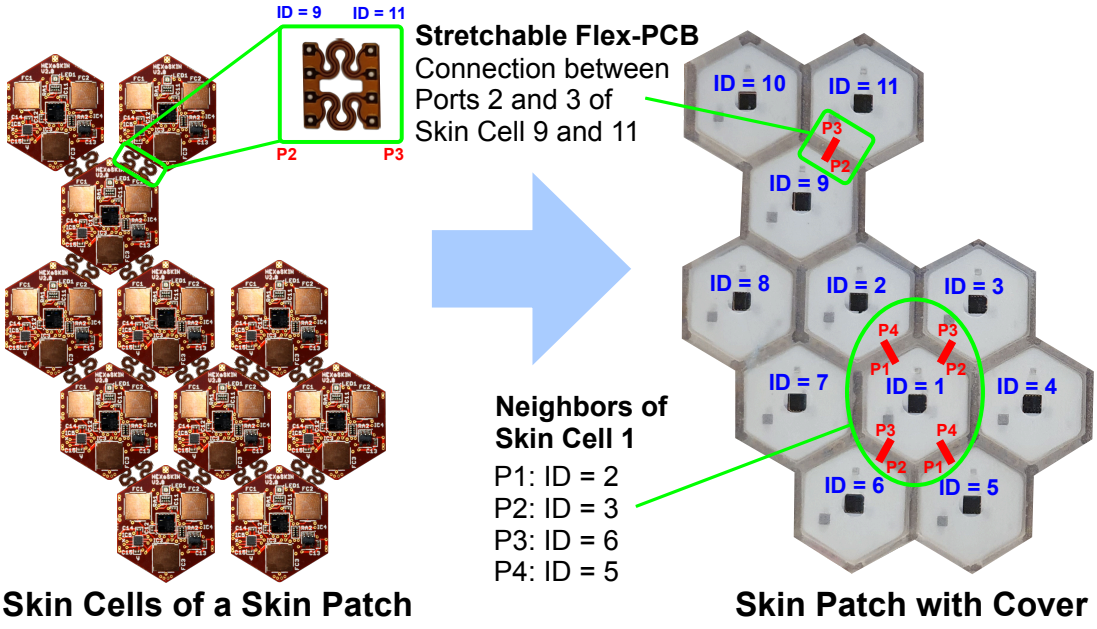


Figure 28 Skin cells directly connected to each other and tessellated into a hexagonal grid form a skin patch.

Skin cells are connected via ports, see Figure 27. Port one (P1) of a skin cell is connected to port four (P4) of another skin cell and respectively port two (P2) to port three (P3). These ports provide bi-directional serial communication interfaces (UART) and power.

A skin cell is identified by a unique ID in the network of skin cells, that is the *skin cell network*. This ID is automatically assigned during the self-organizing startup sequence of the e-skin network [109]. Storing this ID into the non-volatile memory of the skin cell ensures that once an ID is assigned, it no longer changes. IDs allow to map information to its origin (upstream) or send commands to specific skin cells (downstream). When combining the IDs of the neighbors of a skin cell with their port number (1 to 4), this *neighbor information* provides the 2D structure/shape of a skin patch.

In the skin cell network, information is passed on by forwarding fixed size datagram/packets from skin cell to skin cell along the self-organized communication tree. A packet in the skin cell network has a constant size of 20 B (Bytes). The skin cells exploit the Direct Memory Accesss (DMAs) channels of their microcontrollers to efficiently route packets, and thus, the skin cells cannot operate with changing packet sizes. A 20 B packet with an Interpacket Gap (IPG) of 10 µs occupies 240 bit on the 4 Mbit/s UART communication bus.

The size of a skin patch, that is, the number of skin cells a skin patch can contain, is limited by the communication bandwidth and the power it is supplied with. Assuming that a UART connection can be utilized to its physical limit of 4 Mbit/s, and that the skin patch supports the maximum clock-driven sample rate of 250 Hz, then one UART connection can support up to 66 skin cells.

4.3.2.2 The Skin Cell Network – Network of Skin Patches

The *skin cell network* describes the self-organized network of skin cells where all skin cells are connected to each other by UART connections and exchange 20 B packets (240 bit on the bus). The skin cell network not only spans between skin cells in a skin patch but also between patches, where effectively skin cells of the skin patches are connected by longer cables. Similar to the network in skin patches, the size of skin cell networks spanning in a network of skin patches is only limited by the provided communication bandwidth and power. A network of skin patches has to include at least enough inter skin patch connections to collectively support the number of skin cells. More connections improve redundancy. If the connection redundancy between skin patches is large enough, then the network can also compensate for the loss of inter skin patch connections. An example of a skin cell network can be found in Appendix A, Figure 79.

4.3.2.3 Tactile Section Units – Bridge to Generic High Speed Networks

The protocol of the skin cell network, see Sections 4.3.2.1 and 4.3.2.2, is a highly customized protocol optimized for efficiency in inter skin cell communication. Furthermore, the communication capacity of 4 Mbit UART connections is limited to at most 66 skin cells (1.6 kpacket/s). These barriers can be overcome by introducing interfaces that translate packets between the skin cell network and standardized high speed communication protocols. The e-skin system uses interfaces, coined Tactile Section Units (TSUs), that bridge the skin cell network to Ethernet networks and convert skin packets to standard UDP packets and vice versa (Appendix A). A 1 Gbit/s Ethernet connection can theoretically provide the communication bandwidth for up to 6242 skin cells (1.56 Mpacket/s).

4.3.3. Embedding the Event-Driven Operation Mode

This section presents the embedding of this thesis' event-driven design for e-skin (Section 4.1) in the previously presented modular e-skin. The skin cells of this e-skin provide the required local processing capabilities. Furthermore, the e-skin's ability to perform in-system firmware updates allow for embedding the event-driven approach in existing and already deployed skin systems. After embedding the event-driven approach, the skin cells support two operation modes: The *clock-driven operation mode* of the initial skin system [101, 109], and the novel *event-driven operation mode* [21, 16].

Embedding the event-driven mode requires the definition of an event packet, an event generator (Section 4.1.2), and an event unpacker. Analog to the data packets that convey the information of all the sensors of a skin cell, the event packets convey the events of a skin cell. The size of packets in the skin cell network is fixed to 20 B (Section 4.3.2.1). An event packet thus can utilize 20 B to store the ID of the skin cell, the event type mask, and the values of six events, see Figure 29. The event type mask encodes which of the skin cell's sensors triggered an event. Since only six event values fit into an event packet, the event type mask also encodes if the packet is the first containing the first six events, or the second containing the remaining events. The different structure of the event packet, with an event type mask and a generic storage space for 16 bit values, reduces its information encoding density in comparison to the standard data packet used in the clock-driven mode. Noticeably, based on the validation of the event-driven system (Section 4.4), rarely more than three events occur at the same time. A skin cell will seldom have to send two packets because events occurred on all the sensors at the same time.

Byte 1	Data Packet Header
Byte 2 - 3	Cell ID (14 bit)
Byte 4 - 19	9 Sensor Values (104 bit)
Byte 20	End of Packet (EOF)

(a) The format of a skin data packet as used in the clock-driven operation mode

Byte 1	Event Packet Header
Byte 2 - 3	Cell ID (14 bit)
Byte 4 - 5	Event Type Mask (9 bit + 1 bit) 9 bit $\rightarrow$ 9 Event Types for 9 Sensors 1 bit $\rightarrow$ Packet ID (Packet 1 or 2)
Byte 6 - 19	6 $\times$ 16 bit values for 6 events
Byte 20	End of Packet (EOF)

(b) The format of a skin event packet as used in the event-driven operation mode

Figure 29 The format of a data and an event packet in the skin cell network.

The event generation Algorithm 1 implements Figures 22 and 23 of the modular event generator introduced in Section 4.1.2 in the skin cells' firmware.

The work presented in Section 4.3.3 was in part published in:
Bergner, F., Dean-Leon, E., Cheng, G., "Event-based signaling for large-scale artificial robotic skin - Realization and performance evaluation". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, Korea, 2016, pp. 4918–4924.
Copyright permissions: see Appendix D.

Algorithm 1 Event Generation

```
 1: values_prev := 0
 2: loop
 3:     update values
 4:     if mode == CLOCK_DRIVEN then
 5:         create data packet of values
 6:         send data packet
 7:         continue
 8:     end if
 9:     values_diff := | values − values_prev |
10:     event_mask := 0
11:     for s = 1 to 9 do
12:         if values_diff[s] > values_thresh[s] then
13:             set bit for s in event_mask
14:             values_prev[s] := values[s]
15:         end if
16:     end for
17:     if any s of event_mask is enabled then
18:         create event packet
19:         send event packet
20:     end if
21: end loop
```

The firmware of the skin cell utilizes a data structure *values* that stores the sensor values of the skin cell. First, the storage *values_prev* for the previously reported event values of the sensors is initialized to zero (Line 1). The infinite sensor monitoring loop is entered. First, the *values* structure is updated with the sampled sensor data. Then, if the skin cell is operating in clock-driven mode, a data packet containing the information of *values* is created and sent. The algorithm continues at Line 2. If the skin cell is operating in event-driven mode, then the block between Lines 4 and 8 is skipped. Then, the algorithm proceeds in Line 9 with computing the absolute difference between the values of the last reported events and the current sensor values. Before checking if an event threshold of a sensor has been passed (Line 12), the event type mask for novel events is reset to zero (Line 10). The block of Line 11 to Line 16 checks for each sensor if an event occurred. If this is the case, then the sensor is marked in the event type mask (Line 13) and the sensor's value is updated in the container for previous event values (Line 14). If a sensor is enabled in the event-driven operation mode and if it is marked in the event type mask, then an event packet is created and sent (Lines 17 to 20). An event packet only contains events of enabled sensors.

Algorithm 2 describes how to unpack event packets in the computer system. Unpacking is the process of extracting the individual events of a skin cell form an event packet. The Algorithm implements the unpacking following the event-driven processing of information as depicted in Section 4.2.

Algorithm 2 Event Unpacking

```
 1: loop
 2:     select sd
 3:     fill event_packet_queue
 4:     for each event packet in event_packet_queue do
 5:         get event_mask
 6:         get cell_id
 7:         for each bit in event_mask do
 8:             get event_type
 9:             get event_value
10:             create event e
11:             append e to event_list
12:         end for
13:     end for
14:     publish event_list
15: end loop
```

The event packet unpacking thread yields its context until at least one UDP packet (event packet) arrives (Line 2). Then, the event packet queue is filled (Line 3). Afterward, each event packet is unpacked (Lines 4 to 13). Therefore, the unpacker decodes the event type and value (Lines 8 and 9) of each event from the event type mask of the packet (Lines 7 to 12). The events are created containing the skin cell ID, the event type, and the event value (Line 10), and appended to the list of events (Line 11). Eventually, the list of events is published (Line 14). The publishing triggers a post-processing step that signals the event handlers, or respectively the consumer threads.

4.3.4. Large-Area E-Skin on Robots – Validation on Complex Systems

This section presents the large-area e-skin systems that this thesis uses for validating and evaluating its designs for efficient event-driven LASSs. The selected e-skin systems are deployed on robots. Robots are complex systems, and an e-skin on a robot allows for evaluating the very challenging integration of large-area tactile feedback in a complete real-time perception-action loop. These evaluations can provide empirical evidence that this thesis' event-driven approach is successful in delivering tactile feedback of large contact areas to control algorithms that enable large-area physical interactions in machines/robots.

4.3.4.1 Large-Area Deployments of E-Skin

This thesis focuses on two LASSs deployed on two different robots, see Figure 30. One LASS is deployed on an Univeral Robots (UR) robot arm (UR5) [42], and one is deployed on the large surface area of the humanoid robot H1 [29, 20]. This section focuses on introducing these e-skin systems. The subsequent sections of this chapter use these systems to validate and evaluate the efficiency of this thesis' event-driven approach and designs in real e-skin systems.

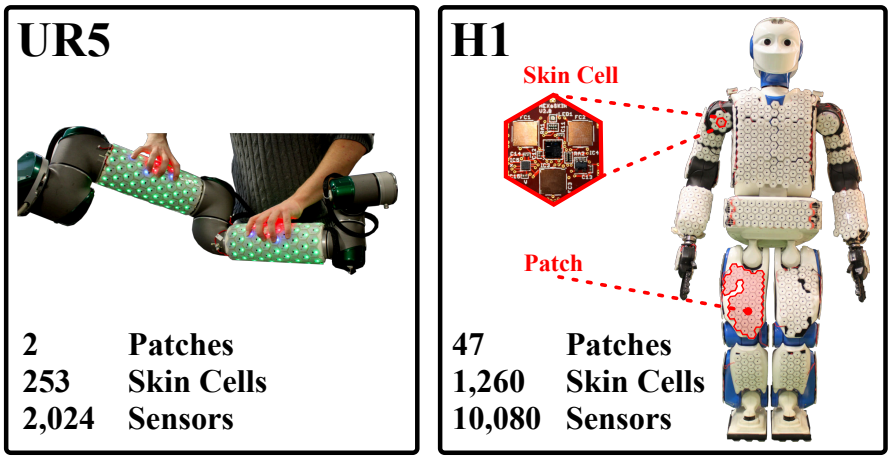


Figure 30 The two Large-Area Skin Systems (LASSs) used as experimental platforms for validation and evaluation.

Furthermore, the robot platforms of these two e-skin systems are suitable for validating and evaluating the e-skins and their event-driven approach in physical human-robot interactions where they provide large-area tactile feedback in the control algorithms of the robots. That is the focus of Chapter 5. Chapter 5 then introduces the required details of the robots.

The e-skin on the UR5 robot arm covers the robot's lower and upper arm (Figure 30). The e-skin employs 253 skin cells in two skin patches (two groups of directly connected skin cells, see Section 4.3.2.1). One skin patch covers the upper arm (143 skin cells) and one the lower arm (110 skin cells). In total, the e-skin covers an area of 0.17 m^2 with 2024 multi-modal tactile sensors. The two skin patches are connected to the robot's computer via four cables and one interface (Tactile Section Unit (TSU), Section 4.3.2.3). Table 3 summarizes the structure of the e-skin. Furthermore, Table 4 presents the nominal packet rates and the communication bandwidth in the Ethernet connection to the computer system, both for the e-skin in clock-driven mode.

TSU	Acronym	# Patches	# Cells
Right Arm	RA	2	253

Table 3 The e-skin on one UR5 robot arm. Two patches contain 253 skin cells and connect to one Tactile Section Unit (TSU). The Tactile Section Unit (TSU) is almost utilized to its capacity and the setup provides only limited cable redundancy.

Sample Rate [Hz]	Packet Rate [kpacket/s]	Bandwidth [Mbit/s]
62.5	15.812	10.375
125	31.625	20.750
250	63.250	41.500

Table 4 The nominal packet rates and bandwidths for 253 skin cells for the different sample rates. The nominal bandwidth in the Ethernet network considers UDP packets on the wire that have a size of 86 B, or respectively, 688 bit. For more information on packet sizes, see Section 4.3.2 and Appendix A.

The subsequent sections utilize this e-skin on the UR5 robot arm as the platform for the early proof of concept validations. The e-skin is distributed in a large area but still contains a moderate number of tactile sensors and induces manageable communication bandwidths. Thus, a standard clock-driven e-skin system can handle the tactile feedback of this e-skin [109, 40] whereby the system allows for direct comparisons between clock-driven and event-driven systems. These direct comparisons provide a reliable basis for consistent performance evaluations in the following sections.

The e-skin on the humanoid robot H1 covers its arms, legs, hands, feet, torso, and hip (Figure 30). The e-skin covers the body of the humanoid robot almost completely. The e-skin employs 1260 skin cells in 47 skin patches. Table 5 summarizes the number of skin patches per body part. For instance, two skin patches cover the upper arm and five skin patches the lower arm. In total, the large-area e-skin covers an area of 0.87 m^2 with 10 080 multi-modal tactile sensors. The 47 skin patches are connected to the robots's computer via 12 interfaces (Tactile Section Unit Satellite (TSU-S), Section 4.3.2.3). Table 5 summarizes the structure of the e-skin. Furthermore, Table 6 presents the nominal packet rates and the communication bandwidth in the Ethernet connection to the computer system, both for the e-skin in clock-driven mode.

Tactile Section Unit Satellite (TSU-S)	Acronym	# Patches	# Cells	TSU-S	Acronym	# Patches	# Cells
Right Upper Arm	RUA	2	47	Left Upper Arm	LUA	2	47
Right Lower Arm	RLA	5	151	Left Lower Arm	LLA	5	151
Right Torso	RT	4	87	Left Torso	LT	4	88
Right Flank	RF	4	154	Left Flank	LF	3	114
Right Upper Leg	RUL	5	108	Left Upper Leg	LUL	5	109
Right Lower Leg	RLL	4	102	Left Lower Leg	LLL	4	102

Table 5 The e-skin on the limbs and torso of H1. 47 patches contain in total 1260 skin cells. The patches are connected to 12 TSU-Ss. The TSU-Ss provide connection redundancy and body-part-wise modularity.

Sample Rate [Hz]	Packet Rate [kpacket/s]	Bandwidth [Mbit/s]
62.5	78.750	51.670
125	157.50	103.34
250	315.00	206.68

Table 6 The nominal packet rates and bandwidths for 1260 skin cells for the different sample rates. The nominal bandwidth in the Ethernet network considers UDP packets on the wire which have a size of 86 B or respectively 688 bit. For more information on packet sizes, see Section 4.3.2 and Appendix A.

The high nominal communication bandwidths (up-to 206.68 Mbit/s for a sample rate of 250 Hz) emphasize the importance of this thesis' event-driven approach for realizing an efficient large-area e-skin. In the standard clock-driven operation mode, this e-skin system can no longer handle its tactile information [20] (Section 4.6). After the validation and evaluation of the event-driven approach for efficient LASSs with the e-skin on the UR5 robot arm, the large-area e-skin on the humanoid robot H1 serves as the ideal platform to confirm the obtained

results with a truly LASS with more than 10 000 sensors. The e-skin on H1 furthermore covers large areas of H1 such that this humanoid robot is the ideal platform to implement previously infeasible and very demanding physical whole-body interactions that utilize the large-area feedback of the event-driven e-skin (Chapter 5). Overall, H1, with its large-area e-skin, is the perfect platform to emphasize the impacts of this thesis. The platform is suitable to investigate if the event-driven large-area e-skin succeeds in supporting the implementations of whole-body interactions on a humanoid robot with tactile feedback whereby standard clock-driven approaches fail.

4.3.4.2 A Flexible Information Handling System for Complex Applications

The e-skin system handles tactile information on standard computing platforms. The information handling system that has been developed can fully support the clock-driven and the event-driven operation mode of the e-skin system. The system architecture follows the design descriptions of Section 4.2.1 and exploits the threading and signaling capabilities of modern Operating Systems (OSs) to realize Event-Driven Systems (EDSs) on standard computer hardware. Furthermore, the architecture is modular easing development with reusable functional blocks and ensuring their flexible interchangeability. The realized information handling system allows for complex system integrations, for instance, in robotic platforms such as the 6-DOF UR5 robot arm and the humanoid robot H1. Appendix C presents further implementation details.

4.3.5. Performance Indicators – Definition and Measurement

The validation of the event-driven approach bases on a consistent and comprehensive performance analysis comparing clock-driven setups with hybrid or event-driven counterparts. First, the comparison between these systems is only consistent when the system achieves the same or comparable results. That is, the error between the clock-driven and the event-driven information is small, and the controllers consuming clock-driven or event-driven information exhibit the same behavior/reaction. In order to validate the effectiveness of comparable systems in clock- and event-driven, or hybrid operation, two performance indicators are used: 1) the CPU usage, and 2) the packet rate. A limited low packet rate is essential for a stable communication in skin cell networks. Furthermore, a low CPU usage on the computer system that handles tactile information improves scalability and enables compact autonomous applications such as mobile robots with tactile sensing capabilities, prostheses, or smart objects.

The work presented in Section 4.3.5 was in part published in:
Bergner, F., Dean-Leon, E., Cheng, G., “Event-based signaling for large-scale artificial robotic skin - Realization and performance evaluation”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, Korea, 2016, pp. 4918–4924, and
Bergner, F., Dean-Leon, E., Guadarrama-Olvera, J. R., Cheng, G., “Evaluation of a Large Scale Event Driven Robot Skin”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4247–4254.

4.3.5.1 Performance Indicators and their Dependencies

The packet rate is closely related to the CPU usage generated by handling and consuming information. In CDSs, all processing is dictated by the sample rate. Furthermore, since acquiring one sample of the skin cell's sensors equals to one skin cell packet, the packet rate directly correlates to the sample rate in CDSs, see Figure 31. Thus, for CDSs, a higher sample rate, results in higher packet rates, and a higher CPU usage.

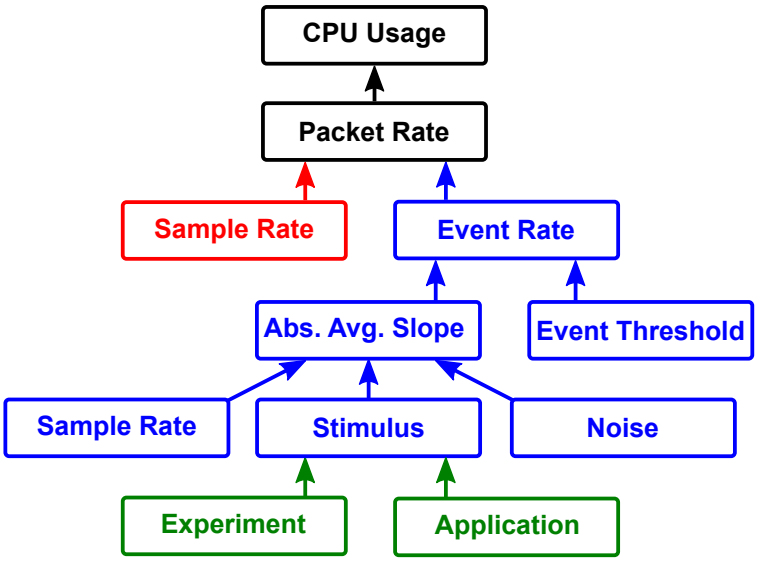


Figure 31 Performance indicator dependency tree.

In EDSs communication and processing is triggered by the arrival of events. Thus, the event rate relates to the event packet rate and the CPU usage. The event rate itself depends on many factors, as has been detailed in Section 3.1. The event rate is directly influenced by the selection of the event threshold δ and the absolute average slope $\left|\overline{\dot{x}_i}\right|$ of the stimulus signal $x(t)$. That is, in contrast to CDSs, the tactile experiment/application itself will influence the performance indicators in EDSs. Thus, to ensure the comparability to CDSs, the experimental design has to cover the full range of realistic event rates that could occur in the application. Besides the stimulus shape, noise and sampling rate have an indirect influence on the event rate. A higher sampling rate increases the event detection rate by increasing the system bandwidth, and a higher noise level increases the number of events triggered by noise.

4.3.5.2 The Packet Rate Indicator

The packet rate indicator measures the number of packets the computer receives from the e-skin. It considers all received packets before they are potentially dropped by a saturated Operating System (OS) or by subsequent processing steps. Thus, this indicator can account for packets lost in the skin cell network or the interfaces (TSU, TSU-S, and TSU-LB, Section 4.3.2.3), especially when the measured packet rate is significantly smaller than the nominal

packet rate in clock-driven mode. The nominal packet rate f_{p} in clock-driven mode is defined by the sampling rate f_{s} and the number of skin cells n

$$f_{\mathrm{p}} = n\, f_{\mathrm{s}}. \tag{4.7}$$

The nominal packet rates of the e-skin systems used for validation can be found in Tables 4 and 6, and in Section 4.3.4.1.

4.3.5.3 The CPU Usage Indicator of the Skin Driver

The CPU usage indicator of the skin driver considers the CPU usage demanded for unpacking the skin packets in the skin driver and forwarding the information to consumers. In the clock-driven operation mode, the indicator takes into account the unpacking of data packets and in the event-driven mode, respectively, the unpacking of event packets. In a hybrid setup, where the skin operates in event-driven mode and the skin driver provides information to clock-driven consumers, the CPU usage indicator additionally includes the event decoder (Appendix C, Figures 81 and 82).

The unpacking of packets and the resulting CPU usage can be described by a CPU usage model. The theoretical background for this model has been introduced in Section 3.3.2.1 and will be validated with real measurements in Sections 4.5 and 4.6. The complexity of unpacking a data packet or an event packet can be assumed to be almost identical.

4.3.5.4 The CPU Usage Indicator of Information Consumers

The CPU usage indicator of information consumers, both clock-driven and event-driven, consider the CPU usage of the whole process. Unlike the skin driver, the consumers usually have a linear processing pipeline without special branches for the clock-driven or the event-driven mode. The process is either completely clock-driven, or event-driven. In a hybrid setup, the process is still clock-driven, and the communication link to the skin driver is identical, whether the e-skin operates in the clock-driven or the event-driven mode (Appendix C, Figure 81). The difference lies in the information passed in the communication link. Since these differences are agnostic to the internals of the consumer process, it is sufficient to monitor only the CPU usage of the whole process.

4.3.5.5 The Total CPU Usage Indicator

The total CPU usage indicator sums up the CPU usage of all executed processes. This indicator naturally includes the CPU usage generated by the information handling system of the e-skin, its performance monitors for all indicators, and the CPU usage of the OS. The total

CPU usage indicator depicts the amount of processing capacities that are consumed. It also indicates when the system saturates.

4.3.5.6 The Packet Drops Indicator

The packet drops indicator quantifies the number of packets the OS drops in the communication link between the network card and the skin driver. Sparse packet drops indicate that the skin driver couldn't handle a burst of skin packets, while continuous packet drops indicate that the OS couldn't assign enough computation time to the skin driver. In the latter case, either the OS or the skin driver was saturated. The system OS is saturated when the total CPU usage indicator approaches its physical maximum (number of CPUs times 100 %), and the skin driver is saturated when its packet unpacking thread reaches 100 %.

4.3.5.7 Measuring Performance Indicators

To monitor the performance indicators during experiments, monitoring applications were created. These applications read pseudo files in the Linux */proc* and */sys* file system to gather information for the packet rate, the packet drops, and the CPU usage indicators for the skin drivers and consumers. The total CPU usage indicator uses information of the *turbostat* application for Intel CPUs. The *turbostat* application additionally provides information on CPU temperature, CPU power, and CPU frequency. The communication traffic in Robot Operating System (ROS) communication links is acquired by utilizing the packet capture library *pcap*.

The monitoring applications are connected to ROS such that the measured performance indicator values can be recorded next to other values such as tactile information, information computed by consumers, information of controllers, or information of robots. All recordings are timestamped to ease synchronization in the evaluation process.

4.4. Effectiveness of Event Generation

This section presents the general validation of this thesis' event-driven approach that aims to significantly improve the efficiency of the tactile feedback provided by Large-Area Skin Systems (LASSs). The presented efficiency analysis collects experimental tactile information of an e-skin in clock-driven operation and subsequently generates events off-line. The integration of the event-driven operation mode into the e-skin system follows later in Section 4.3.3. Overall, this section provides first evidence on the efficiency of the event-driven approach in LASSs and delivers the insights that later ensure the optimal realization of the approach in the e-skin. This section presents the validation and evaluations as follows.

First, Section 4.4.1 describes the experimental setup and the acquisition of the experimental data. Then, Section 4.4.2 analyzes noise profiles that, according to Section 3.1.6, provide a first reference for event thresholds that minimize the contribution of noise to event rates. Afterward, Section 4.4.3 analyzes the encoding error of events. It furthermore assesses event rates for different stimulation profiles and event thresholds. Together, the results of this analysis provide the references for selecting event thresholds that minimize the encoding error for a wide range of realistic stimulus profiles. Reflecting on the acquired results, Section 4.4.4 discusses the best compromise of good event thresholds for the e-skin's sensors. Therefore, it follows the parameterization guidelines of Section 3.1.7. Employing these event thresholds, Section 4.4.5 presents the resulting event rates for different tactile stimulations on an e-skin. These results additionally present how often events of different sensors of a skin cell occur at the same time. Drawing from these results, Section 4.4.6 determines the optimal event packet size that minimizes overhead in transmission.

The work presented in Section 4.4 was in part published in:
Bergner, F., Mittendorfer, P., Dean-Leon, E., Cheng, G., "Event-based signaling for reducing required data rates and processing power in a large-scale artificial robotic skin". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. Hamburg, Germany, 2015, pp. 2124–2129.

4.4.1. Experimental Setup and Protocol – Off-line Event Generation

The analysis in the following sections bases on the clock-driven raw sensor data acquisition of 253 skin cells mounted on a UR5 robot arm of Univeral Robots (UR), see illustration in Figure 32.

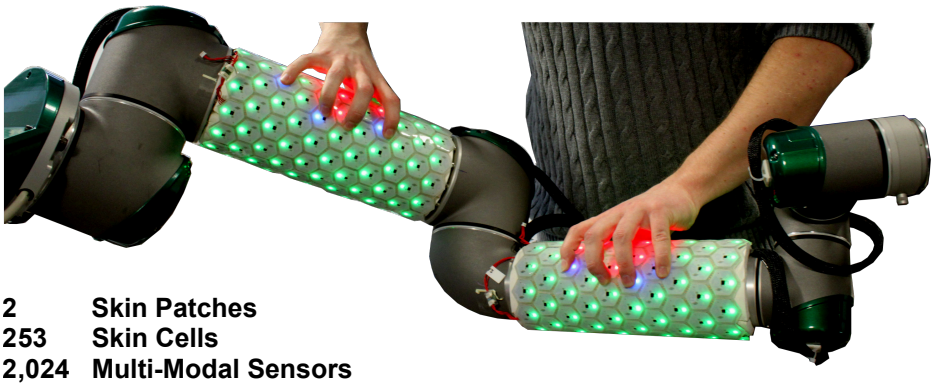


Figure 32 An UR5 robot arm covered with e-skin. The high-resolution temperature sensor is not deployed in this setup and the skin cells report a constant value for this sensor. Different stimulus profiles are applied to investigate the improvement of the transmission rate in the event-driven mode for various scenarios between the best case and the worst case.

To gather an appropriate data basis for the different investigations of this analysis, this thesis designed experiments that focus on generating multi-modal tactile stimuli representative for a wide range of applications. The experiments target stimulations that are components of realistic interactions, such as stroking, punching, teasing, and motion stimuli of a moving robot arm. Furthermore, the absence of external stimuli in an *idle* system provides insights on noise. An idle (resting) system represents the best case for Event-Driven Systems (EDSs) since an optimal EDS should not create any events when the information rate of its sensors is zero (Section 3.1). The measurements will show that a trade-off between sensitivity and noise prevents an event rate of zero in real systems.

In each experiment, the system collects the raw sensor values of all skin cells in clock-driven mode with a sampling frequency f_S of 62.5 Hz for around 20 s. Then, the collected samples are imported and analyzed in Matlab. The off-line evaluation in Matlab also comprises the event generators and the event decoders as introduced in Sections 4.1.2 and 4.2.2. The event generator computes the events and the event decoder reconstructs the original signal from these events.

4.4.2. Distribution of Differences – Noise Distributions and Event Thresholds

Sections 3.1.6 and 3.1.7 presented the relation between noise and event thresholds. An optimal parameterization of the event threshold considers the distribution of the Additive White Gaussian Noise (AWGN) to minimize the number of events triggered by noise, see Equation (3.63). This section presents the acquisition of these noise distributions for the e-skin's sensors on the experimental platform.

The signal $x(t)$ of one of the skin cell's sensors composes of the stimulus $s(t)$ and the additive noise (AWGN) $z(t)$, see Equation (3.61). The noise $z(t)$ is normally distributed with an expected value μ_{Z} of zero and a variance $\mathrm{Var}(Z) = \sigma_{\mathrm{Z}}^2$, see Equation (3.62). The stochastic process is a time-series

$$\{X_t\} = \{S_t\} + Z, \tag{4.8}$$

where $\{S_t\}$ is the stochastic process of the stimulus and Z is the random variable of the noise.

The sensor values of all skin cells on the UR5 robot arm are acquired in clock-driven mode (Section 4.4.1) with a sample frequency f_{s} of 62.5 Hz for 20 s. The robot arm is not moving and the skin is not stimulated, that is the skin system is considered idle. In this case, the stochastic process $\{S_t\}$ of the stimulus is a constant offset ξ. Then, the stochastic process of a sensor is

$$\{X_t\} = Z + \xi \tag{4.9}$$

and thus stationary. The noise described by the random variable Z is identical for all sensors and not correlated.

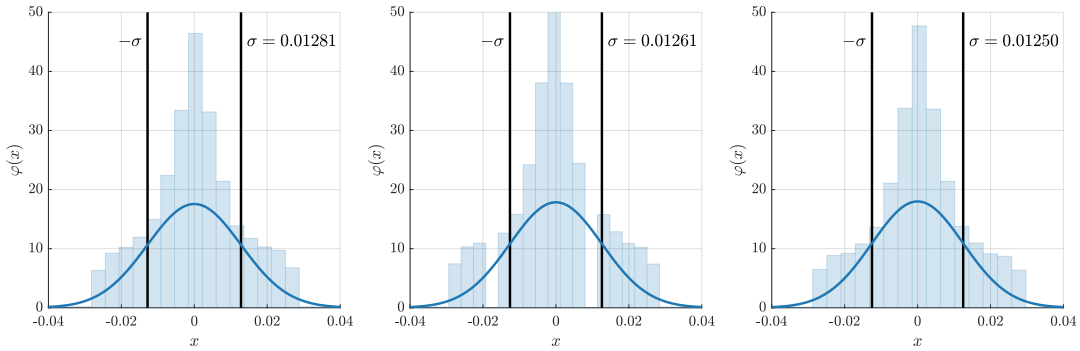
The evaluation focuses on the general noise distribution for particular sensor modalities. Therefore, the evaluation combines the realizations Z of one specific sensor, for instance, the force sensor S1, for all 253 skin cells. The combination of realizations then not only accounts for noise but also for stochastic uncertainties in the group of sensors. However, before combining the realizations Z, the constant offset ξ, which is different for each skin cell, has to be compensated.

To simplify the combination and to avoid determining ξ for each of the nine sensors for all skin cells (in total 2277 offsets ξ for 253 skin cells), the *distribution of differences* rather than the distribution of compensated realizations were analyzed. Therefore, the difference Δ_t of consecutive samples is computed by

$$\Delta_t = X_t - X_{t-1} = Z_t - Z_{t-1} \tag{4.10}$$

such that the ξ of a sensor is compensated by the difference. Employing the properties of independent random variables, the expected value μ_Δ is then zero, and the variance $\sigma_\Delta^2 = \sigma_\mathrm{Z}^2 + \sigma_\mathrm{Z}^2 = 2\,\sigma_\mathrm{Z}^2$.

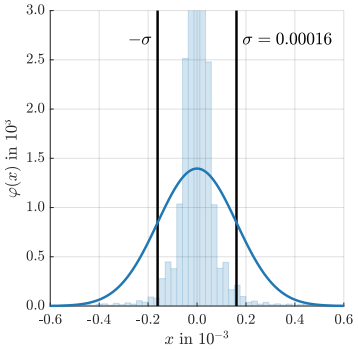
The resulting distributions of differences Δ for the skin cells' nine sensors are depicted in Figure 33. The determined standard deviations deliver first indications for event thresholds that minimize noise (Section 3.1.6). The following sections will refine these indications to optimal thresholds.



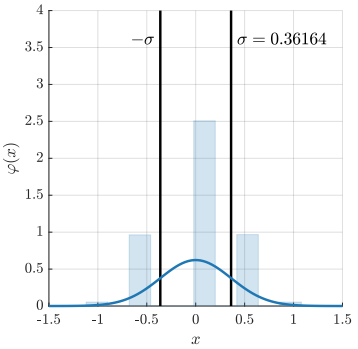
(a) Probability density function of the distribution of differences for the acceleration a_{x} along the x-axis.

(b) Probability density function of the distribution of differences for the acceleration a_{y} along the y-axis.

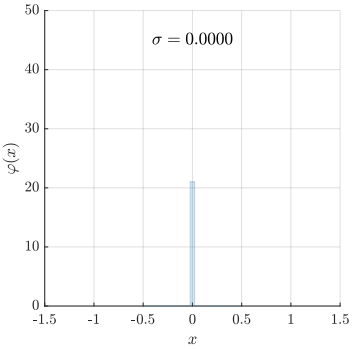
(c) Probability density function of the distribution of differences for the acceleration a_{z} along the z-axis.



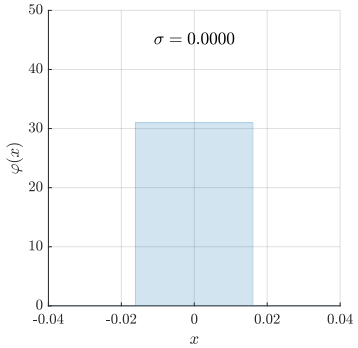
(d) Probability density function of the distribution of differences for the proximity sensor.



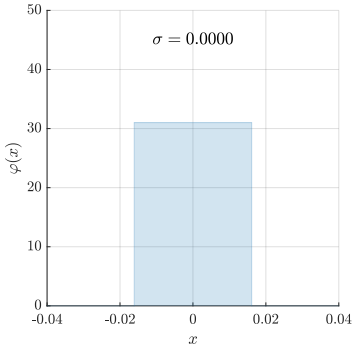
(e) Probability density function of the distribution of differences for the temperature sensor (low sensitivity).



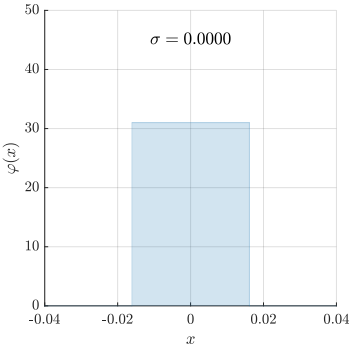
(f) Probability density function of the distribution of differences for the temperature sensor (high sensitivity).



(g) Probability density function of the distribution of differences for the force $F_{\mathrm{z},1}$ of sensor 1.



(h) Probability density function of the distribution of differences for the force $F_{\mathrm{z},2}$ of sensor 2.



(i) Probability density function of the distribution of differences for the force $F_{\mathrm{z},3}$ of sensor 3.

Figure 33 Distribution of differences Δ of the skin cells' nine sensors (the high-resolution temperature sensor is not deployed and reports a constant value) for 253 idle (resting) skin cells sampled at 62.5 Hz for 20 s. The accelerometer, the proximity sensor, and the temperature sensor have wider distributions, thus are affected more by noise. These sensors will require larger event thresholds δ than the force sensors. The values of the proximity sensor and the force sensors are normalized, that is, proximity and force measurements range from 0.0 to 1.0. Proximity sensor values represent *closeness*, where 1.0 equals to a distance of zero. The force values range from no force at 0.0 to the maximum at 1.0 (collapsing force cell, the insulated capacitor plates touch each other).

4.4.3. Stimulation Profiles – Transmission Rates and Encoding Errors

This section presents the results of the experimental evaluations that analyze the transmission rates and reconstruction errors of signals $x_{\mathrm{e}}(t)$ conveyed by events. Therefore, the evaluation determines the transmission rate reduction ratio $p_{f_{\mathrm{t}}}$ and the relative reconstruction error p_{RMSE}. The reduction ratio relates the transmission rate of the event-driven signal $x_{\mathrm{e}}(t)$ to the transmission rate of its clock-driven counterpart $x_{\mathrm{c}}(t)$, and the relative reconstruction error computes the error between these two signals. Both, the transmission rate and the error, depend on the sensor input signal $x(t)$. Therefore, the evaluation not only considers their dependency on different event thresholds δ, but also considers different stimulus profiles, see Table 7.

Stimulus	Description
Resting	The skin system is idle and no stimuli are occurring. The robot arm is not moving and the skin is not touched.
Slowly Moving	The skin system is not touched but the robot arm is slowly moving along a defined trajectory.
Swiftly Moving	The skin system is not touched but the robot arm is swiftly moving along the same trajectory as defined in the *Slowly Moving* experiment.
Hammering	Hammering with a fist on the base of the robot arm. The impacts' impulses and vibrations travel along the robot arm. The robot arm is holding a defined position and moves only because of the impacts.
Stroking	Stroking the robot arm with multiple hands continuously generates tactile changes throughout the whole surface of the skin. The robot arm is holding a defined position and does not move.
Pushing	Forcefully moving the robot arm by pushing it repeatedly with a fist. The robot arm moves into the pushed direction.
Sinusoidal Force Profile	The skin cell is placed in a force test stand similar to [102]. The skin cell is probed with a force of 3 N and a sinusoidal force profile with a frequency of 5 Hz or 30 Hz. The skin cell only moves because of the force impacts and vibrations.
Feather Teasing	Tickling a skin cell with a feather. The robot arm is holding a defined position and does not move.

Table 7 Different stimulus profiles with the focus to create changes in specific sensing modalities. Creating temperature changes requires complex experimental setups and is thus not considered. The experimental platform is not equipped with high sensitivity temperature sensors. Furthermore, the response of the standard temperature sensor is very slow ($f_{\mathrm{B}} < 1$ Hz). Thus, the event rate of this temperature is also very low.

The reduction ratios p_{f_t} of the sensors are computed by

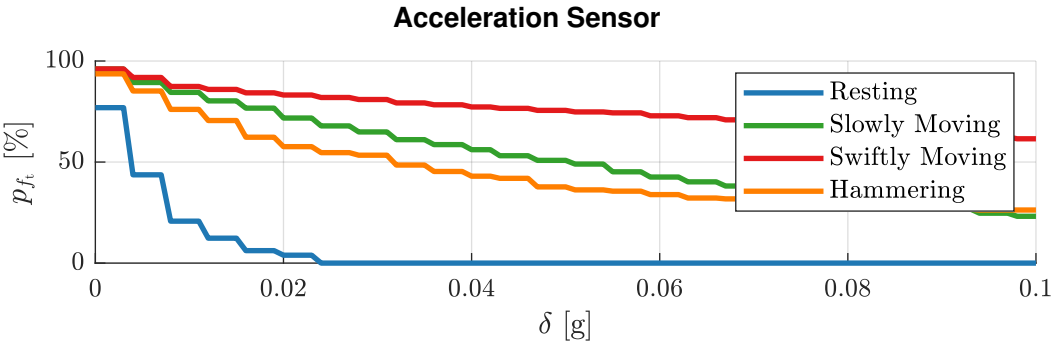
$$p_{f_t} = \frac{f_e}{f_s} \tag{4.11}$$

where f_e denotes the event rate of the sensor and f_s the sample rate of the clock-driven reference. The relative reconstruction error p_{RMSE} of a sensor signal encoded in events is computed by the relative Root Mean Square Error (RMSE) between the reconstructed signal $x_e(t)$ and the raw signal $x_c(t)$

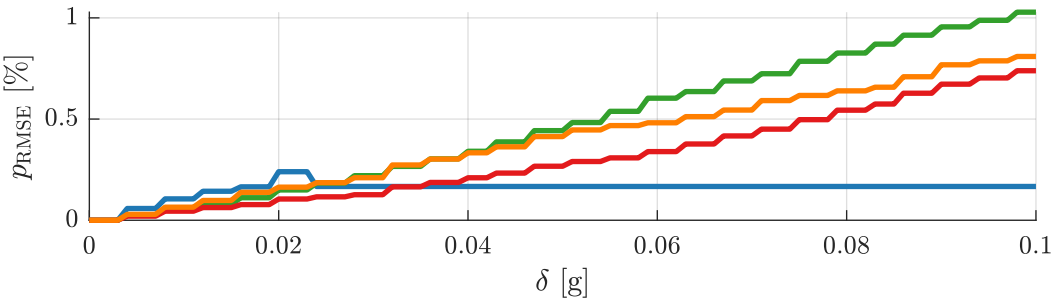
$$p_{RMSE}(x_e, x_c) = \frac{\sqrt{\frac{1}{n}\sum_{k=1}^{n}\left[x_e(t_k) - x_c(t_k)\right]^2}}{x_{max} - x_{min}} \tag{4.12}$$

where $k = 1, 2, \ldots, n$ are the sample indexes of the signals x_c and x_e. The signals have a range of $x_c, x_e \in [x_{min}, x_{max}]$.

Figures 34, 35, and 36 respectively depict the results for one selected acceleration, force, and proximity sensor of one representative skin cell. The Figure 35 for the force sensor also contains the results of a skin cell in the force test stand.

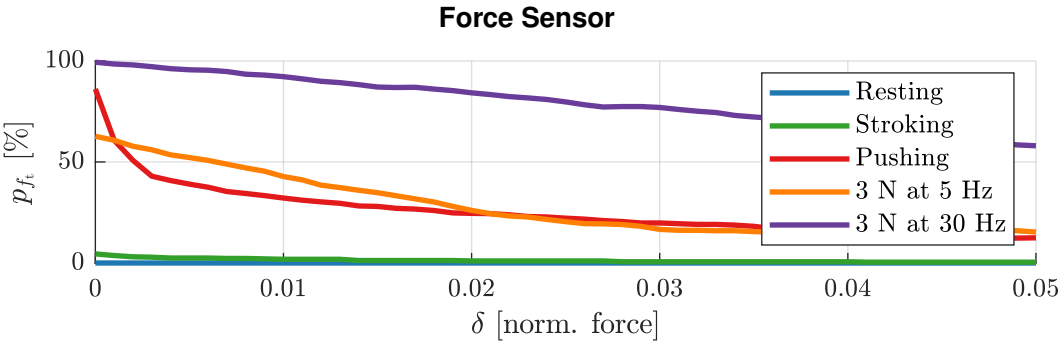


(a) The transmission rate reduction ratio p_{f_t} for one axis of the accelerometer.

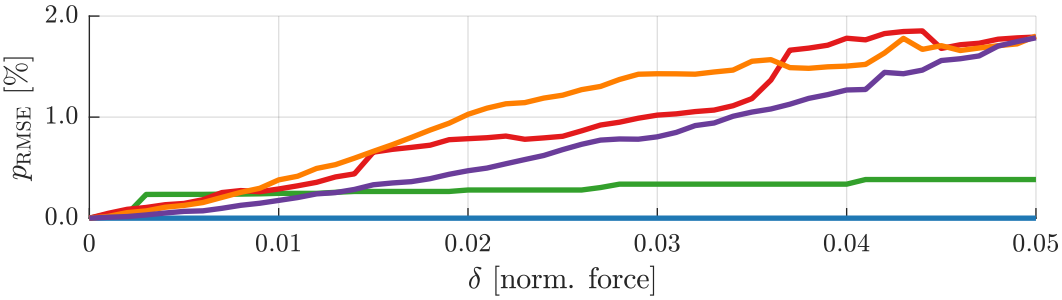


(b) The relative RMSE p_{RMSE} for one axis of the accelerometer.

Figure 34 The transmission rate reduction ratio and the relative error for one axis of the accelerometer of one representative skin cell. The sensor values are sampled at 62.5 Hz.

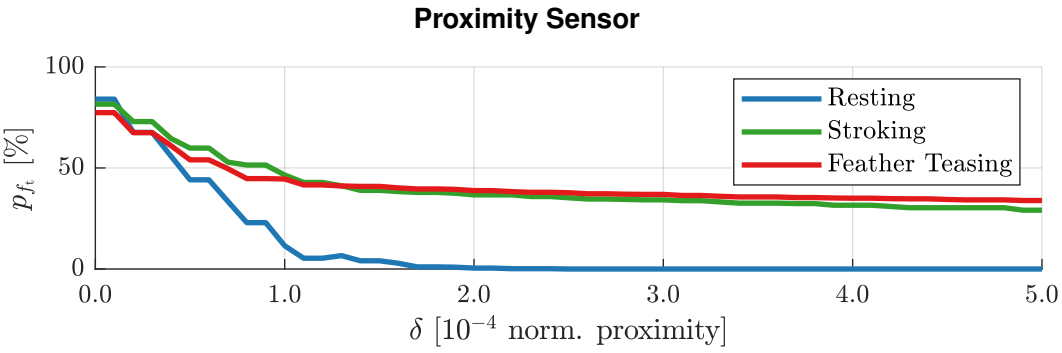


(a) The transmission rate reduction ratio p_{f_t} for one force sensor.

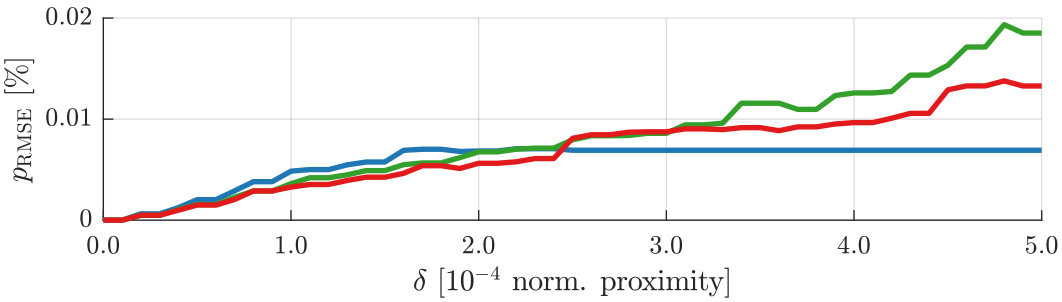


(b) The relative RMSE p_{RMSE} for one force sensor.

Figure 35 The transmission rate reduction ratio and the relative error for one force sensor of one representative skin cell. The sensor values are sampled at 62.5 Hz. The plot additionally contains the results for a skin cell in the force test stand that generates sinusoidal force profiles. The force sensor values are normalized to the range 0.0 to 1.0 where 1.0 represents maximum force.



(a) The transmission rate reduction ratio p_{f_t} for the proximity sensor.



(b) The relative RMSE p_{RMSE} for the proximity sensor.

Figure 36 The transmission rate reduction ratio and the relative error for the proximity sensor of one representative skin cell. The sensor values are sampled at 62.5 Hz. The plot additionally contains the results for a skin cell tickled with a feather. The proximity sensor values are normalized to the range 0.0 to 1.0 where 1.0 represents zero distance. The proximity values represent closeness.

4.4.4. Event Threshold Tuning

This section takes the results of Sections 4.4.2 and 4.4.3 and follows the guidelines of Section 3.1.7 to determine the event thresholds δ for each sensor modality such that these thresholds achieve a good compromise between a low idle event rate (noise), a small encoding error, and a high reduction ratio for transmission rates.

Figures 34, 35, and 36 show that an event threshold above the standard deviation of the noise (see Figure 33) indeed significantly reduces the idle event rate of the resting system (Table 7). Such a threshold also reduces the event rates of more intense stimulus profiles while the encoding error stays low (below 0.5 %). Larger event thresholds further reduce event rates but at the cost of a higher encoding error and reduced sensitivity. Thus, the best trade-off for the event thresholds of the investigated e-skin system lies around the standard deviation of the noise. Table 8 depicts the selected event thresholds.

	Acceleration	**Force**	**Proximity**	**Temperature**
δ	0.02	0.001	0.0001	0.5

Table 8 Selected event thresholds for the different modalities of the e-skin. The acceleration is in g, the force, and proximity thresholds are normalized (they range from 0.0 to 1.0), and the temperature is in °C.

4.4.5. Stimulation Profiles – Activity Ratios and Event Rates

This section evaluates the activity ratios of the skin cells' sensors, the activity of the skin cells, and the number of active sensors within a skin cell for different stimulation profiles (Table 7). These activity ratios compare the event rate in the event-driven mode with the sample rates in the clock-driven mode. The ratios and the event activity patterns within a skin cell provide insights into the reduction of transmission rates of the event-driven e-skin and the number of events that occur at the same time.

The stimulations used for this evaluation (slowly moving, swiftly moving, hammering, and stroking) target the skin cell's different sensing modalities with the objective to uniformly stimulate as many skin cells as possible. The stimulation profiles *slowly moving* to *hammering* mostly target the acceleration sensors while *stroking* targets the force and proximity sensors.

The raw sensor values are captured and processed according to Section 4.4.3. The off-line event generators use the event thresholds of Table 8.

A sensor of a skin cell is *active* when it provides a sample or an event. In clock-driven mode, a sensor l is continuously active for each sample and all the sensors together accumulate $A_{\mathrm{s,c}}$ activity states at time t

$$A_{\mathrm{c},l} = n_{\mathrm{sc}}\, f_{\mathrm{s}}\, t \tag{4.13}$$

where n_{sc} is the number of skin cells, and f_{s} the sampling time. In the event-driven mode, the sensors accumulate $A_{\mathrm{e},l}$ activity states

$$A_{\mathrm{e},l} = \sum_{i=1}^{n_{\mathrm{sc}}} \sum_{k} \alpha_{i,l}(t_k) \tag{4.14}$$

with the sensor activity $\alpha_{i,l}$

$$\alpha_{i,l}(t_k) = \begin{cases} 1 & \textbf{if event } \mathbf{e} \textbf{ on sensor } l \textbf{ of skin cell } i \textbf{ at } t_k \\ 0 & \textbf{otherwise} \end{cases}. \tag{4.15}$$

The activity ratio $p_{\mathrm{A},l}$ for a sensor l of an e-skin with n_{sc} skin cells with respect to the clock-driven reference is then

$$p_{\mathrm{A},l} = \frac{A_{\mathrm{e},l}}{A_{\mathrm{c},l}} = \frac{\sum_{i=1}^{n_{\mathrm{sc}}} \sum_{k} \alpha_{i,l}(t_k)}{n_{\mathrm{sc}}\, f_{\mathrm{s}}\, t}. \tag{4.16}$$

The results of these activity ratios $p_{\mathrm{A},l}$ for the skin cells' nine sensors are depicted in Table 9.

Sensor	Resting	Slowly Moving	Swiftly Moving	Hammering	Stroking
Acceleration x	9.90 %	46.6 %	68.3 %	49.3 %	13.1 %
Acceleration y	9.27 %	43.2 %	65.2 %	48.2 %	13.0 %
Acceleration z	9.29 %	45.9 %	68.2 %	48.6 %	12.0 %
Proximity	6.51 %	6.98 %	7.54 %	9.73 %	14.6 %
Temperature 1	3.17 %	3.37 %	4.57 %	3.74 %	2.45 %
Temperature 2	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
Force 1	0.00 %	0.00 %	0.00 %	0.00 %	0.517 %
Force 2	0.00 %	0.00 %	0.00 %	0.00 %	0.518 %
Force 3	0.00 %	0.00 %	0.00 %	0.00 %	0.403 %

Table 9 Activity ratios of the nine sensors of 253 skin cells. In event-driven mode, the sensors are rarely all active at the same time. The activities in the different modalities heavily depend on the stimulus.

A skin cell is active when at least one of its sensors is active. In the clock-driven mode a skin cell is active for each sample. The accumulated skin cell activity states at time t are thus

$$A_{\mathrm{c,sc}} = n_{\mathrm{sc}}\, f_{\mathrm{s}}\, t. \tag{4.17}$$

In the event-driven mode, the accumulated skin cell activity states are

$$A_{\mathrm{e,sc}} = \sum_{i=1}^{n_{\mathrm{sc}}} \sum_{k} \alpha_{i}(t_k) \tag{4.18}$$

with the skin cell activity α_i

$$\alpha_i(t_k) = \begin{cases} 1 & \textbf{if event } \mathbf{e} \textbf{ on skin cell } i \textbf{ at } t_k \\ 0 & \textbf{otherwise} \end{cases}. \tag{4.19}$$

The activity ratio p_A for n_sc skin cells with respect to the clock-driven reference is then

$$p_\mathrm{A} = \frac{A_\mathrm{e,sc}}{A_\mathrm{c,sc}} = \frac{\sum_{i=1}^{n_\mathrm{sc}} \sum_k \alpha_i(t_k)}{n_\mathrm{sc}\, f_\mathrm{s}\, t}. \tag{4.20}$$

These activity ratios are depicted in Table 10.

	Resting	**Slowly Moving**	**Swiftly Moving**	**Hammering**	**Stroking**
Active Cells	**32.9 %**	80.0 %	**90.1 %**	77.2 %	43.9 %

Table 10 Activity ratios of 253 skin cells. If all events fit into one event packet, then the event-driven mode is expected to reduce the transmission rates by at least 9.9 % (worst case) and at most by 67.1 %.

To compute the ratio of a specific number n_as of active sensors per skin cell, first their accumulated activities are determined by

$$A_\mathrm{e,sc}(n_\mathrm{as}) = \sum_{i=1}^{n_\mathrm{sc}} \sum_k \alpha_i(t_k, n_\mathrm{as}) \tag{4.21}$$

with the skin cell activity α_i for n_as active sensors (events at time t_k)

$$\alpha_i(t_k, n_\mathrm{as}) = \begin{cases} 1 & \textbf{if events } \mathbf{e} \textbf{ for } n_\mathrm{as} \textbf{ sensors on skin cell } i \textbf{ at } t_k \\ 0 & \textbf{otherwise} \end{cases}. \tag{4.22}$$

Then, the ratio for skin cells with n_as active sensors is

$$p_\mathrm{A}(n_\mathrm{as}) = \frac{A_\mathrm{e,sc}}{A_\mathrm{c,sc}} = \frac{\sum_{i=1}^{n_\mathrm{sc}} \sum_k \alpha_i(t_k, n_\mathrm{as})}{n_\mathrm{sc}\, f_\mathrm{s}\, t} \tag{4.23}$$

and thus the sum of all of these ratios $p_\mathrm{A}(n_\mathrm{as})$ is

$$p_\mathrm{A} = \sum_{l=1}^{9} p_\mathrm{A}(l). \tag{4.24}$$

Table 11 summarizes the results for these ratios.

n_{as}	Resting	Slowly Moving	Swiftly Moving	Hammering	Stroking
1	28.0 %	33.0 %	16.6 %	25.0 %	33.4 %
2	4.52 %	29.5 %	28.7 %	25.4 %	8.64 %
3	0.350 %	15.9 %	39.6 %	23.5 %	1.38 %
4	0.0131 %	1.57 %	5.08 %	3.16 %	0.283 %
5	0.00 %	0.0285 %	0.157 %	0.101 %	0.0865 %
6	0.00 %	0.00 %	0.00 %	0.00 %	0.0286 %
7	0.00 %	0.00 %	0.00 %	0.00 %	0.00159 %
8	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
9	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
Active Cells	**32.9 %**	80.0 %	**90.1 %**	77.2 %	43.9 %

Table 11 The activity ratios for a specific number sensors that are active at the same time.

In summary, the off-line evaluation shows that when all events fit into one event packet, then the event-driven mode is expected to reduce the transmission rates by at least 9.9 % (worst case) and at most by 67.1 % (best case). Furthermore, the activity ratios for a specific number of sensors that are active at the same time indicate that rarely all sensors of a skin cell generate events at the same time. Thus, smaller event packets might, on average, improve the reduction of transmission rates by reducing overhead.

4.4.6. Optimal Event Packet Size

This section analyzes the results presented in Section 4.4.5 to determine the optimal event packet size. The packet size in the skin cell network is fixed (Section 4.3.2.1) and cannot change dynamically to fit to the number of events to transmit. Therefore, the optimal packet size is a trade-off that, on average, minimizes the overhead. Overhead occurs when an event packet contains fewer events than it could transport, or when multiple event packets have to be sent because several events occurred at the same time and did not fit into one event packet. Thus, to minimize overhead, the event packet must neither be too small nor too big.

Assuming that an event packet can fit in worst case all the events of the skin cell's sensors, then the ratio between the event packet rate and the data packet rate of the clock-driven system will match with the ratios of the active skin cells of Table 10. The improvement would be at least 9.9 % (worst case) and at most by 67.1 % (best case). An optimal event packet size will improve these ratios since events only rarely occur for all sensors at the same time, see Table 11.

In the following, we investigate the reduction ratios for different packet sizes. Despite this section's focus on optimizing the event packet size for skin cells with nine sensors, the presented approach is not limited to these specific skin cells. An e-skin system with fewer or more than nine sensors per skin cell requires the analysis of the activation of its sensors that would

lead to results similar to ones presented in Table 10. Then, from this point on, the following analysis of this section only differs in the packet size and the number of sensors.

Table 12 presents the size of event packets in bytes that fit up to nine events into one packet.

n_{e}	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
$s_{\mathrm{p,e}}(n_{\mathrm{e}})$	7	9	10	12	14	15	17	18	20
$s_{\mathrm{p,e,ts}}(n_{\mathrm{e}})$	10	12	13	15	17	18	20	21	22

Table 12 Event packet sizes in bytes for fitting up to nine events into one packet. $s_{\mathrm{p,e}}$ is the packet size in bytes without timestamps and $s_{\mathrm{p,e,ts}}$ with timestamps. n_{e} denotes the maximum number of events that fit into one packet. Timestamps are assumed to require 21 bit.

The table additionally presents the packet sizes in bytes for event packets containing 21 bit time stamps.

The reduction ratio of the transmission rate $p_{f_{\mathrm{t}}}$ computes then by

$$p_{f_{\mathrm{t}}} = \frac{n_{\mathrm{p,e}}(n_{\mathrm{e}})}{n_{\mathrm{p,d}}} \frac{s_{\mathrm{p,e}}(n_{\mathrm{e}})}{s_{\mathrm{p,d}}} \tag{4.25}$$

where $n_{\mathrm{p,e}}$ is the number of event packets, $n_{\mathrm{p,d}}$ the number of data packets in clock-driven mode, $s_{\mathrm{p,e}}$ the size of the event packets, and $s_{\mathrm{p,d}}$ the size of the data packets.

The number of event packets $n_{\mathrm{p,e}}$ depends on the number of events n_{e} that an event packet can fit and the number of events that occur at the same time, and can be computed by

$$n_{\mathrm{p,e}}(n_{\mathrm{e}}) = \sum_{l=1}^{9} A_{\mathrm{e,sc}}(l) \left\lceil \frac{l}{n_{\mathrm{e}}} \right\rceil . \tag{4.26}$$

$A_{\mathrm{e,sc}}(l)$ defined by Equation (4.21) is the accumulated activity of skin cells that have l sensors generating events at the same time and is multiplied by the number of event packets that are required to contain these l events. The sum of all evaluation for one to nine events occurring at the same time represent the number of event packets.

The results of the reduction ratio $p_{f_{\mathrm{t}}}$ are depicted in Table 13 and Figure 37.

	Resting	Slowly Moving	**Swiftly Moving**	Hammering	Stroking
Without Time Stamps					
1 event/packet	13.35 %	51.13 %	74.84 %	55.86 %	19.80 %
2 events/packet	14.96 %	43.87 %	60.79 %	46.85 %	20.59 %
3 events/packet	**16.45 %**	40.78 %	**47.69 %**	40.24 %	22.13 %
4 events/packet	19.73 %	48.00 %	54.18 %	46.39 %	26.38 %
5 events/packet	23.02 %	55.98 %	63.09 %	54.05 %	30.89 %
6 events/packet	24.66 %	59.98 %	67.61 %	57.91 %	32.89 %
7 events/packet	27.95 %	67.98 %	76.62 %	65.63 %	37.28 %
8 events/packet	29.59 %	71.97 %	81.13 %	69.49 %	39.47 %
9 events/packet	32.88 %	79.97 %	90.14 %	77.21 %	43.86 %
With Time Stamps					
1 event/packet	19.07 %	73.04 %	106.9 %	79.80 %	28.30 %
2 events/packet	19.95 %	58.49 %	81.05 %	62.47 %	27.45 %
3 events/packet	**21.38 %**	53.02 %	**62.00 %**	52.31 %	28.77 %
4 events/packet	24.66 %	60.00 %	67.72 %	57.98 %	32.98 %
5 events/packet	27.95 %	67.98 %	76.62 %	65.63 %	37.30 %
6 events/packet	29.59 %	71.97 %	81.13 %	69.49 %	39.47 %
7 events/packet	32.88 %	79.97 %	90.14 %	77.21 %	43.86 %
8 events/packet	34.52 %	83.97 %	94.65 %	81.07 %	46.05 %
9 events/packet	37.81 %	91.97 %	103.7 %	88.79 %	50.43 %

Table 13 Transmission rate ratios for different event packet sizes with respect to the clock-driven reference. The ratios are lowest for packets that fit three events.

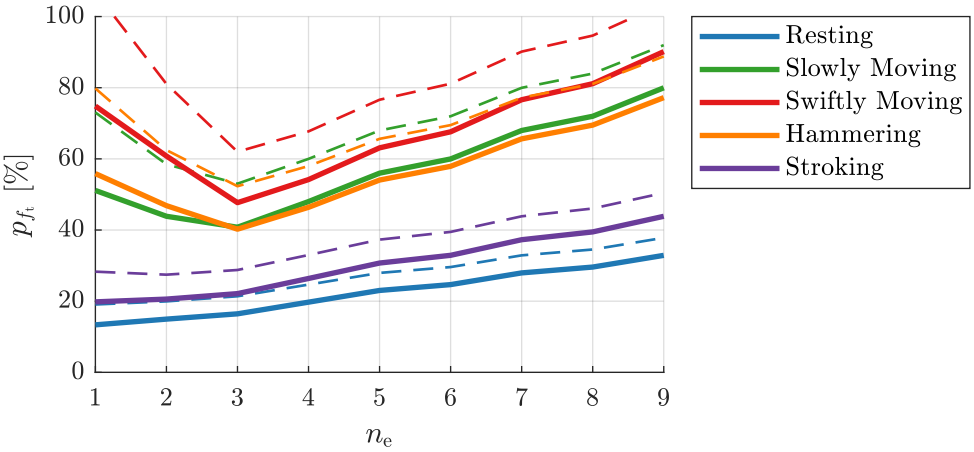


Figure 37 Transmission rate ratios versus the number of events that fit into one packet. The solid lines refer to event packets without timestamps and the dashed lines refer to packets with timestamps.

An event packet with an optimal size fits three events. That result is reasonable since activity on the three axes of the accelerometer or the three force sensors is highly correlated (Table 11). The optimal event packet size improves the reduction ratio in general. The ratio improves for a heavily stimulated system from 90.1 % to 47.69 % or respectively to 62.0 % when timestamps are employed. When the system rests, the optimal packet size improves the ratio from 32.9 % to 16.45 % or respectively to 21.38 %.

4.5. Effectiveness of Event-Driven Skin Systems

This section presents the evaluation of the fully implemented event-driven approach, that this thesis introduces for efficient Large-Area Skin Systems (LASSs). The implementation of the event-driven approach in the modular e-skin system followed the descriptions of Sections 4.3 and 4.3.3. Thus, the e-skin system to evaluate supports the event-driven and the clock-driven operation modes, which eases the analysis since one e-skin system can provide the experimental data for the event-driven approach and the clock-driven reference. The evaluation compares the transmission rate of the event-driven e-skin and its demand for computational power with their clock-driven references. These comparisons deliver reduction ratios for the event-driven system in comparison to the clock-driven reference. The analysis of these ratios then allows for assessing the performance gain of the event-driven e-skin, which delivers the empirical proof for the efficiency of this thesis' approach in real e-skin systems. The subsequent validation of the CPU usage model introduced in Section 3.3.2 with the measurements of the real e-skin delivers the basis for extrapolating the performance assessment of the event-driven approach. These extrapolations, combined with the performance evaluation of the implemented event-driven e-skin system, provide the empirical proof that this thesis' approach renders LASSs possible and that it delivers significant improvements for efficiency. Both results indicate the scalability of information handling and the effectiveness of tactile large-area feedback for event-driven LASSs.

This section presents the evaluation as follows. First, Section 4.5.1 introduces the experimental setup. Then, Section 4.5.2 presents the reduction ratios of the e-skin's transmission rate. Afterward, Section 4.5.3 analyzes the relation between transmission rates and CPU usage and connects it to the CPU usage model to extrapolate the demand for computational power in Section 4.5.5. Furthermore, Section 4.5.4 analyses the experimental data and extrapolates the event packet rates for larger e-skin systems. Together, Sections 4.5.4 and 4.5.5 deliver a first comprehensive impression on the scalability of the event-driven approach towards LASSs. This impression delivers empiric support that LASSs, as realized in Section 4.6, are feasible.

4.5.1. Experimental Setup and Protocol – Fully Integrated Event Generation

This section uses the same experimental platform as presented in Section 4.4. However, in contrast to Section 4.4, the event-driven approach is now fully integrated into the e-skin's 253 skin cells, and all experimental data is collected on-line. All performed experiments are introduced in Table 14.

The work presented in Section 4.5 was in part published in:
Bergner, F., Dean-Leon, E., Cheng, G., "Event-based signaling for large-scale artificial robotic skin - Realization and performance evaluation". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, Korea, 2016, pp. 4918–4924.

Experiment	Description
Idle	The robot is not moving and the skin is not stimulated.
Stroking Upper Arm	The upper arm skin of the robot is stroked by one person. The robot arm is inactive and not moving.
Stroking Arm	The upper and lower arm skin of the robot is stroked by two persons. The robot arm is inactive and not moving.
Hammering	The skin is not touched but one person hammers on the end effector of the robot arm. The robot arm is inactive and not moving.
Moving Arm	The skin is not touched. The robot arm is active and moving.
Reactive Control	The robot arm is controlled by the skin. Touching the skin results in arm movements.

Table 14 Overview and description of experiments.

In addition to the experiments that target to stimulate the different tactile modalities of the e-skin similar to Section 4.4, the *reactive control* experiment utilizes the e-skin system's tactile feedback in real-time to realize a contact avoidance (kinesthetic) controller [45]. Tactile interaction controllers will be explained and validated in Chapter 5. Here, the reactive control is solely a tool to additionally evaluate the e-skin system for interactions that generate tactile and motion stimuli.

The listed experiments generate stimuli that are representative and realistic in physical interactions and enclose the best and the worst case. Each experiment was performed for around 10 s with different sample rates in clock-driven and event-driven operation modes. During the execution of the experiments, the experimenters applied the stimuli as uniformly and continuously as possible. Thus, the collected data can be averaged in the evaluation. The event generators of the skin cells utilize the optimized event thresholds of Table 8 (Section 4.4.4).

4.5.2. Event Rates and Reduction Ratio

Table 15 presents the average event rates per skin cell for the different modalities. Its last column presents the average event packet rate per skin cell. Each entry of the table contains also the ratio of the respective rate in the event-driven mode compared to their baselines in the clock-driven mode. A ratio of 100 % describes that the respective rate in event-driven mode equals the rate in clock-driven mode. Then, the transmission rate reduction of the event-driven system is zero.

Experiment	Acceleration	Force	Proximity	Temperature	Packet Rate
Idle	1.23 (0.33 %)	0.21 (0.17 %)	4.17 (3.33 %)	0.15 (0.12 %)	5.71 (4.57 %)
Stroking Upper Arm	9.33 (2.49 %)	1.95 (1.56 %)	17.9 (14.3 %)	0.20 (0.16 %)	25.2 (20.1 %)
Stroking Arm	14.6 (3.90 %)	3.09 (2.47 %)	26.2 (21.0 %)	0.20 (0.16 %)	37.3 (29.9 %)
Hammering	137 (36.6 %)	0.57 (0.46 %)	5.69 (4.55 %)	0.29 (0.23 %)	89.6 (71.7 %)
Moving Arm	174 (46.6 %)	1.85 (1.49 %)	4.65 (3.72 %)	0.36 (0.29 %)	95.8 (76.6 %)
Reactive Control	86 (22.9 %)	3.66 (2.93 %)	8.60 (6.88 %)	0.27 (0.22 %)	64.7 (51.8 %)

Table 15 The average event rate per skin cell and the ratio between the event rate and the baseline activity related to the sampling frequency in the clock-driven operation mode (in brackets). The baseline is, for example, three times the sampling frequency for acceleration and force sensors. The packet rate is the average event packet rate per skin cell per second and the ratio describes the ratio of the event packet rate with respect to the data packet rate in clock-driven mode. The sample rate of the sensors is 125 Hz. The colored fields highlight the dominant modalities of an experiment.

For all experiments, the event-driven e-skin shows good reductions for packet rates. In the best case (idle) the ratio is 4.57 %, that is the Event-Driven System (EDS) reduces the packet rate by 95.43 %. In the worst case (moving arm), the reduction is still 76.6 %. The representative tactile interaction example (reactive control) shows a good ratio of 51.8 %, which implies a reduction of the packet rate by 48.2 %.

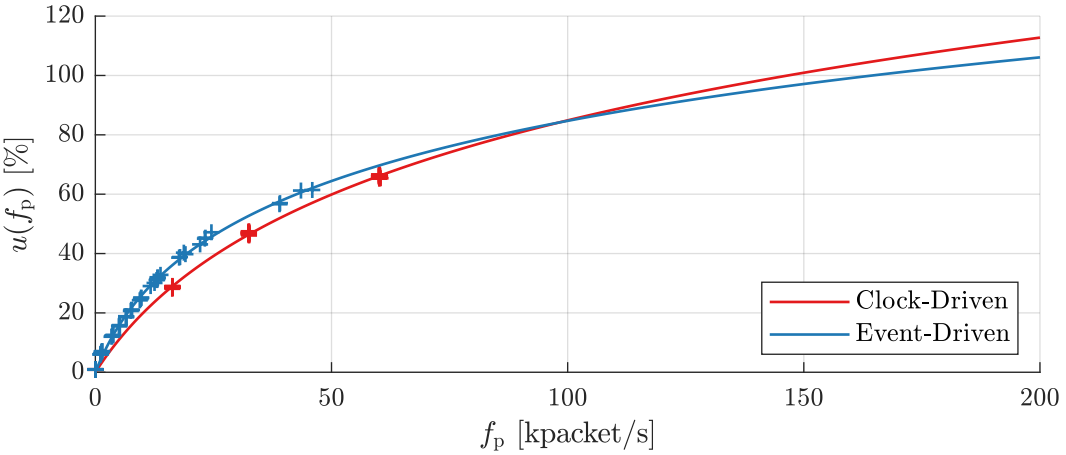
4.5.3. CPU Usage Model

This section investigates the relation between two performance indicators (Section 4.3.5) that are the packet rate and the CPU usage to unpack these packets. The CPU usage is expected to increase for higher packet rates (Section 3.3.1). Furthermore, identical packet rates in clock- or event-driven mode are expected to cause approximately the same CPU usage (Section 4.2.2.2).

Plotting the CPU usage u of all the experiments against the packet rate f_p, Figure 38a reveals that the relationship is approximately logarithmic

$$u(f_\mathrm{p}) = a \cdot \log(b \cdot f_\mathrm{p} + 1) \tag{4.27}$$

where the parameters a and b can be determined via curve fitting. The table in Figure 38b summarizes these parameters for data and event packets.



(a) The CPU usage models and measurements against the packet rate f_p. The solid lines represent the models, and the crosses the measurements. The relation between CPU usage and packet rate is approximately identical for both operation modes with a small advantage for the event-driven mode for higher packet rates.

Mode	a	b	R^2
Clock-Driven	45.52	$5.453 \cdot 10^{-5}$	0.9993
Event-Driven	32.65	$1.239 \cdot 10^{-4}$	0.9969

(b) The model parameters for both operation modes. The last column contains the fitting accuracies of the parameters.

Figure 38 The CPU usage versus the packet rate and the model parameters. The CPU usage here denotes the CPU usage of one of the eight cores of an Intel Core i7 4770 CPU. The data points are acquired by conducting the experiments listed in Table 14.

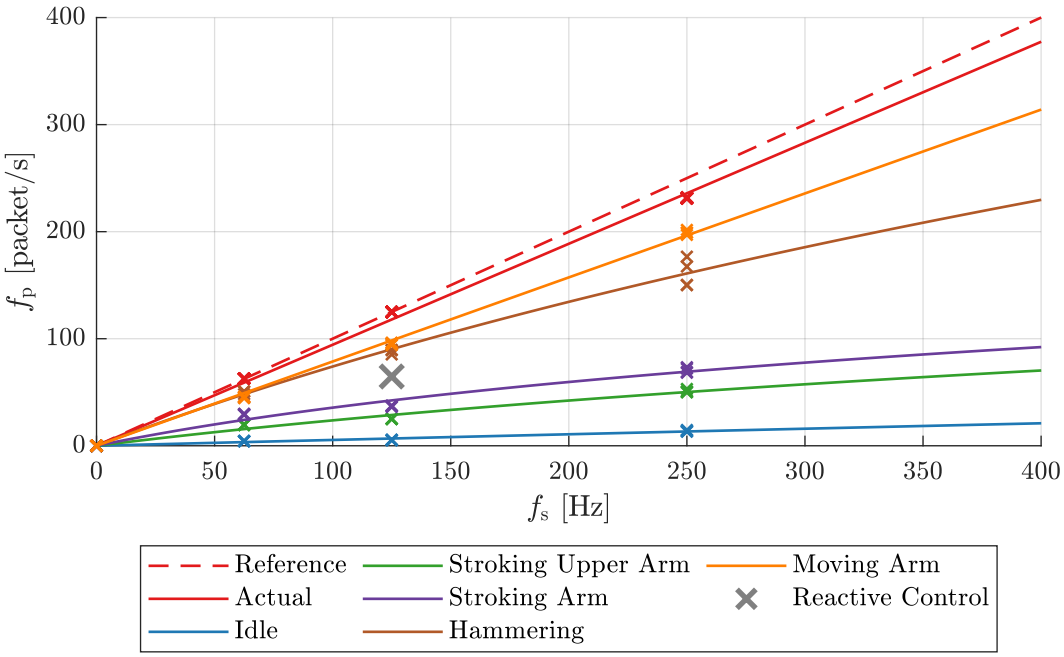
The logarithmic relationship indicates that handling data and event packets is more efficient for higher packet rates. Section 3.3.2 introduced a comprehensive CPU model. This model indeed explains that the overhead of resuming the packet handling in the skin driver process reduces for higher packet rates. The comprehensive CPU model and the logarithmic approximation agree well, where the logarithmic model better describes the relation for high CPU usages. Overall, the logarithmic CPU model has high fitting accuracy and is a simple mathematic function. Therefore, the model poses an ideal basis for the CPU usage extrapolation of Section 4.5.5.

4.5.4. Event Packet Rate Extrapolation

This section extrapolates the event packet rate for larger e-skin systems. Therefore, it investigates the relationship between the event packet rate f_p, the experiment, and the sample rate f_s. The relationship between stimulation intensity, sample rate, and the resulting event packet rate is expected to be better than linear. In the worst case, the event packet rate equals the sample rate (or respectively the data packet rate in clock-driven mode). In this case, the performance of the event-driven e-skin falls back to the performance of clock-driven e-skin.

Figure 39a depicts the packet rate f_p of the experiments against the sample rate f_s. Experiments with a higher stimulation intensity, such as the moving arm experiment, have a high

packet rate close to the linear boundary. Experiments with lower stimulation intensities have low packet rates, and the relationship is better than linear.



(a) The packet rate models and measurements against the sample rate f_s. The solid lines represent the models, and the crosses the measurements.

Experiment / Model	**Mode**	a	b	R^2
Reference	Clock-Driven	1.0	–	–
Actual	Clock-Driven	0.9435	–	0.9971
Idle	Event-Driven	243.8846	$2.248 \cdot 10^{-4}$	0.9651
Stroking Upper Arm	Event-Driven	85.0385	$3.215 \cdot 10^{-3}$	0.9764
Stroking Arm	Event-Driven	74.2308	$6.154 \cdot 10^{-3}$	0.9732
Hammering	Event-Driven	330.2692	$2.512 \cdot 10^{-3}$	0.9888
Moving Arm	Event-Driven	$1.214 \cdot 10^{7}$	$1.687 \cdot 10^{-5}$	0.9983

(b) The model parameters for the clock- and event-driven operation modes. The last column contains the fitting accuracies of the parameters.

Figure 39 The packet rate per skin cell in packets per second against the sample rate of the skin cell's sensors. The *reference* and the *actual* packet rates describe the data packets rates of the system in clock-driven mode. The other packet rates are event packet rates of the corresponding experiments, e.g. *reactive control*. The data packet rate of the system in clock-driven mode is not dependent on the experiment.

The model for data packet rates in clock-driven operation mode is linear

$$f_{\mathrm{p,c}}(f_\mathrm{s}) = a \cdot n_\mathrm{sc} \cdot f_\mathrm{s} \tag{4.28}$$

where n_{sc} denotes the number of skin cells, and a a scaling factor. For the ideal reference, a is one. In the real system, a compensates the difference between expected and actual sampling frequency.

The model for event packet rates is approximately logarithmic

$$f_{\mathrm{p,e}}(f_{\mathrm{s}}) = a \cdot n_{\mathrm{sc}} \cdot \log(b \cdot f_{\mathrm{s}} + 1) \tag{4.29}$$

where the parameters a and b are different for each experiment. All parameters are determined via curve fitting and are listed in the table of Figure 39b.

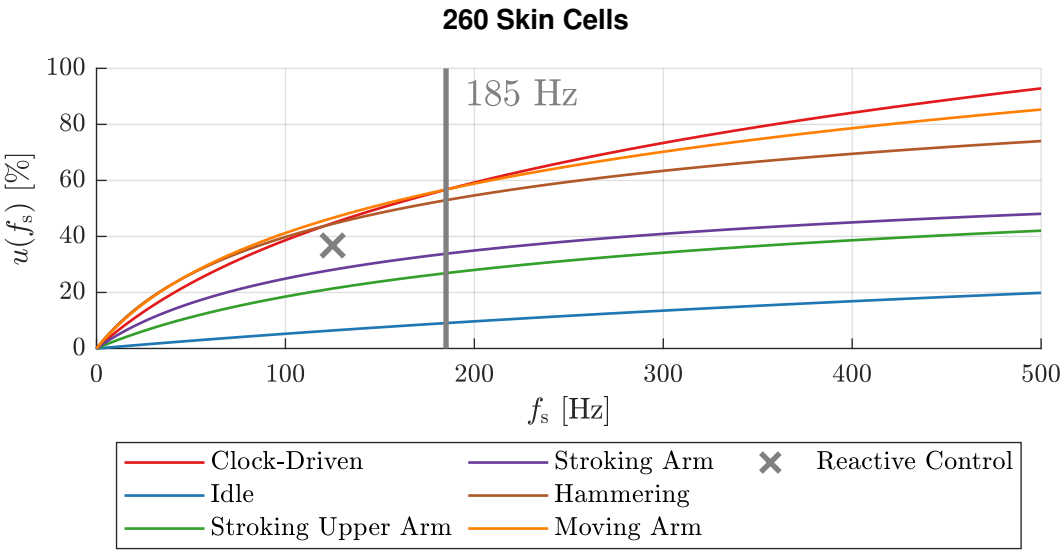
The results indicate that the performance edge of the event-driven e-skin increases for higher sample rates.

4.5.5. CPU Usage Extrapolation

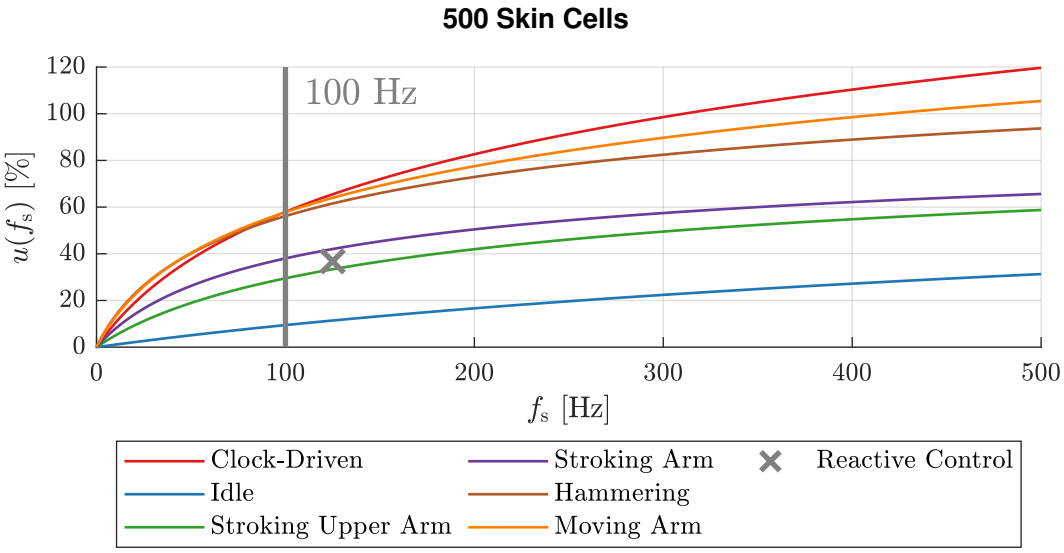
This section combines the results of Sections 4.5.3 and 4.5.4 to extrapolate the CPU usage for larger skin systems. Therefore, we combine the CPU model with the model for extrapolating packet rates

$$u(f_{\mathrm{s}}, n_{\mathrm{sc}}) = u(f_{\mathrm{p}}(f_{\mathrm{s}}, n_{\mathrm{sc}})). \tag{4.30}$$

First, we fix the number of skin cells n_{sc} to 260 and 500 skin cells, and plot the extrapolated CPU usage u against the sample rate f_{s}. Then, we fix the sample rate f_{s} to 62.5 Hz and 250 Hz. The results are depicted in Figures 40 and 41.

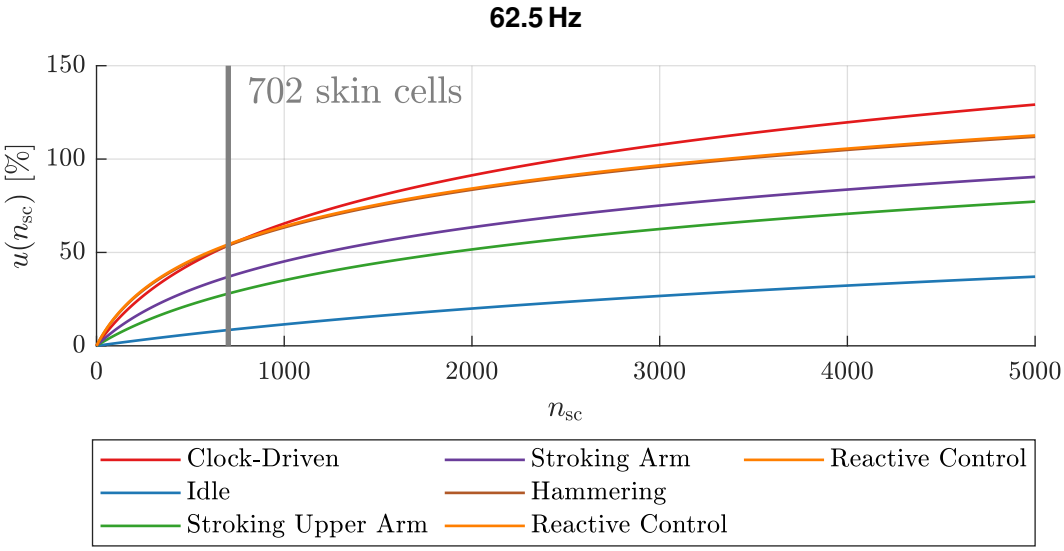


(a) The CPU usage extrapolation for 260 skin cells against the sample rate f_s. In the worst case, the event-driven e-skin outperforms the clock-driven one for sample rates greater than 185 Hz.

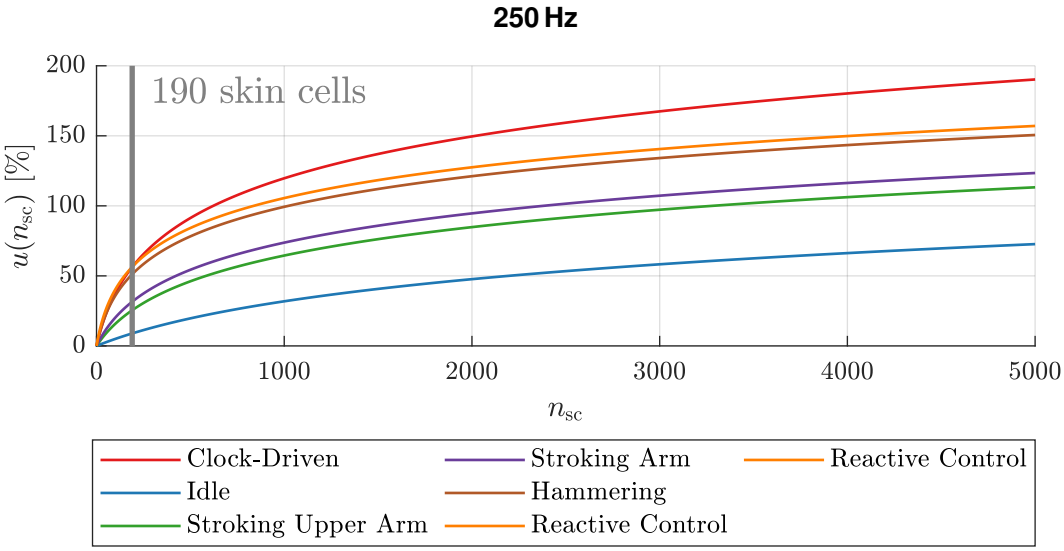


(b) The CPU usage extrapolation for 500 skin cells against the sample rate f_s. In the worst case, the event-driven e-skin outperforms the clock-driven one for sample rates greater than 100 Hz.

Figure 40 The CPU usage extrapolation against the sample rate. The gray bars are the worst-case bounds for the sample rate, after which the event-driven e-skin is better than its clock-driven reference.



(a) The CPU usage extrapolation for a sample rate of 62.5 Hz against the number of skin cells n_{sc}. In the worst case, the event-driven e-skin outperforms the clock-driven one for more than 702 skin cells.



(b) The CPU usage extrapolation for a sample rate of 250 Hz against the number of skin cells n_{sc}. In the worst case, the event-driven e-skin outperforms the clock-driven one for more than 190 skin cells.

Figure 41 The CPU usage extrapolation against the number of skin cells. The gray bars are the worst-case bounds for the number of skin cells, after which the event-driven e-skin is better than its clock-driven reference.

The extrapolation results show that the event-driven e-skin outperforms the clock-driven reference for higher sample rates and more skin cells. The gray bars in the figures indicate the lower bounds for the number of skin cells and the sample rate. As long as the number of skin cells and the sample rate stay above these bounds, the event-driven e-skin shows superior performance, even in the worst case. Nevertheless, in most cases, the event-driven e-skin outperforms the clock-driven e-skin well below these lower bounds. In the worst-case scenario, with 260 skin cells at 250 Hz, the event-driven e-skin shows a reduction of the packet rate by 21.2 % and of the CPU usage by 2.72 %. For the higher number of 5000 skin cells,

these reductions improve to 21.2 % and 17.46 % respectively. Table 16 lists some ratios for the extrapolated worst-case performance indicators.

Mode	Number of Skin Cells	Sample Rate	Packet Rate Ratio	CPU Usage Ratio
Clock-Driven	–	–	100%	100%
Event-Driven	500	62.5 Hz	79.4%	104%
Event-Driven	500	100 Hz	78.7%	100%
Event-Driven	500	500 Hz	78.4%	88.1%
Event-Driven	100	250 Hz	78.8%	107%
Event-Driven	190	250 Hz	78.8%	100%
Event-Driven	260	250 Hz	78.8%	97.2%
Event-Driven	5000	250 Hz	78.8%	82.54%

Table 16 Worst case CPU usage extrapolations for different numbers of skin cells and sample rates. The packet rate and the CPU usage ratios are ratios with respect to clock-driven references.

The presented extrapolations provide a first impression on the performance of the event-driven approach in LASSs and its potential to render LASSs feasible. However, these extrapolations base on empirical models and, of course, cannot account for all system characteristics that will impact the performance of real LASSs. For instance, the CPU usage model does not consider saturation and packet loss. The extended evaluation of a real LASS in Section 4.6 indeed shows that saturation and packet loss have a large impact on performance, especially in the clock-driven operation mode.

4.6. Evaluation of a Large-Area Event-Driven Skin System

This section demonstrates the feasibility of this thesis' event-driven approach for Large-Area Skin Systems (LASSs) by evaluating its implementation in a modular, event-driven large-area e-skin. This e-skin integrates more than 10 000 multi-modal sensors in 1260 skin cells distributed on the body surface of a humanoid robot (Section 4.3). The detailed evaluation of the implemented large-area e-skin refines the performance models of the previous sections with the experimental data of a larger e-skin system, and proves the efficiency of this thesis' approach in a real LASS. Furthermore, the evaluation of experiments with intensive large-area stimulations demonstrates the event-driven e-skin's capability to effectively handle and provide large-area tactile feedback. In the clock-driven operation mode, the same large-area e-skin lacks this capability. There, the evaluations show that the information handling system saturates, and information is continuously lost. This saturation and information loss results in delays and discontinuities, which both severely deteriorate the clock-driven e-skin's capability to provide effective tactile feedback. Therefore, this section not only evidences that this thesis' event-driven approach is effective and efficient in the realization of a LASS. It also demonstrates that the efficiency of this thesis' approach is strictly required to realize a LASS with the capability to provide real-time feedback for large-area contacts. The actual utilization of this feedback for realizing physical interactions and the validation of their effectiveness will follow in Chapter 5.

This section presents the evaluations as follows. First, Section 4.6.1 introduces the experimental setup. Then, Section 4.6.2 presents the refined CPU usage model of the LASS. After that, Section 4.6.3 introduces the CPU usage models for multiple skin drivers and compares the models to the real measurements. Section 4.6.4 assesses the measurements and models and determines the operation zones of the LASS, that is, the ranges of packet rates, where the LASS operates optimally or where the system performance degrades because of saturation and packet loss. Then, Section 4.6.5 presents performance extrapolations to even larger e-skin systems. Finally, Section 4.6.6 presents a worst-case experiment with an intensive large-area stimulation and assesses the effectiveness of the implemented event-driven LASS.

4.6.1. Experimental Setup and Protocol – Event-Driven Large Area E-Skin

Figure 42 illustrates the LASS of the humanoid robot H1 that this section evaluates. Section 4.3 has presented its detailed description along with the definition of the performance indicators for its evaluation.

The work presented in Section 4.6 was in part published in:
Bergner, F., Dean-Leon, E., Guadarrama-Olvera, J. R., Cheng, G., "Evaluation of a Large Scale Event Driven Robot Skin". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4247–4254.

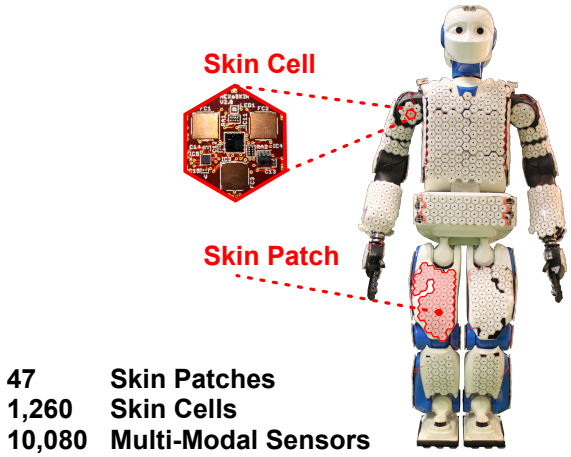


Figure 42 The humanoid robot H1 covered with e-skin. The high-resolution temperature sensor is not deployed. Additionally, acceleration events are deactivated. The robot produces vibrations that have not been compensated yet. The evaluation thus focuses on the tactile force, proximity, and temperature stimuli.

The change of the experimental platform from the UR5 robot arm (Sections 4.4 and 4.5) to the robot H1 requires the adjustment of the event thresholds because the noise profiles change with the experimental platform. The tuning of the event thresholds follows the same procedure as in the previous sections. Table 17 lists the results.

Modality	Proximity	Force	Temperature
Threshold	0.005	0.02	0.5

Table 17 The selected event thresholds for the different modalities of the e-skin. The force and proximity thresholds are normalized, and the temperature is in °C. The event thresholds are adjusted to the new system and are thus different from the ones introduced in Table 8. The noise profiles of the accelerometers are not determined yet. The H1 robot produces vibrations that are not adequately filtered yet.

The evaluations base themselves on two sets of experiments. One set to model, evaluate, and extrapolate the performance of the LASS in Sections 4.6.2 to 4.6.5. The other set to compare the event-driven with the clock-driven operation mode in a whole-body tactile perception task (Section 4.6.6). In total, more than 90 experiments allow for presenting the results with their stochastic significance. Each presented data point in the following figures and tables results from the statistical analysis of at least ten experiments.

4.6.2. CPU Usage Model

The CPU usage model describes the relationship between the packet rate of the e-skin and the demanded CPU usage for handling its packets. This relationship depends on the computer system that realizes the information handling. Ideally, the model should be valid for different computer systems with a small change of parameters that reflect the distinct properties of the computer systems. Therefore, this section determines the parameters for the e-skin system of H1 and assesses the results in comparison to the e-skin system on the UR5 robot arm presented in Section 4.5.3. In the following, the refined CPU usage model serves

two purposes. First, the CPU usage model allows for assessing and extrapolating the performance of the evaluated e-skin system. Second, the CPU usage model allows for deciding if the packet handling process saturates or not (Section 4.6.3).

Figure 43 presents the relationship between packet rate and CPU usage for one of the 12 skin driver processes (RLA, see Table 5 in Section 4.3.4.1) of H1's LASS. Different packet rates result from changing the number of skin cells and their sample rate. To realize a constant packet rate (on average) in the event-driven mode, the event thresholds are lowered such that the skin cells continuously generate event packets. These packets result from noise. Figure 43 additionally presents the measurements and models of the e-skin on the UR5 robot arm to allow for comparisons.

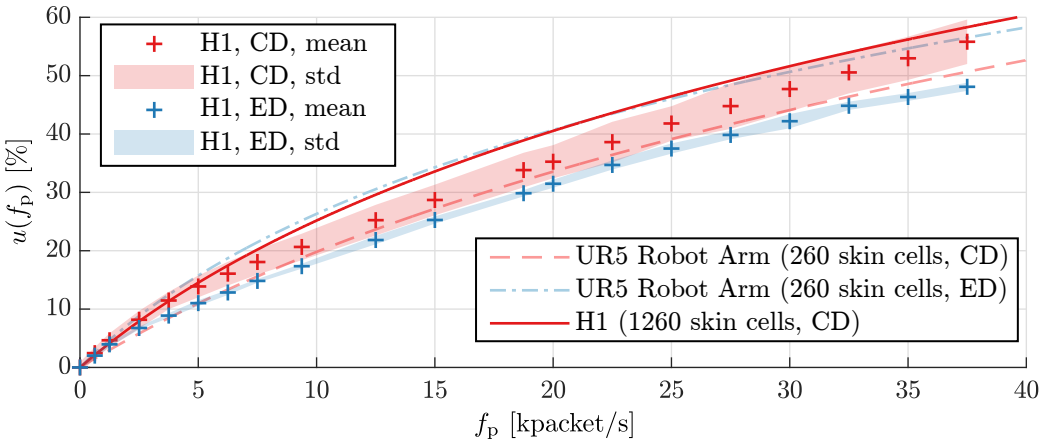


Figure 43 The CPU usage against the packet rate. The figure contains measurements and models for the skin system evaluated in Section 4.5.3 and the LASS of H1. The measurements of the LASS are attached with their stochastic significance. The clock-driven measurements and models are denoted with CD and the event-driven ones with ED. The shadow depicts their variance (std) and the crosses their expected values (mean). The model fitted to the clock-driven measurements of the LASS can be utilized as upper bound.

Figure 43 shows that the logarithmic model introduced in Section 4.5.3 is also valid for the LASS on H1. Additionally, the figure underlines that, despite the different computer systems and the different operation modes, the CPU usage per transmission rate is approximately the same for all measured configurations. Furthermore, the model for H1's e-skin in the clock-driven operation mode

$$u(f_\mathrm{p}) = 40.22 \log(8.698 \cdot 10^{-5} f_\mathrm{p} + 1) \tag{4.31}$$

represents the upper bound for all measurements, including the ones of the previous section. Therefore, this model represents the ideal baseline model for all the following performance evaluations. There, the baseline model delivers the upper bound results for assessments and extrapolations of the CPU usage.

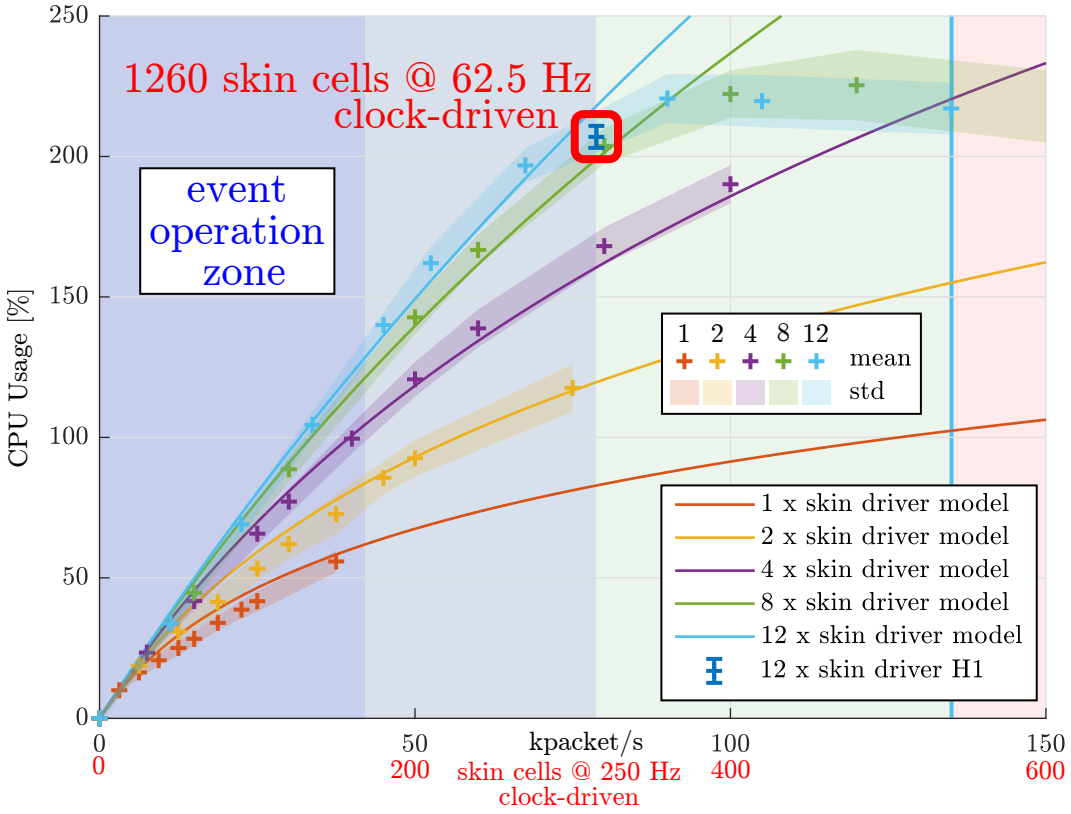
4.6.3. CPU Usage Model for Multiple Processes

The LASS on H1 employs a total of 12 interfaces (TSU-S, see Section 4.3.4) connected to H1's computer system. The computer system executes one process (a skin driver) per interface, thus 12 processes concurrently. Therefore, to evaluate the LASS of H1, the CPU usage model of one process (Section 4.6.2) has to be extended to multiple processes. The CPU usage of multiple processes is the sum of the CPU usage of individual processes

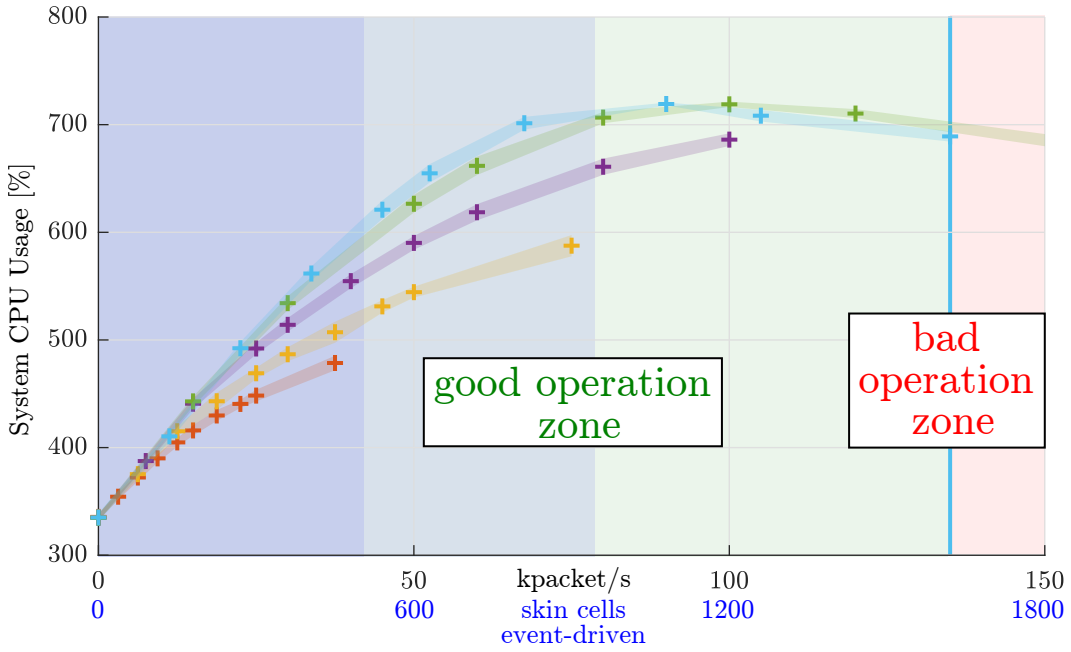
$$u_{\mathrm{total}}(f_{\mathrm{p}}) = \sum_i u_i(f_{\mathrm{p},i}). \tag{4.32}$$

The CPU usage model is valid until the whole PC is saturated, the skin driver is saturated, or the system drops packets. In these cases, the measured CPU usage falls below the estimated CPU usage of the model since it cannot increase further (saturation) or decreases because of information loss.

Figure 44a depicts the CPU usage models for multiple skin drivers (solid lines) and their corresponding real measurements (crosses for means, and shadows for standard deviations). The total CPU usage of the system is depicted in Figure 44b. Figure 45 contains two CPU usage models for 12 skin drivers. Similar to the other models, the light blue model assumes balanced packet rates for all skin drivers, i.e. each of the 12 skin drivers receives 1/12 of the total packet rate. The dark blue model/measurement considers the unbalanced packets rates as found in the e-skin on H1, where each driver is connected to a different number of skin cells (Table 5 in Section 4.3.4.1).



(a) The measurements and the models for the CPU usage for different numbers of skin drivers. The differences between measured values and the model indicate saturation and packet loss.



(b) The measured CPU usage of the whole OS. The measurements include the CPU usage of the evaluation system. The operating system starts to saturate with a CPU usage of around 700 %. The physical limit of the eight-core system (4 cores + HT (Hyper Threading, Intel)) is 800 %.

Figure 44 The CPU usage for the 12 skin drivers and the CPU usage of the total system. The former includes the CPU usage models as defined in Equations (4.31) and (4.32). A significant difference between measurement and model indicates a saturated system that drops packets. The CPU usage measurements of the system provide additional information to decide if the saturation occurs in the skin driver process or if the operating system is saturated. The colored background indicates the good/bad operation zones of the 12 skin driver configuration. The specified numbers of skin cells in event-driven mode empirically assume (Section 4.5) that skin cells in the event-driven mode generate, on average, only one third of the packet rate of skin cells operating in the clock-driven mode. The blue background encloses the operation zone of the LASS for the experiment presented in Section 4.6.6.

Comparing the different performance indicators in Figures 44 and 45, one can observe that the model for the CPU usage differs significantly from the observed CPU usage when the model's constraints are violated. The system is either saturated or drops packets. This observation leads to the definition of operations zones in Section 4.6.4.

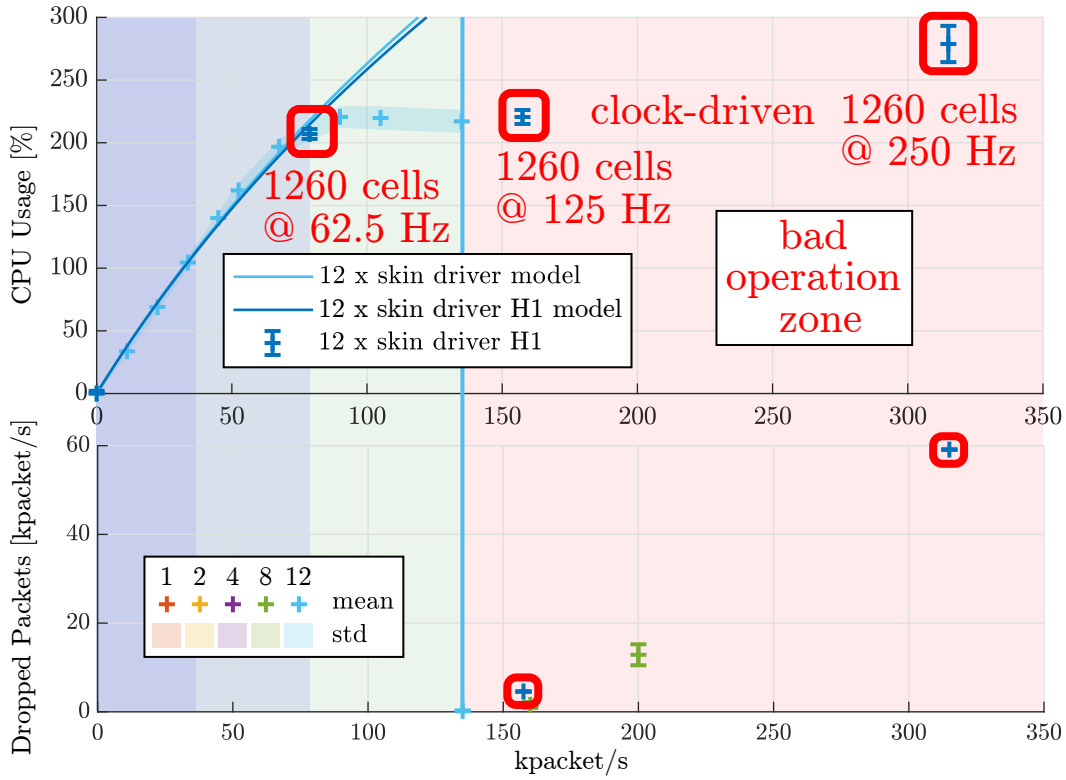


Figure 45 The CPU usage of 12 skin drivers and the dropped packets per second. The dropped packets indicator is explained in detail in Section 4.3.5.6.

4.6.4. Operation Zones of the Large-Area Skin System

The background in Figures 44 and 45 marks the different operation zones of the fully operating LASS on H1 with 12 skin drivers and 1260 skin cells (Section 4.6.1). The vertical light blue line indicates the border between zones with zero/non-zero packet drops. The line divides the figures into two parts with the good operation zone on the left (zero packet drops) and with the bad operation zone on the right (packet drops). The thick red squares in Figures 44 and 45 accentuate the operation points of the system in clock-driven mode with the sample rates of 62.5 Hz, 125 Hz, and 250 Hz. Only one of these operation points (62.5 Hz) lies in the good operation zone. Still, in this operation point, the LASS continuously induces a high CPU usage (close to the saturation of the whole computer). The other two operation points (125 Hz, and 250 Hz) lie within the bad operation zone. These two operation points are not feasible.

The blue operation zones on the very left side of Figures 44 and 45 depict the operation area of the LASS in event-driven mode for the experiment presented in Section 4.6.6. The sensors of the skin cells sample with 250 Hz. The experiment incorporates an intensive stimulation of

large skin areas. However, the operation points of the system in the event-driven mode stay most of the time on the very left in the dark blue zone and they thus induce only low CPU usage. Some peak stimulations occasionally move a few operation points into the light blue zone. These peak CPU usages have only a minor impact and their effect on the performance of the system is neglectable.

4.6.5. Extrapolation towards Larger Skin Systems

The CPU usage models introduced in Sections 4.6.2 and 4.6.3, and their validation with real measurements can be exploited to extrapolate the CPU usage for larger packet rates. Figure 46 depicts these extrapolations.

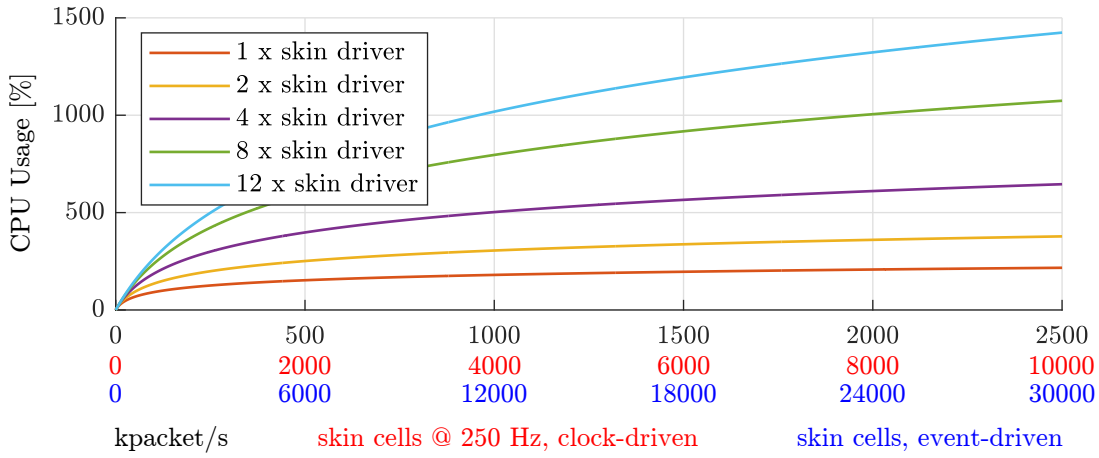


Figure 46 The comparative extrapolation for higher packet rates with a LASS operating in clock-driven and event-driven mode.

The estimated number of skin cells for the LASS operating in the event-driven mode bases on empirical results (Section 4.5). Skin cells in event-driven mode generate, on average, only one-third of the packet rate of skin cells operating in the clock-driven mode. The total CPU usage increases with the number of skin drivers, see Figure 46. This fact displays the need to trade-off between CPU usage and the modularity. However, the CPU usage model does not reflect that a real skin driver cannot be utilized up to its physical limit of 100 %. Depending on the overall CPU usage of the system, the skin driver will saturate before reaching 100 %. Thus, the LASS only properly scales with increasing the number of skin drivers. Then, the LASS can exploit the multi-threading capabilities of computer systems and scale with the number of CPU cores. This capability fits well with the current trend in technology to increase the number of CPU cores in computer systems.

4.6.6. Effectiveness of the Large-Area Deployment

This section presents a comparison of the performance of the LASS in the clock-driven and the event-driven modes. Therefore, a whole-body tactile perception experiment was conducted as depicted in Figure 47. For both modes, the sensors sample at 250 Hz.

In the clock-driven mode, the CPU usage, packet rate, and packet loss are continuously very high during all stages of the experiment. Contrarily, in the event-driven mode, the CPU usage and packet rate are low when no tactile stimuli are present and tractable when many tactile stimuli (up to 680 active skin cells) are present. In the event-driven mode, the system only once lost 80 packages. That loss is by far lower than 0.1 % of all captured packets. The results of Figure 47 are supported by the statistical evaluation of the experimental stages in Table 18. In any case, Figure 47 clearly evidences that the only feasible way to utilize the evaluated LASS in applications that require feedback is operating it in the event-driven mode.

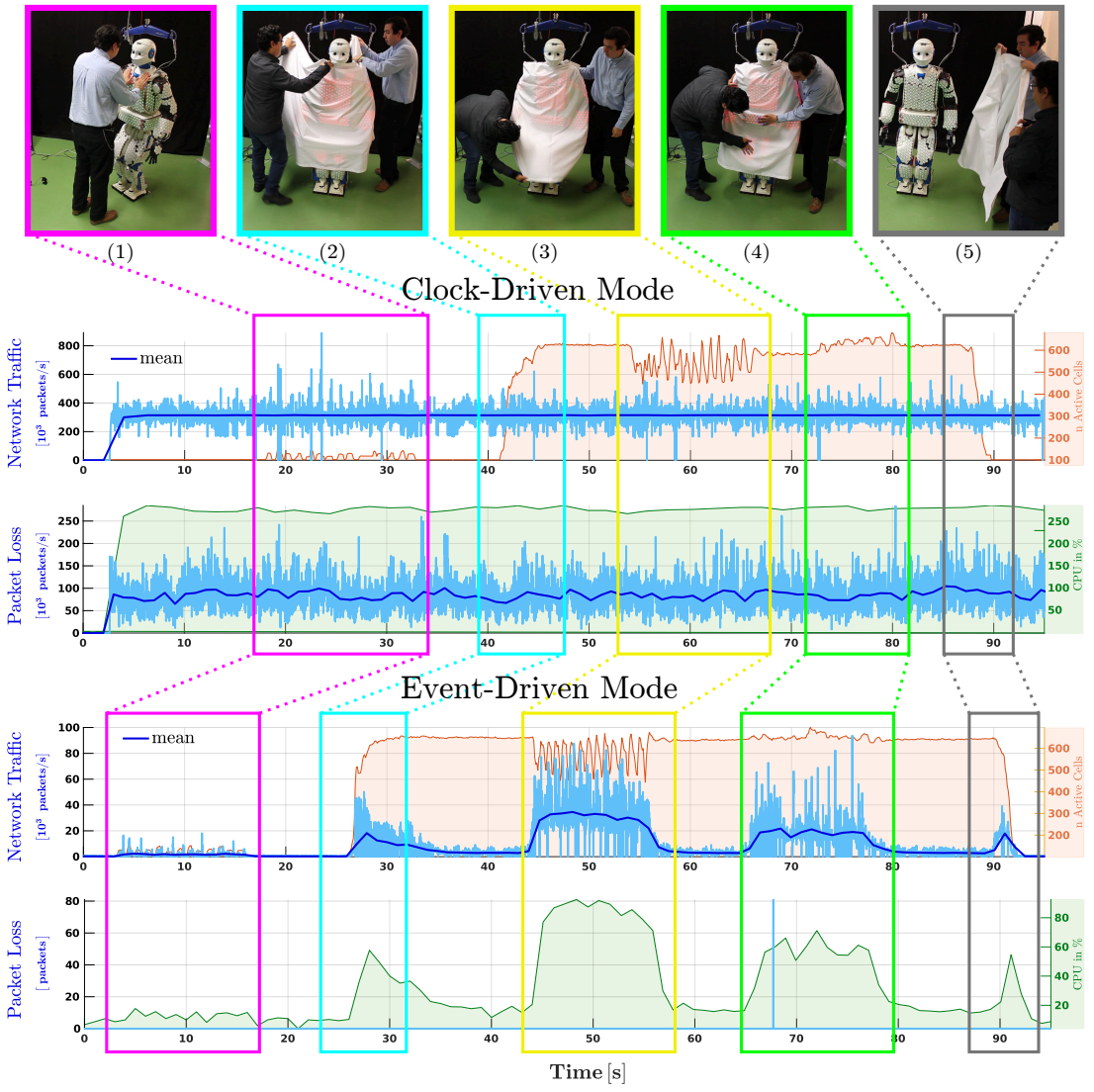


Figure 47 The whole-body tactile perception experiment to compare the performance of LASS in the clock-driven and event-driven operation mode. (1) We repeatedly touch the robot on its torso, mainly generating proximity and force stimuli in the interaction area. (2) We cover H1 with a cloth, generating many proximity stimuli in a large number of skin cells. (3) We move the cloth, again generating many proximity stimuli. (4) We touch H1 indirectly by touching the cloth, generating force, and proximity stimuli in the interaction area. (5) We uncover H1, generating proximity stimuli. The stochastic significance of the results can be found in Table 18. Please note that the scale of the network traffic and the CPU usage is significantly lower in the event-driven mode than in the clock-driven mode.

Experiment Stage	1 (CD)	3 (CD)	4 (CD)	1 (ED)	3 (ED)	4 (ED)
CPU in %	260.656 (2.79748)	258.942 (4.40611)	261.59 (4.04644)	40.3544 (0.520806)	100.445 (1.59067)	64.1453 (2.41937)
Network traffic in packets/s	310,241 (366.196)	310,191 (328.908)	310,276 (257.035)	11,594.9 (76.2378)	34,221.6 (754.492)	19,407.4 (1,039.42)
Active cells	131.838 (3.52125)	591.409 (23.0707)	686.341 (23.3716)	136.236 (2.42389)	574.274 (13.4872)	666.372 (11.1482)
Dropped packets / s	82,233 (2,450.49)	84,203.4 (3015.49)	81,268 (3,526.51)	0 (0)	0 (0)	0.008 (0.0252982)

Table 18 Statistical evaluation of the experiment depicted in Figure 47. CD denotes the clock-driven mode and ED the event-driven mode. The table contains the averages/expected values for stages 1, 3, and 4 for all conducted experiments (10 trails per mode). The standard deviations are denoted in the brackets. Since we cannot feasibly repeat the experiments in exactly the same manner, we average the measurements of stages 1, 3, and 4 for each conduced experiment. Evaluating the stochastic processes of stages 2 and 5 is not feasible.

The results demonstrate the superior performance of the event-driven LASS. In the clock-driven mode, the LASS continuously produces 315 000 packet/s, while in the event-driven mode the system at most produces 40 000 packet/s (13 %). The CPU usage reduces from continuously 270 % to at most 100 % (37 %). In the clock-driven mode, the computer drops on average 80 000 packet/s (25 % of all packets), while in event-driven mode, the package loss is practically neglectable (< 0.1 %). The efficiency of the event-driven LASS enables its complete on-board integration into a humanoid robot without the need for additional external power or processing capabilities.

4.7. Summary

This chapter presented the design and realization of this thesis' event-driven approach for LASSs with subsequent empirical assessments and evaluations validating its efficiency and effectiveness. The design and realization utilized the event-driven approach (Chapter 3) to solve the challenge of efficiently handling information in LASSs. The performance evaluations and assessments evidenced the efficiency of the realized event-driven design allowing for the scale-up of existing e-skin systems to LASSs that can provide low-latency feedback while being computationally efficient. The design of the event-driven information handling system for standard computers not only provides the efficiency to scale the LASS, but it also provides the flexibility to enhance existing skin systems without further hardware modifications.

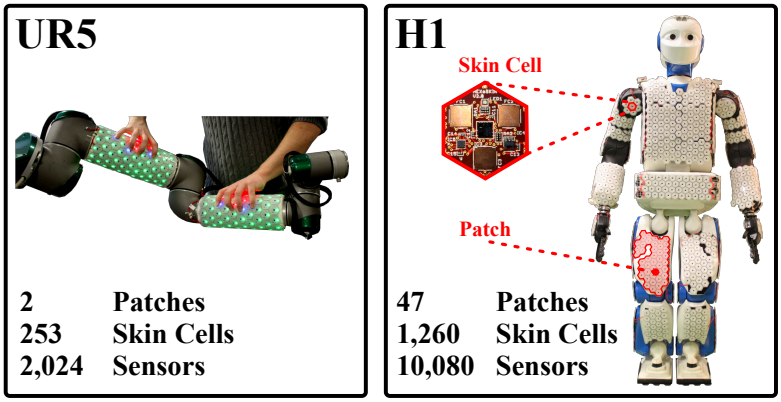
The feasibility of the proposed event-driven sensing and event-driven information handling were demonstrated by their successful implementation on an existing e-skin system. Furthermore, the designs successfully elevated the e-skin to an effective LASS. Table 19 lists the challenges of LASSs, the designs and realizations that contributed to their solution or mitigation, and the designs' impacts on improving the implemented e-skin system. The contributions of this chapter are highlighted in green.

The subsequent evaluation of the implemented LASS on two experimental platforms evidenced the effectiveness of the presented designs. Figure 48 summarizes the obtained results. The event-driven approach reduces the network traffic by 94 % and the CPU usage of tactile perception by 81 %. The LASS can only effectively operate in event-driven mode. In clock-driven mode, the LASS loses continuously 25 % of the tactile information while the loss in event-driven mode is practically neglectable.

This chapter evaluated the designs of the LASS in implementations with experiments focusing on tactile information, which is on perception. The subsequent Chapter 5 will proceed beyond perception and will validate the LASS in applications that act on tactile information.

Challenges	Designs/Realizations	Impacts/Results (UR5 + H1)
Reliability/Robustness (C-1)	• modules: hexagonally shaped skin cells • local processing capabilities at skin cells • redundant meshed network of skin cells • dynamic routing	• upto N+3 redundancy in skin patches • automatic online failure recovery within less than 50 ms
Deployability (C-2)	• hierarchical modular structure (cells, patches, segments) • self-organizing network	• no manual construction of communication trees • flexible addition/removal of modules (cells, patches, segments) • eased deployment of: – UR5: 2 patches instead of 253 cells (2,024 sensors) – H1: 47 patches in 12 segments instead of 1260 cells (10,080 sensors)
Wiring (C-3)	• hierarchical modular structure • modular interfaces	• huge reduction of wire count by a factor > 50 • reduction of connections from: – UR5: 253 (2,024) point-to-point to 4 patch connections – H1: 1260 (10,080) point-to-point to 12 interface connections
Sensor Localization (C-4)	• 2D information of self-organized modular structure • rotation measurement between cells • automatic 3D surface reconstruction of patches	• automatic self-calibration of relative sensor locations • huge reduction of manual localization tasks by a factor > 80 • manual/semi-automatic localization of: – UR5: 2 patches instead of 253 cells (2,024 sensors) – H1: 47 patches instead of 1260 cells (10,080 sensors)
Efficient Information Handling (C-5, C-6)	• modular SoDP with event generators in skin cells • efficient event decoders to bridge to clock-driven algorithms • event-driven information handling framework exploiting the asynchronous scheduling capabilities of standard operating systems	• the system scales well from 253 to 1260 cells • EDS is the key for information handling in LSSSs: – loss-less information handling (H1: 25% loss when clock-driven) – effective reduction of the data rate by around 90% – efficient information handing: computational load reduced by around 60%

Table 19 Challenges, designs, and their impacts on scalable Large-Area Skin Systems (LASS). The contributions of this chapter are highlighted in green.



(a) Skin on the UR5 robot arm and the humanoid robot H1.

Robot	Network Traffic	Clock-Driven	Event-Driven	Reduction
UR5	Idle	1.0 MB/s	0.0024 MB/s	99.76%
	Interaction (peak)	1.0 MB/s	0.0791 MB/s	92.09%
H1	Idle	19.2 MB/s	0.0146 MB/s	99.924%
	Interaction (peak)	19.2 MB/s	2.4414 MB/s	87.284%
			Average	**94%**

(b) Network traffic.

Robot	Information Loss	Clock-Driven	Event-Driven
UR5	(saturated system)	0%	0%
H1		25%	0%

(c) Information loss in a saturated system.

Robot	CPU Usage (Perception Module)	Clock-Driven	Event-Driven	Reduction
UR5	Idle	95%	5.5%	94%
	Interaction (peak)	95%	25%	74%
H1	Idle	270%	10%	96%
	Interaction (peak)	270%	100%	63%
			Average	**81%**

(d) CPU Usage in the perception module.

Figure 48 Overview of the network traffic and CPU usage of *perception* on the two evaluated experimental setups. The UR5 robot arm's sensors sample with 125 Hz and H1's with 250 Hz. The idle e-skin system does not register any tactile interactions. Any interaction with the e-skin generates information rate peaks that mirror event rate peaks and thus lead to peaks in the network traffic or the CPU usage. The tactile interactions varied during our experimental setups. We ensure the comparability in this overview by focusing on the measured peaks and treating them as worst case measurements. 100 % CPU usage equals the saturation of one CPU core in HT mode. The CPU usages measured in the clock-driven setup of H1 are lower than expected since the information handling system is totally saturated and can not assign more CPU time to the perceptive information handling. The CDS of H1 effectively handles 25 % less information than required since 25 % of the information is lost.

5. Realization and Validation in Applications

This chapter presents the effectiveness of the event-driven Large-Area Skin System (LASS) in the very challenging task of providing large-area tactile feedback in complex systems. While the previous chapters presented the design, realization, and evaluation of event-driven LASSs, this chapter demonstrates that LASSs implementing this thesis' approach can provide the tactile feedback for previously infeasible whole-body interactions.

To this end, this chapter presents the successful integration of the event-driven LASS in robotic systems, which not only validates the effectiveness of this thesis' approach in complex systems but which also delivers insights and general methods for efficient integrations.

This chapter collects and analyzes in total four methods: i) Hybrid event-driven systems, that efficiently connect event-driven e-skin with standard clock-driven control algorithms (Section 5.1); ii) The decentralization of computations, that distributedly process tactile feedback (Section 5.2); iii) The exploitation of event-driven information, that increases the efficiency of clock-driven algorithms (Section 5.3); and iv) The transformation of clock-driven to event-driven control algorithms, that increases the efficiency of the algorithms (Section 5.3).

Through these four methods, this thesis provides the foundation for the effective utilization of large-area tactile feedback in complex systems (Section 5.4) whereby previous approaches failed.

5.1. Hybrid Event-Driven Systems

This section presents the integration of the event-driven e-skin with clock-driven control to evaluate the effectiveness and efficiency of hybrid event-driven systems. Hybrid event-driven systems ensure the flexibility and applicability of the approaches presented in this thesis since they represent the successful and efficient integration of large-area event-driven e-skin systems in existing complex clock-driven systems without the need for further modifications.

The importance of hybrid event-driven systems lies in the fact that many algorithms in complex systems, for instance, controllers, are still clock-driven. Especially real-time low-level control for actuation is often provided by third parties and cannot be modified. Furthermore, the stability of controllers depends on the theory that the Nyquist-Shannon sampling theorem is fulfilled at all times (Section 4.2.2). Control in event-driven systems is for itself an emerging new research field [99], and up to the near future, Event-Driven Systems (EDSs) will need to be combined with Clock-Driven Systems (CDSs).

One way to demonstrate the effectiveness of this thesis' event-driven approach for LASSs is to show that clock-driven algorithms can efficiently utilize their event-driven feedback. This section validates this effectiveness by implementing a perception/action loop where a robot arm reacts to tactile stimuli (Section 5.1.1). The clock-driven control algorithm fuses tactile feedback with proprioceptive robot information to motor actions (Section 5.1.2). The subsequent performance evaluation (CPU usage of perception and control, and latency) compares the hybrid event-driven system with respect to a purely clock-driven reference (Section 5.1.3).

The results presented in this chapter demonstrate that the hybrid event-driven system profits from its event-driven components and is more efficient than a clock-driven system. Thus, the combination of event-driven LASSs with complex clock-driven systems is effective, and the overall hybrid event-driven system is more efficient than its clock-driven counterpart. Both results empirically prove the effectiveness and efficiency of this thesis' approach in complex systems.

5.1.1. Experimental Setup and Protocol – Hybrid Event-Driven Systems

The experimental platform is the UR5 robot arm covered with 253 skin cells as shown in Figure 49. In contrast to the previous experiments presented in Chapter 4 that focused on tactile sensing, this experimental setup incorporates controllers implementing reactions to tactile interactions. In event-driven mode, the optimal event thresholds of Table 8, determined in Section 4.4.4, were used. Figure 49 depicts the implemented perception-action loop.

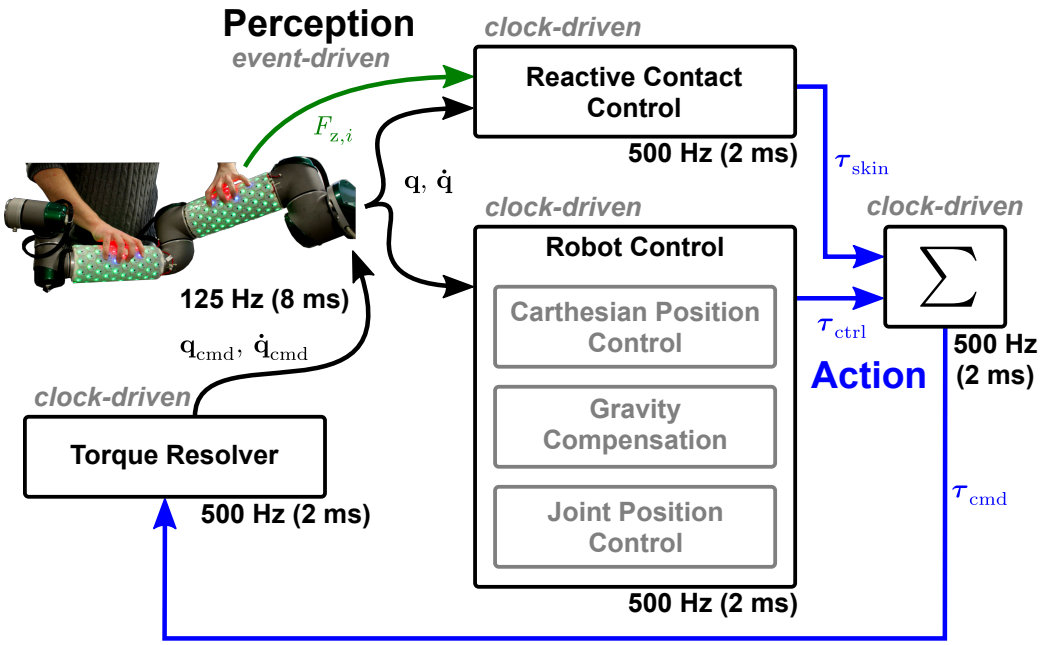


Figure 49 The controller block fuses tactile with proprioceptive information. The reactive contact controller is combined with several other controllers to achieve a desired control behavior, e.g. a compliant behavior. The motor actions are generated with joint torques $\boldsymbol{\tau}$. The torque resolver [40, 44] transforms the resulting joint torque $\boldsymbol{\tau}_{\text{cmd}}$ to joint position $\mathbf{q}_{\text{cmd}}$ or joint velocity commands $\dot{\mathbf{q}}_{\text{cmd}}$.

The control block consists of two types of controllers: controllers only operating with proprioceptive information (joint positions $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$), and controllers fusing tactile with proprioceptive information. The former are controllers such as Gravity Compensation Control, Joint Position Control, and Cartesian Position Control, and the latter are multi-modal controllers, such as Reactive Contact Control [45, 40, 44], see Table 20.

Controller	Description
Gravity Compensation Control	This controller compensates for the effects of gravity such that forces produced by the gravitational acceleration do not affect the robot arm.
Joint Position Control	This controller controls the position of the robot arm in the joint space, that is, it forces the robot arm to reach a desired joint position $\mathbf{q}_{\mathrm{d}} \in \mathbb{R}^{\mathrm{DOF}}$.
Cartesian Position Control	This controller controls the position of the robot arm in the Cartesian space, that is, it forces the robot arm to reach a desired position $\mathbf{p}_{\mathrm{d}} \in \mathbb{R}^3$.
Reactive Contact Control	This controller fuses proprioceptive with tactile information and generates reactions according to the tactile feedback, that is, it forces the robot arm to avoid contacts.

Table 20 Overview and description of controllers [45, 40, 44]. The controllers can be combined to achieve desired behaviors, for instance, contact avoidance, or active compliance.

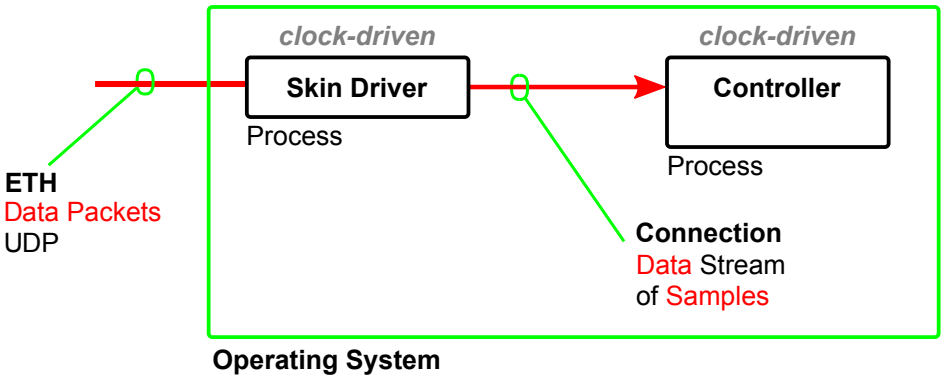
Section 5.1.2 will detail how the Reactive Contact Control fuses proprioceptive information with tactile information to motor commands.

All controllers encode their motor actions in joint torques $\boldsymbol{\tau}$, which are the desired torques commanded to the joints' motors to achieve the desired system behavior (e.g. compensate gravity, reach a desired position, avoid a contact, etc.). Joint torques do not require coordinate frames, and thus, behaviors can be superimposed by adding and weighting their respective joint torque contributions. When a robot supports joint torque commands, then the sum of the joint torques can be directly sent to the robot as a control command. Since the UR5 robot arm only supports position and velocity commands, a torque resolver first converts the joint torque commands $\boldsymbol{\tau}_{\mathrm{cmd}}$ to position $\mathbf{q}_{\mathrm{cmd}}$ or velocity commands $\dot{\mathbf{q}}_{\mathrm{cmd}}$ [45, 40, 44].

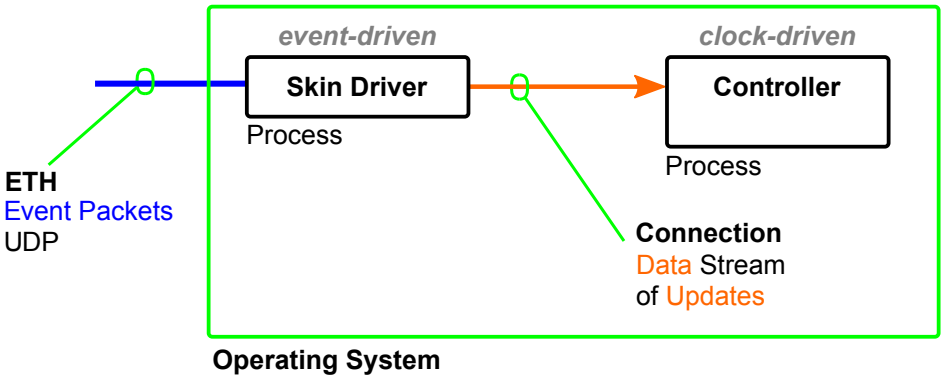
The controllers of the experimental setup are executed in a real-time loop with an update rate of 500 Hz[1], or respectively, with a cycle time T_{ctrl} of 2 ms. A velocity interface to the UR5 robot arm was employed and velocity commands were sent with an update rate of 125 Hz.

The experiments are conducted both in the clock-driven setup (Figure 50a) providing the ground truth information, and in the hybrid setup (Figure 50b). Additional information regarding the implementation of the hybrid event-driven information handling system is presented in Appendix C, Figure 81. In the hybrid setup, the skin driver operates in event-driven mode. The controller always operates in the clock-driven mode. The communication between the event-driven skin driver and the clock-driven controller is update-driven (Appendix C), that is, the communication only contains the data structures of the skin cells that need to be updated. Therefore, in the hybrid setup, less CPU usage is expected in the controller process.

[1] The closed loop dynamics use a virtual dynamic model which requires a fast update frequency, in this case 500 Hz. It is noticeable that slower frequencies lead to instabilities.



(a) The clock-driven reference system.



(b) The hybrid system with the event-driven skin driver and the clock-driven controller. The communication between the skin driver and the controller is update-driven (Appendix C).

Figure 50 The clock-driven reference and the hybrid system. ETH denotes the Ethernet connection of the communication interface to the e-skin. Additional information regarding the implementation of the hybrid event-driven information handling system is presented in Appendix C.

The experimental evaluation analyzes the performance indicators (Section 4.3.5) and the responsiveness of the system. Therefore, a total of 30 experiments were performed as depicted in Figure 51.

The experiments have to be performed as similar as possible to acquire comparable information. First, in Stage 1, touch is made on a resting robot arm. With the applied force, the robot moves away to avoid the contact (Stage 1 to Stage 2). Once touching the robot stopped (Stage 2), the robot continues its movement (due to its natural dynamics, e.g. interia and friction), but slows down (Stage 2 to Stage 3). Finally, the robot arm moves back to its initial position (for instance, in Stage 4).

The controller of the experiments implements a compliant behavior and combines Gravity Compensation Control, Reactive Contact Control, and Joint Position Control (Figure 49).

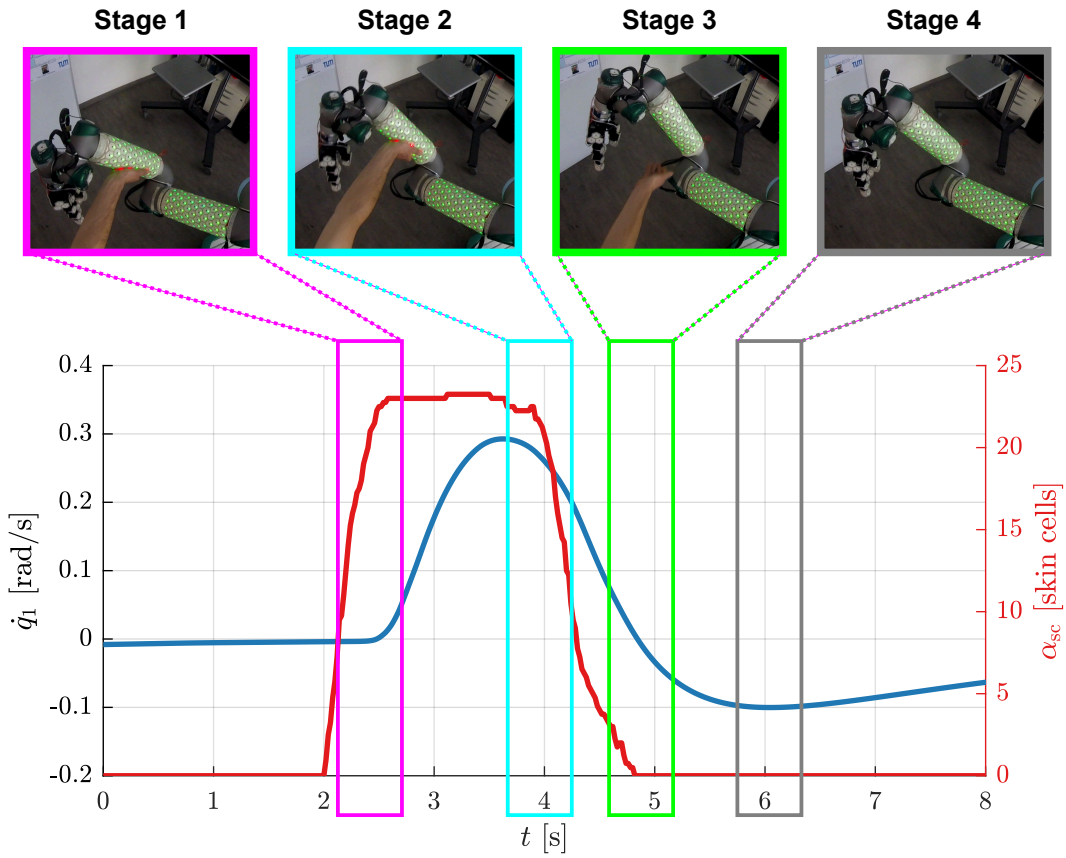


Figure 51 The experiment for evaluating the hybrid event-driven system. The experiment is carried out five times for each operation mode (clock-driven/event-driven) and for each sample rate (62.5 Hz/125 Hz/250 Hz). Stage 1: The robot arm is touched. The number of active skin cells α_{sc} increases. The robot arm starts to move and avoids the contact $\dot{q}_1 > 0$. Stage 2: The contact ceased. The number of active skin cells decreases. The movement slows down ($\dot{q}_1$ decreases). Stage 3: The robot arm is no longer touched. The number of active skin cells is zero. The movement stops and changes its direction $\dot{q}_1 < 0$. Stage 4: The robot arm moves back to its original position yielding a compliant behavior.

The experiments are conducted with a human in the loop. Therefore, to minimize the effects between small differences in our experimental trials, each experiment was carried out five times to allow for stochastic evaluations. Six different experiments were performed for the three different sample rates of the skin cells' sensors (62.5 Hz, 125 Hz, and 250 Hz) with the skin driver operating in clock-driven or event-driven mode.

5.1.2. Reactive Contact Control – From Tactile Stimuli to Motor Commands

This section describes the fusion of tactile stimuli provided by the e-skin with proprioceptive information provided by the robot. This fusion is mapped to motor commands that realize appropriate reactions for tactile interactions. The mapping of tactile stimuli to motor commands requires structural information, which is the locations (position and orientation) of the tactile sensors with respect to the limbs of the robot, and the structural arrangement of the robot limbs (the robot kinematics). This structural information can be automatically provided by the e-skin system [102, 107, 45, 29]. The e-skin can automatically localize its skin cells on 3D surfaces, and self-exploration or minimal external information can contribute to the automatic acquisition of sensory-motor mappings [107].

In the following, a step by step presentation of the transformation of tactile stimuli to virtual forces (Section 5.1.2.1) and the mapping of these forces to joint torques (Section 5.1.2.2) is given. Algorithm 4, presents the skin cell wise computation of skin torques (Section 5.1.2.3).

5.1.2.1 Virtual Forces – From Tactile Stimuli to Forces

The skin cells i detect external normal forces along their own $\mathbf{z}$-axes. A virtual force $F_{\mathrm{z},i} \in \mathbb{R}$ for each skin cell i is computed by fusing the three normalized capacitive forces $F_{\mathrm{c}} = \sum_{l=1}^{3} F_{\mathrm{c},l}$ with the normalized proximity value F_{p}, see Algorithm 3 [40, 17].

Algorithm 3 Calculate F_{z} for F_{c} and F_{p}

$F_{\mathrm{c}} := \sum_{l=1}^{3} F_{\mathrm{c},l}$
if $F_{\mathrm{c}} < F_{\mathrm{c,th}}$ **then**
    $F_{\mathrm{c}} := 0$
end if
if $F_{\mathrm{p}} < F_{\mathrm{p,th}}$ **then**
    $F_{\mathrm{p}} := 0$
end if
$F_{\mathrm{z}} := \beta_{\mathrm{c}}\, F_{\mathrm{c}} + \beta_{\mathrm{p}}\, F_{\mathrm{p}}$
if $F_{\mathrm{z}} < F_{\mathrm{z,th}}$ **then**
    $F_{\mathrm{z}} := 0$
else
    $\alpha_{\mathrm{sc}} := \alpha_{\mathrm{sc}} + 1$
end if

Both, F_{c} and F_{p} range from zero to one. Zero denotes zero force, or respectively, maximum distance, and one denotes maximum force, or respectively, zero distance. The fusion of pre-contact (proximity) and contact (force) information is weighted with positive gains β_{c} and $\beta_{\mathrm{p}} \in \mathbb{R}$ and thresholded by $F_{\mathrm{c,th}}$, $F_{\mathrm{p,th}}$, and $F_{\mathrm{z,th}}$. All contributions below these thresholds are set to zero. These thresholds cancel noise. All thresholds for computing the virtual force are determined heuristically and are the result of a trade-off between noise and reactiveness to small forces.

A skin cell i that contributes a virtual force $F_{\mathrm{z},i} > 0$ is an *active skin cell* with $\alpha_i = 1$. The number of active skin cells α_{sc} (Figure 51) is the sum of all active skin cells:

$$\alpha_{\mathrm{sc}} = \sum_{i} \alpha_i. \tag{5.1}$$

The number of active skin cells can be used, for instance, to compute the contact pressure [62].

5.1.2.2 Skin Joint Torques – Fusing Tactile and Proprioceptive Information

The reactive contact controller fuses tactile information provided by the e-skin (virtual force $F_{\mathrm{z},i}$, Section 5.1.2.1) with proprioceptive information provided by the robot arm (joint positions $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$) to desired joint torques $\boldsymbol{\tau}$. The fusion requires a mapping of the virtual force $F_{\mathrm{z},i}$ of a skin cell i to its appropriate joint torque $\boldsymbol{\tau}_i$. The sum of the joint torque contributions of all skin cells i

$$\boldsymbol{\tau}_{\mathrm{skin}} = \sum_i \boldsymbol{\tau}_i \tag{5.2}$$

leads to the skin joint torque $\boldsymbol{\tau}_{\mathrm{skin}}$. This joint torque represents the motor command for the resulting reaction to the tactile contact.

The mapping of virtual forces to joint torques is obtained by applying the principle of virtual work [40]. An external wrench $\mathbf{w}_i \in \mathbb{R}^6$ exerted on a skin cell i contributes to torques $\tau_l \in \mathbb{R}$ in joints $l = 1, \ldots, k$, see Figure 52.

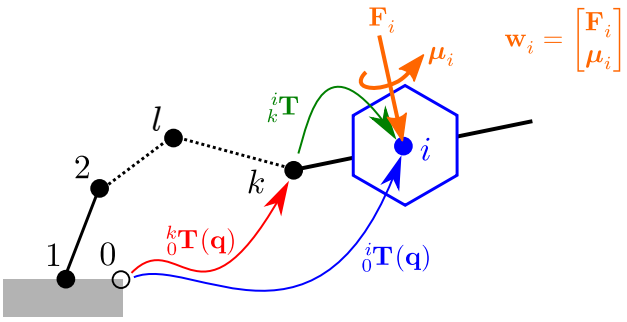


Figure 52 Kinematic chain from skin cell i on active joint k to the base frame of the robot 0.

The torques τ_l in the rest of the joints $l = k+1, \ldots, \mathrm{DOF}$ are zero. Thus, the joint torque contribution $\boldsymbol{\tau}_i \in \mathbb{R}^{\mathrm{DOF}}$ of skin cell i is

$$\boldsymbol{\tau}_i = \begin{bmatrix} \tau_1 & \cdots & \tau_k & 0 & \cdots & 0 \end{bmatrix}^\top \in \mathbb{R}^{\mathrm{DOF}} \tag{5.3}$$

where DOF denotes the robot's degrees of freedom. The UR5 robot arm of the experimental setup (Section 5.1.1) employs six revolute joints and has thus six DOF.

The joints $1, \ldots, k$ actuated by the wrench $\mathbf{w}_i$ of a skin cell i are termed *active joints* since the wrench causes joint torques greater than zero in these joints. All the other joints ($l > k$) are then not actuated since the wrench does not cause joint torques in these joints.

The wrench $\mathbf{w}_i$ is the external wrench sensed by skin cell i and is composed of

$$\mathbf{w}_i = \begin{bmatrix} 0 & 0 & F_{\mathrm{z},i} & 0 & 0 & 0 \end{bmatrix}^\top \in \mathbb{R}^6. \tag{5.4}$$

The skin cells only sense forces aligned to their $\mathbf{z}$-axes (Normal Forces, Section 5.1.2.1), thus the wrench $\mathbf{w}_i$ only contains that contribution.

The mapping from wrenches $\mathbf{w}_i$ to torques $\boldsymbol{\tau}_i$ is resolved by computing the transposed Jacobians $\mathbf{J}_i^\top(\mathbf{q})$ of the skin cells i applying the principle of virtual work:

$$\boldsymbol{\tau}_i = \mathbf{J}_i^\top(\mathbf{q})\ {}_0\mathbf{w}_i \tag{5.5}$$

The wrench ${}_0\mathbf{w}_i$ is the wrench of the skin cell i with respect to the robot base coordinate frame 0 and is computed by

$${}_0\mathbf{w}_i = \begin{bmatrix} {}^k_0\mathbf{R}\ {}_k\mathbf{z}_i\ F_{\mathrm{z},i} \\ 0 \\ 0 \\ 0 \end{bmatrix} \in \mathbb{R}^6 \tag{5.6}$$

where ${}^k_0\mathbf{R}$ denotes the orientation of the coordinate frame of joint k with respect to the coordinate frame 0. ${}_k\mathbf{z}_i \in \mathbb{R}^3$ is the $\mathbf{z}$-axis of the coordinate frame of skin cell i with respect to the coordinate frame of joint k, that is, ${}_k\mathbf{z}_i = {}^i_k\mathbf{R} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$.

The Jacobian $\mathbf{J}_i(\mathbf{q})$ of a skin cell i can be determined geometrically by applying the laws of physics for mapping linear and angular velocities $\dot{\mathbf{x}}$ and $\boldsymbol{\omega}$ to joint velocities $\dot{\mathbf{q}}$. For revolute joints, this Jacobian has the form:

$$\mathbf{J}_i(\mathbf{q}) = \begin{bmatrix} {}_0\mathbf{z}_0 \times [{}_0\mathbf{t}_i - {}_0\mathbf{t}_0] & \dots & {}_0\mathbf{z}_{l-1} \times [{}_0\mathbf{t}_i - {}_0\mathbf{t}_{l-1}] & \mathbf{0} & \dots & \mathbf{0} \\ {}_0\mathbf{z}_0 & \dots & {}_0\mathbf{z}_{l-1} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{6\times \mathrm{DOF}} \tag{5.7}$$

In general, the l-th column of $\mathbf{J}_i$, that is,

$$\mathbf{j}_{l,i}(\mathbf{q}) = \begin{bmatrix} {}_0\mathbf{z}_{l-1} \times \left({}_0\mathbf{t}_i - {}_0\mathbf{t}_{l-1}\right) \\ {}_0\mathbf{z}_{l-1} \end{bmatrix} \in \mathbb{R}^6 \tag{5.8}$$

corresponds to a joint l. Thus, the joint torque $\tau_{l,i} \in \mathbb{R}$ of a joint l, when a skin cell i is mounted on a robot limb actuated by a joint k (Figure 52), is computed by

$$\tau_{l,i}(\mathbf{q}) = \begin{cases} \mathbf{j}_{l,i}^\top(\mathbf{q})\ {}_0\mathbf{w}_i \in \mathbb{R} & \textbf{if} \quad l \le k \\ 0 & \textbf{otherwise} \end{cases}. \tag{5.9}$$

With Equation (5.6), $\tau_{l,i}$ simplifies to

$$\tau_{l,i}(\mathbf{q}) = J_{l,i}(\mathbf{q})\ F_{\mathrm{z},i} \qquad \in \mathbb{R} \tag{5.10}$$

where

$$J_{l,i}(\mathbf{q}) = \left[{}_0\mathbf{z}_{l-1} \times \left({}_0\mathbf{t}_i - {}_0\mathbf{t}_{l-1}\right)\right]^\top\ {}^k_0\mathbf{R}\ {}_k\mathbf{z}_i \qquad \in \mathbb{R}. \tag{5.11}$$

The vector ${}_0\mathbf{z}_{l-1} \in \mathbb{R}^3$ is the $\mathbf{z}$-axis of the coordinate frame $l-1$ of joint $l-1$ and ${}_0\mathbf{t}_i \in \mathbb{R}^3$ is the origin of the coordinate frame of skin cell i, both with respect to the base coordinate frame 0. The transformation from joint l to the base coordinate frame 0 is defined by the homogeneous transformation matrix ${}^l_0\mathbf{T} \in \mathbb{R}^{4\times4}$. This matrix contains the axes $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ and the origin $\mathbf{t}$ of the coordinate frame of joint l with respect to the base coordinate frame 0:

$$ {}^l_0\mathbf{T}(\mathbf{q}) = \begin{bmatrix} {}^l_0\mathbf{R} & {}_0\mathbf{t}_l \\ \mathbf{0}^\top & 1 \end{bmatrix} = \begin{bmatrix} {}_0\mathbf{x}_l & {}_0\mathbf{y}_l & {}_0\mathbf{z}_l & {}_0\mathbf{t}_l \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \tag{5.12} $$

Therefore, computing the joint torque $\boldsymbol{\tau}_i$ of a skin cell i requires the following homogeneous transformation matrices (Figure 52, and Equation (5.10)):

- the forward kinematics from the base coordinate frame 0 to the coordinate frame of joint k: ${}^l_0\mathbf{T}(\mathbf{q})$ for $l \leq k$, and
- the static transformation of skin cell i with respect to the coordinate frame of joint k: ${}^i_k\mathbf{T}$,

where joint k directly actuates the limb where a skin cell is mounted on, see Figure 52.

The skin system is able to self-calibrate [102, 107] and automatically determines the static transformations ${}^i_k\mathbf{T}$ of skin cells i with respect to the joint k. This was proposed by Mittendorfer et al. since manually obtaining ${}^i_k\mathbf{T}$ for hundreds of skin cells is not feasible. The capability to self-calibrate is an essential element that was employed in this thesis to implement reactive contact control.

5.1.2.3 Skin Torque Computation – Skin Cell-wise Computation

The general descriptions of Sections 5.1.2.1 and 5.1.2.2 lead to a skin cell wise procedure that transforms the virtual external forces $F_{\mathrm{z},i}$ for each skin cell to its corresponding reactive contact motor command τ_i. The sum of all these contributions (Equation 5.2) result in the global motor command τ_{skin}. Algorithm 4 summarizes this procedure.

Algorithm 4 Skin cell wise skin joint torque calculation

1: **update** ${}^{l}_{0}\mathbf{T}(\mathbf{q}) \quad \forall l \in 1, \ldots, \mathrm{DOF}$
2: $\boldsymbol{\tau}_{\mathrm{skin}} := \mathbf{0}$
3: **for all** skin cells i **do**
4: **calculate** $F_{\mathrm{z},i}$ according to Algorithm 3
5: **if not** $F_{\mathrm{z},i} > 0$ **then**
6: **continue**
7: **end if**
8: **get** joint k for skin cell i
9: **extract** ${}^{k}_{0}\mathbf{R}$ **from** ${}^{k}_{0}\mathbf{T}(\mathbf{q})$
10: **extract** ${}_{k}\mathbf{z}_{i}$ **and** ${}_{\mathrm{k}}\mathbf{t}_{i}$ **from** ${}^{i}_{k}\mathbf{T}$
11: **calculate** ${}_{0}\mathbf{w}_{i}$ according to Equation (5.6)
12: ${}_{0}\mathbf{t}_{i} := {}^{k}_{0}\mathbf{T}(\mathbf{q})\ {}_{\mathrm{k}}\mathbf{t}_{i}$
13: **calculate** $\mathbf{J}^{T}_{i}(\mathbf{q})$
14: $\boldsymbol{\tau}_{i} := \mathbf{J}^{T}_{i}(\mathbf{q})\ {}_{0}\mathbf{w}_{i}$
15: $\boldsymbol{\tau}_{\mathrm{skin}} := \boldsymbol{\tau}_{\mathrm{skin}} + \boldsymbol{\tau}_{i}$
16: **end for**

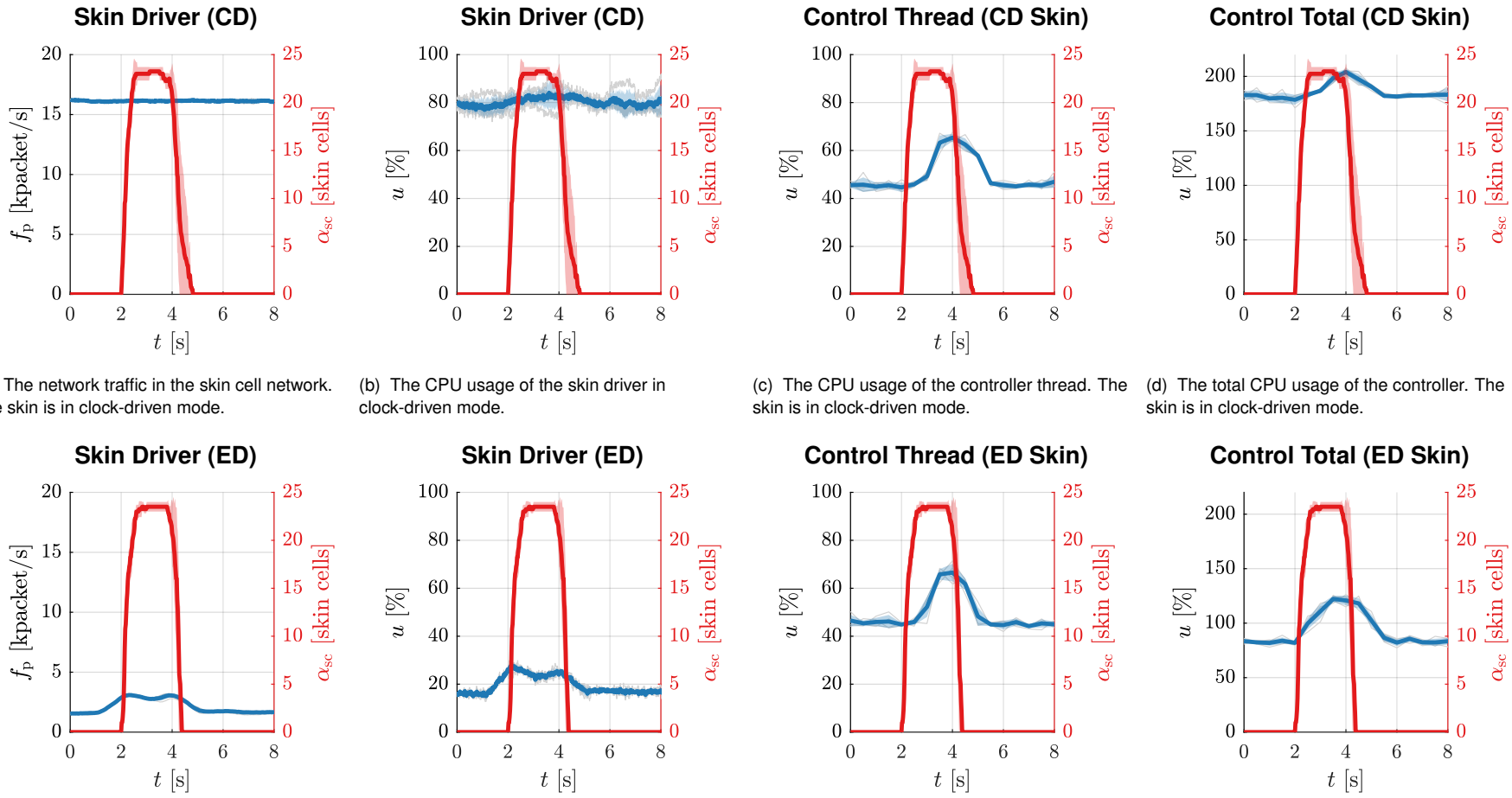
Overall, the calculation of joint torques $\boldsymbol{\tau}_{i}$ is computationally expensive. Especially, Lines 13 and 14 are expensive (the calculation of the Jacobian and the skin joint torques). However, the thresholding of the virtual force $F_{\mathrm{z},i}$ in Line 5 greatly reduces the computational costs. Rather than calculating the joint torque $\boldsymbol{\tau}_{i}$ for each skin cell i, in total n_{sc} times, the expensive computations are reduced to the number of active skin cells $\alpha_{\mathrm{sc}} \leq n_{\mathrm{sc}}$ [40, 17]. The computation of joint torques will be further improved in Sections 5.2 and 5.3.

5.1.3. Performance Evaluation of a Hybrid Event-Driven System

To validate the efficiency gain of hybrid event-driven systems (Section 4.2.2.2) in comparison to purely CDSs, the performance indicators presented in Section 4.3.5 are employed. Particularly, focusing on the packet rate and the CPU usage (Section 5.1.3.1), this section provides evidence for the hybrid event-driven system's effectiveness. Additionally, this section analyzes the controller response and latency to ensure comparable control performance. An efficiency gain in handling information is only valid when the control performance is equal or better to the clock-driven counterpart.

5.1.3.1 Packet Rate and CPU Usage

The measurements of the packet rate, the CPU usage of the skin driver, the CPU usage of the control thread, and the CPU usage of the control process are depicted in Figures 53, 54, and 55, respectively, for a sample rate of 62.5 Hz, 125 Hz, and 250 Hz. The top row of each of these figures presents the measurements of the clock-driven setup, and the bottom row the measurements of the hybrid setup.



(a) The network traffic in the skin cell network. The skin is in clock-driven mode.

(b) The CPU usage of the skin driver in clock-driven mode.

(c) The CPU usage of the controller thread. The skin is in clock-driven mode.

(d) The total CPU usage of the controller. The skin is in clock-driven mode.

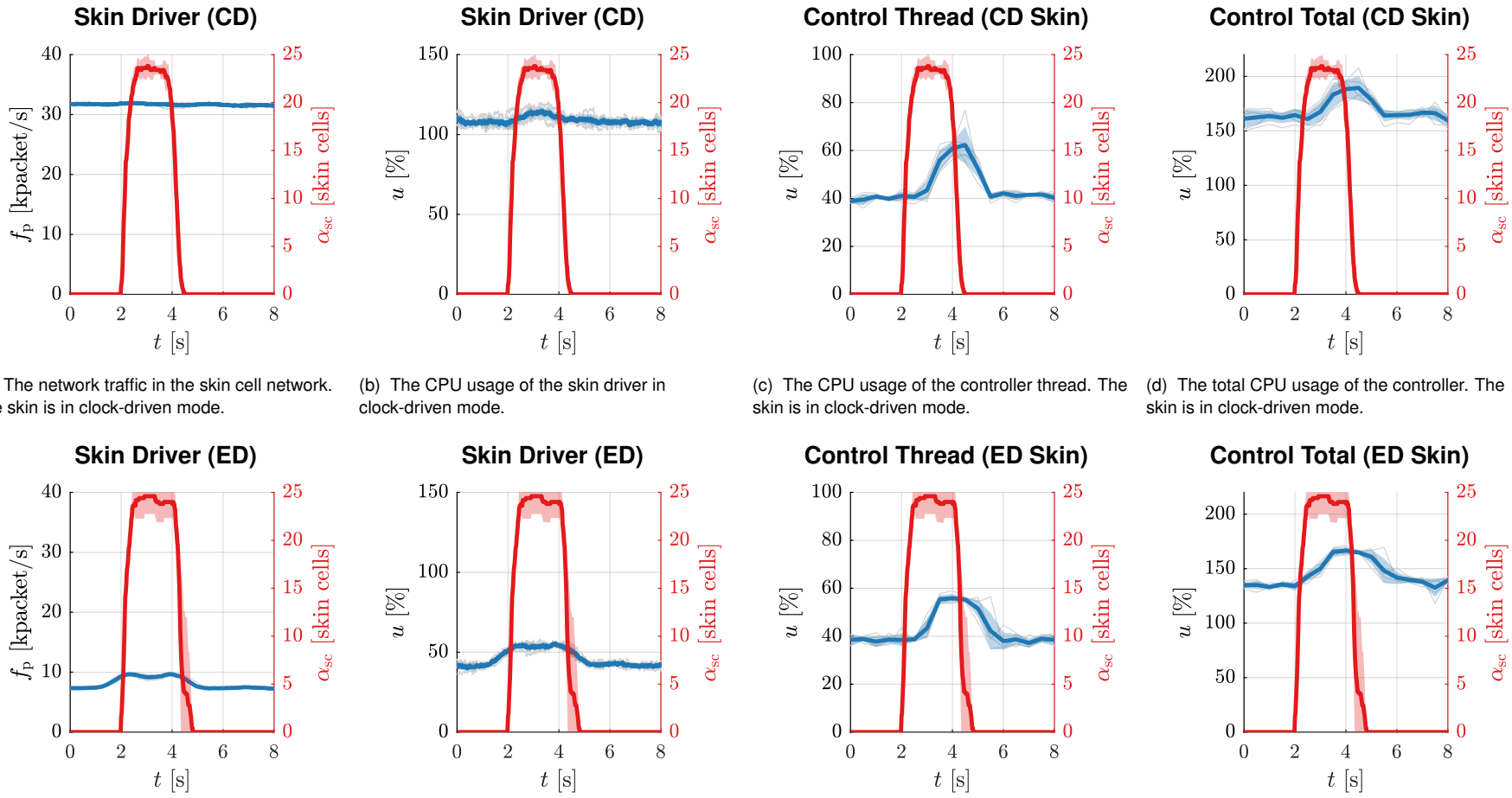
(e) The network traffic in the skin cell network. The skin is in event-driven mode.

(f) The CPU usage of the skin driver in event-driven mode.

(g) The CPU usage of the controller thread. The skin is in event-driven mode.

(h) The total CPU usage of the controller. The skin is in event-driven mode.

Figure 53 The CPU usage and network traffic for a skin sample rate of 62.5 Hz. Top row: The skin is in clock-driven (CD) mode. Bottom row: The skin is in event-driven (ED) mode. All figures contain, as a reference, the number of active cells α_{sc} computed by the skin driver. The solid lines represent the mean, the shadows the standard deviation. The gray lines represent the measurements. The plots show the clear advantage of using EDSs.



(a) The network traffic in the skin cell network. The skin is in clock-driven mode.

(b) The CPU usage of the skin driver in clock-driven mode.

(c) The CPU usage of the controller thread. The skin is in clock-driven mode.

(d) The total CPU usage of the controller. The skin is in clock-driven mode.

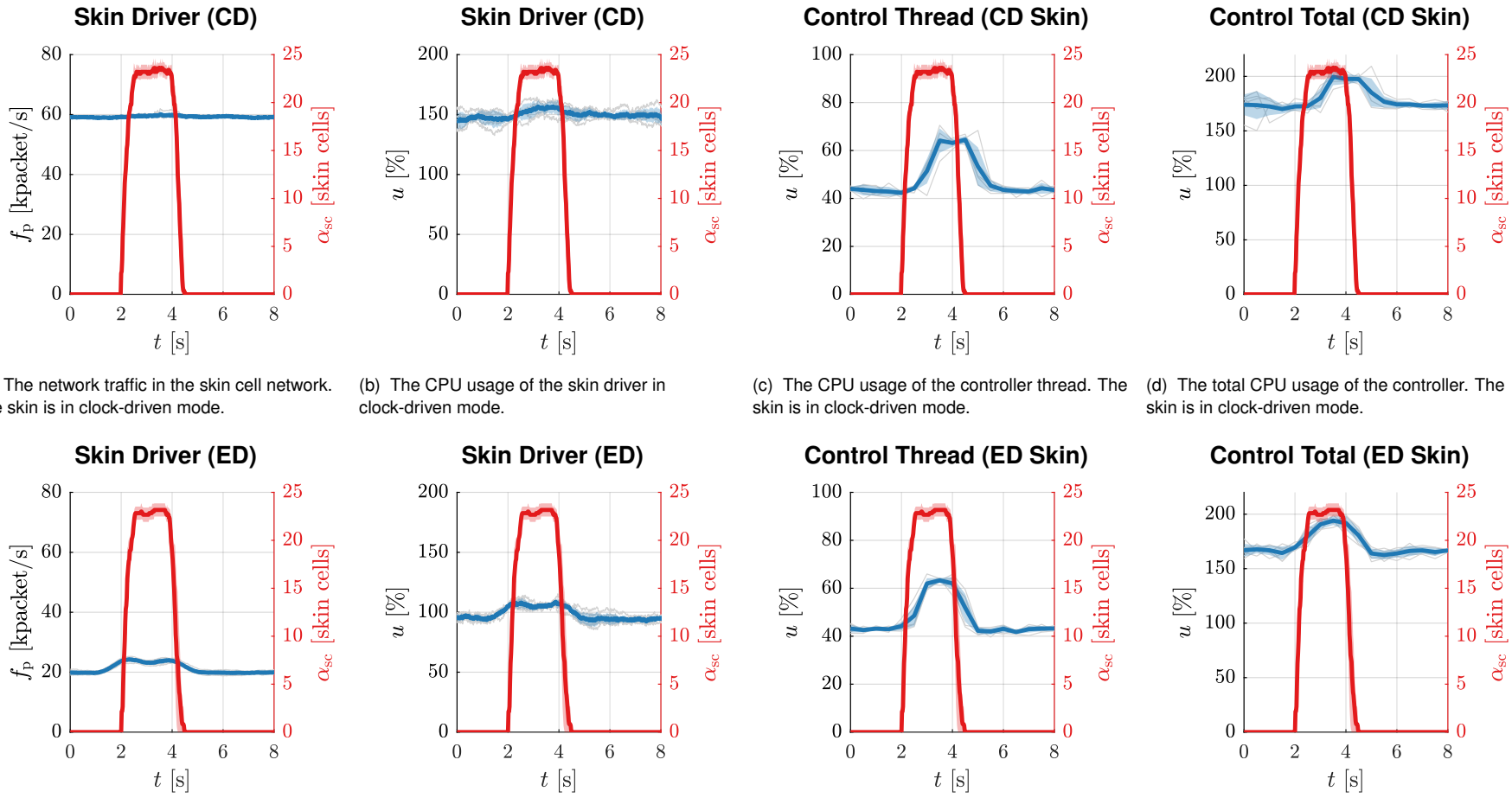
(e) The network traffic in the skin cell network. The skin is in event-driven mode.

(f) The CPU usage of the skin driver in event-driven mode.

(g) The CPU usage of the controller thread. The skin is in event-driven mode.

(h) The total CPU usage of the controller. The skin is in event-driven mode.

Figure 54 The CPU usage and network traffic for a skin sample rate of 125 Hz. Top row: The skin is in clock-driven (CD) mode. Bottom row: The skin is in event-driven (ED) mode. All figures contain, as a reference, the number of active cells α_{sc} computed by the skin driver. The solid lines represent the mean, the shadows the standard deviation. The gray lines represent the measurements. The plots show the clear advantage of using EDSs.



(a) The network traffic in the skin cell network. The skin is in clock-driven mode.

(b) The CPU usage of the skin driver in clock-driven mode.

(c) The CPU usage of the controller thread. The skin is in clock-driven mode.

(d) The total CPU usage of the controller. The skin is in clock-driven mode.

(e) The network traffic in the skin cell network. The skin is in event-driven mode.

(f) The CPU usage of the skin driver in event-driven mode.

(g) The CPU usage of the controller thread. The skin is in event-driven mode.

(h) The total CPU usage of the controller. The skin is in event-driven mode.

Figure 55 The CPU usage and network traffic for a skin sample rate of 250 Hz. Top row: The skin is in clock-driven (CD) mode. Bottom row: The skin is in event-driven (ED) mode. All figures contain, as a reference, the number of active cells α_{sc} computed by the skin driver. The solid lines represent the mean, the shadows the standard deviation. The gray lines represent the measurements. The plots show the clear advantage of using EDSs.

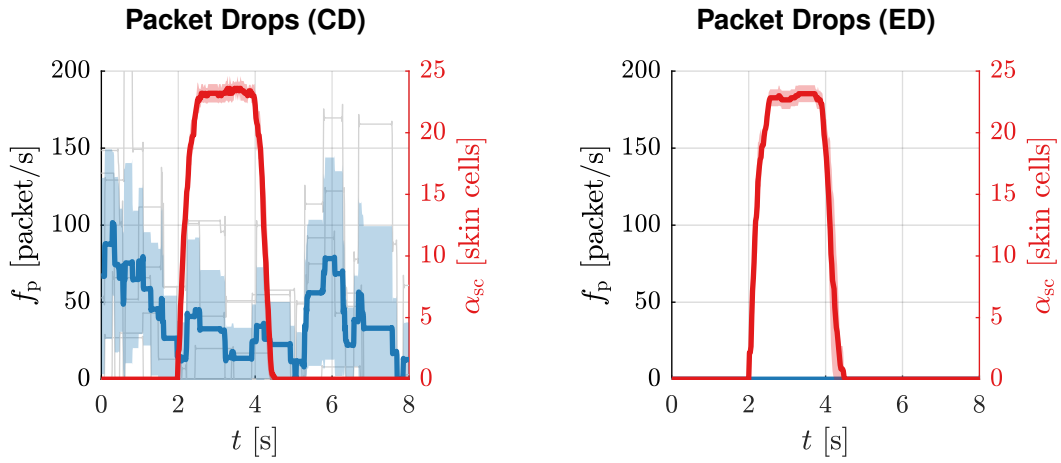
Table 21 summarizes the idle and peak packet rates for around 23 active skin cells when the skin operates in event-driven mode (Figures 53e, 54e, and 55e), and the constant packet rates when the skin operates in clock-driven mode (Figures 53a, 54a, and 55a). The significant reduction ratio of the packet rates in event-driven mode to around 33 % matches our expectations and the results presented in Chapter 4. Figures 53e, 54e, and 55e also depict that the change of the total number of active skin cells α_{sc} coincides with the peaks in the packet rate. This behavior is expected since the event rate, and thus the packet rate, correlates with the information rate of the tactile interaction (Sections 2.3.3, 3.1.3, and 4.1.1.2).

Packet Rate [packet/s]	ED Idle	ED Peak	Clock-Driven
62.5 Hz	1,700 (10.6 %)	3,100 (19.3 %)	16,100 (16,250 @ 260 skin cells[2])
125 Hz	7,300 (23 %)	9,700 (30 %)	31,700 (32,500 @ 260 skin cells)
250 Hz	19,700 (33.2 %)	24,200 (40.8 %)	59,300 (65,000 @ 260 skin cells)

Table 21 The measured packet rates of skin packets received by the computer system for a sampling rate of 62.5 Hz, 125 Hz, and 250 Hz. Columns two and three present the packet rates when the skin driver operates in event-driven (ED) mode. The idle packet rate represents the packet rate when the skin is not touched. The peak packet rate represents the highest packet rate measured during the tactile interaction. The ratios in the brackets relate these packets rates with respect to the measured packet rates when the skin driver is in clock-driven mode. The numbers in brackets after the packet rates for the clock-driven mode denote the expected packet rates in this mode. The small differences between measured and expected packet rates for 62.5 Hz and 125 Hz result from the small variances for the skin cells' sample rates. The packet drop is zero. The difference for 250 Hz partially results from packet drops (Figure 56).

The computer system only drops packets when the skin system operates in clock-driven mode with a sample rate of 250 Hz. In this setup, the system loses constantly around 50 packet/s, see Figure 56. In comparison to the total packet rate, the loss is around 0.08 %, which is still low. The system operates close to its saturation, nevertheless, the information loss will only have a minor impact on the performance of the system. The evaluation in Section 5.1.3.2 will show no significant impact of packet loss. Thus, for this scenario, the e-skin system can be used in both modes.

[2] The robot arm is covered with 253 skin cells (Section 5.1.1). However 260 skin cells are connected to the skin driver. These seven additional skin cells are not used in the control but their information is processed by the skin driver.



(a) The packet drop rate f_p of the system with the skin driver in clock-driven (CD) mode.

(b) The packet drop rate f_p of the system with the skin driver in event-driven (ED) mode.

Figure 56 The packet drop rate of the system. First, for the skin driver in clock-driven mode, then in event-driven mode. The sample rate of the e-skin is 250 Hz. The packet drop rate is zero for 62.5 Hz and 125 Hz. All figures contain, as a reference, the number of active cells α_{sc} computed by the skin driver (red). The solid lines (blue/red) represent the mean, the shadows (light blue/light red) the standard deviation. The gray lines represent the measurements of the different experimental trials.

Table 22 summarizes the idle and peak CPU usage of the skin driver in event-driven mode (Figures 53f, 54f, and 55f), and the constant CPU usage in clock-driven mode (Figures 53b, 54b, and 55b). Throughout all conditions, the CPU usage is significantly lower in event-driven mode. Similar to the packet rates, the peaks of the CPU usage in the event-driven mode coincide with the maximum change of the number of active skin cells. As expected (Sections 3.3.1 and 4.1.1.2) the EDS requests computation time on demand, and the CPU usage changes according to the number of skin packets/events.

Skin Driver CPU Usage [%]	ED Idle	ED Peak	Clock-Driven
62.5 Hz	17 (21.3 %)	28 (35 %)	80
125 Hz	43 (39.1 %)	55 (50 %)	110
250 Hz	95 (63.3 %)	110 (73.3 %)	150

Table 22 The CPU usage of the skin driver including the additional CPU usage to operate the ROS interface (Appendix C, Figure 82) for the connection to the controller ROS node. Columns two and three present the idle and peak CPU usage of the skin driver in event-driven (ED) mode. The system is idle when the skin is not touched (the number of active skin cells is zero). The ratios in the brackets are with respect to the measured CPU usages in clock-driven mode. Overall, the event-driven mode significantly reduces the computational load in the skin driver.

The CPU usage of the control thread, that is the thread that executes all control algorithms depicted in Figure 49, is shown in Figures 53c, 54c, and 55c for the purely clock-driven system, and in Figures 53g, 54g, and 55g for the hybrid event-driven system. As expected, the CPU usage is similar throughout all operation modes and sample rates. The control algorithm is exactly the same for all conditions. The sample rate of the skin, or respectively, the amount of information provided by the skin driver, has no impact on the CPU usage of the controller thread. Tactile information is processed and provided by other background threads

of the controller process. When the number of active skin cells is zero, the controller thread only processes proprioceptive information with a constant computational complexity. Then the CPU usage settles at around 45 %. The CPU usage increases with the number of active skin cells (Algorithm 4) and reaches a peak of around 65 %.

The CPU usage of the control process is the sum of the CPU usage of all its threads. This CPU usage thus also includes the CPU usage demanded by the background threads that provide tactile information to the controller thread. Table 23 summarizes the idle and the peak CPU usage of the controller process for all operation modes and sample rates (Figures 53d, 54d, and 55d for the clock-driven system, and Figures 53h, 54h, and 55h for the hybrid event-driven system). Overall, the hybrid event-driven system profits from the redundancy reduction in the update-driven information stream between the skin driver and the controller (Figure 50). However, the positive effect decreases for higher sampling rates. The CPU usage for 250 Hz is practically the same for both systems. Furthermore, the CPU usage does not increase significantly in the clock-driven system when doubling or quadrupling the sample rate. This missing increase indicates that the background threads handling the tactile information before passing it to the controller thread saturate. The CPU usage in the hybrid event-driven system increases for higher sampling rates. Nevertheless, the numbers indicate that the hybrid event-driven system is also close to saturation for a sampling rate of 250 Hz. It is expected that the saturation impacts the responsiveness of the controller, which will be further investigated in Section 5.1.3.2.

Controller CPU Usage [%]	**CDS Idle**	**CDS Peak**	**HS Idle**	**HS Peak**
62.5 Hz	180	200	82 (45.6 %)	122 (61.0 %)
125 Hz	164	190	135 (82.3 %)	166 (87.4 %)
250 Hz	174	200	166 (95.4 %)	194 (97.0 %)

Table 23 The CPU usage of the controller process, that is the sum of CPU usage of all its threads. Columns two and three present the idle and peak CPU usage for the Clock-Driven System (CDS), and the last two columns respectively for the hybrid event-driven system (HS). Overall, the hybrid event-driven system profits from the redundancy reduction in the update-driven information stream between the skin driver and the controller (Figure 50). The positive effect decreases for higher sampling rates. The CPU usage for 250 Hz is practically the same for both systems.

Tables 24, 25, and 26 respectively summarize the performance gain in total CPU usage for 62.5 Hz, 125 Hz, and 250 Hz. The gain is significant for 62.5 Hz and deteriorates to barely noticeable for higher frequencies. As discussed before, this effect is mainly caused by the saturation of the data handling in the controller.

CPU Usage @ 62.5 Hz	CD Idle	CD Peak	ED/H Idle	ED/H Peak
Skin Driver	80 %	80 %	17 %	28 %
Controller	180 %	200 %	82 %	122 %
Total	260 %	280 %	99 %	150 %
Ratio	100 %	100 %	**38 %**	**54 %**

Table 24 The CPU usage of the skin driver and the controller with the skin sampling at 62.5 Hz. CD and ED denote the clock-driven and the event-driven operation modes. H stands for the hybrid event-driven system and refers to the clock-driven the controller.

CPU Usage @ 125 Hz	CD Idle	CD Peak	ED/H Idle	ED/H Peak
Skin Driver	110 %	110 %	43 %	55 %
Controller	164 %	190 %	135 %	166 %
Total	274 %	300 %	178 %	221 %
Ratio	100 %	100 %	**65 %**	**74 %**

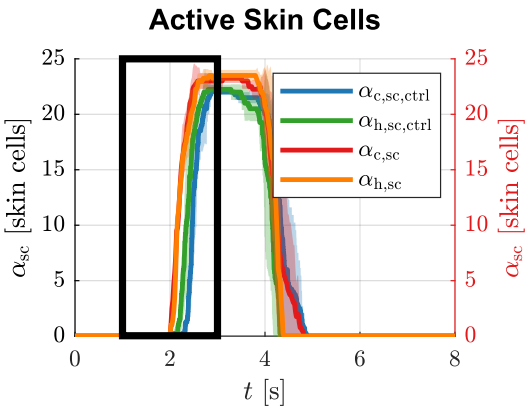
Table 25 The CPU usage of the skin driver and the controller with the skin sampling at 125 Hz. CD and ED denote the clock-driven and the event-driven operation modes. H stands for the hybrid event-driven system and refers to the clock-driven the controller.

CPU Usage @ 250 Hz	CD Idle	CD Peak	ED/H Idle	ED/H Peak
Skin Driver	150 %	150 %	95 %	110 %
Controller	174 %	200 %	166 %	194 %
Total	324 %	350 %	273 %	304 %
Ratio	100 %	100 %	**84 %**	**94 %**

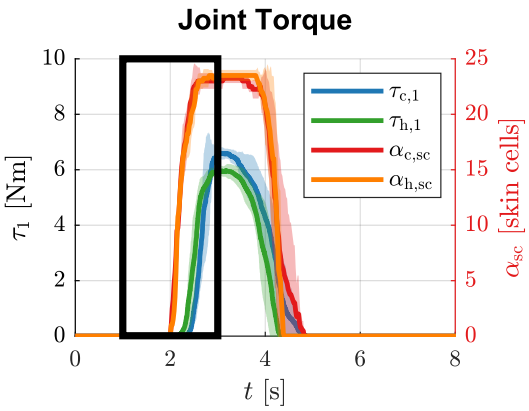
Table 26 The CPU usage of the skin driver and the controller with the skin sampling at 250 Hz. CD and ED denote the clock-driven and the event-driven operation modes. H stands for the hybrid event-driven system and refers to the clock-driven the controller.

5.1.3.2 Controller Response & System Latency

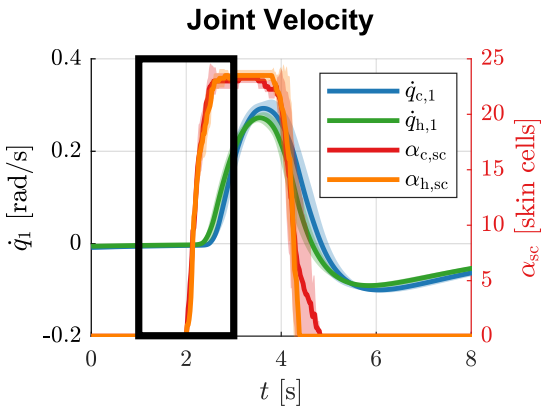
The measurements of the number of active skin cells, the joint torque, and the joint velocity are depicted in Figures 57, 58, and 59, respectively, for a sample rate of 62.5 Hz, 125 Hz, and 250 Hz. The top row in each of these figures presents and compares the measurements of the clock-driven and the hybrid event-driven system. The bottom row presents the respective close-ups in a smaller time interval.



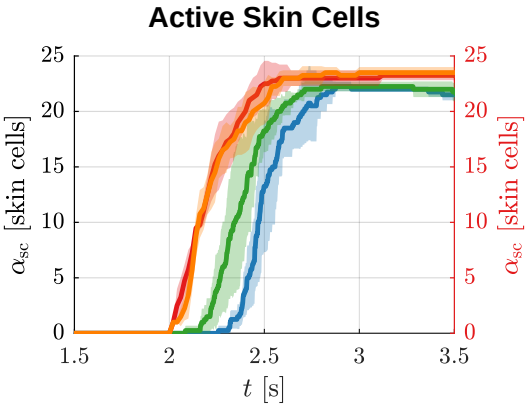
(a) The number of active skin cells $\alpha_{\mathrm{sc,ctrl}}$ computed by the controller.



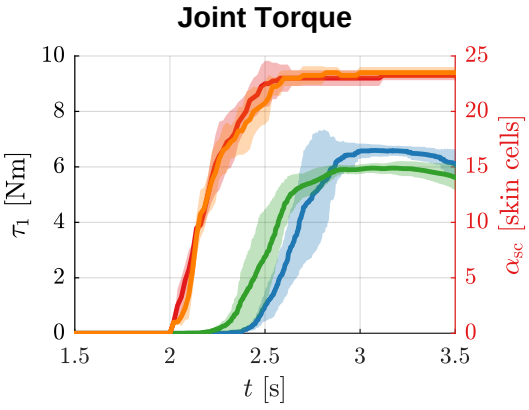
(b) The joint torque τ_1 of joint one (shoulder joint).



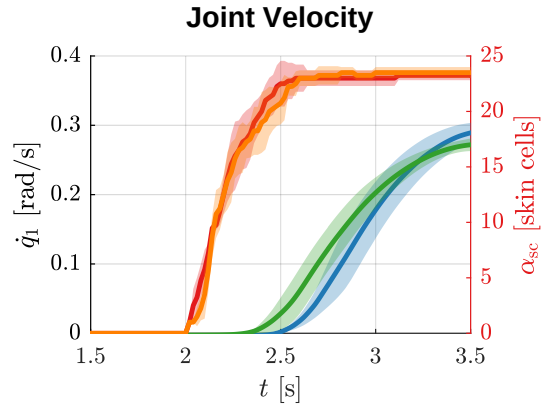
(c) The joint velocity $\dot{q}_1$ of joint one (shoulder joint).



(d) Zoom in with the number of active skin cells in the range $t \in [1.5,\ 3.5]$.

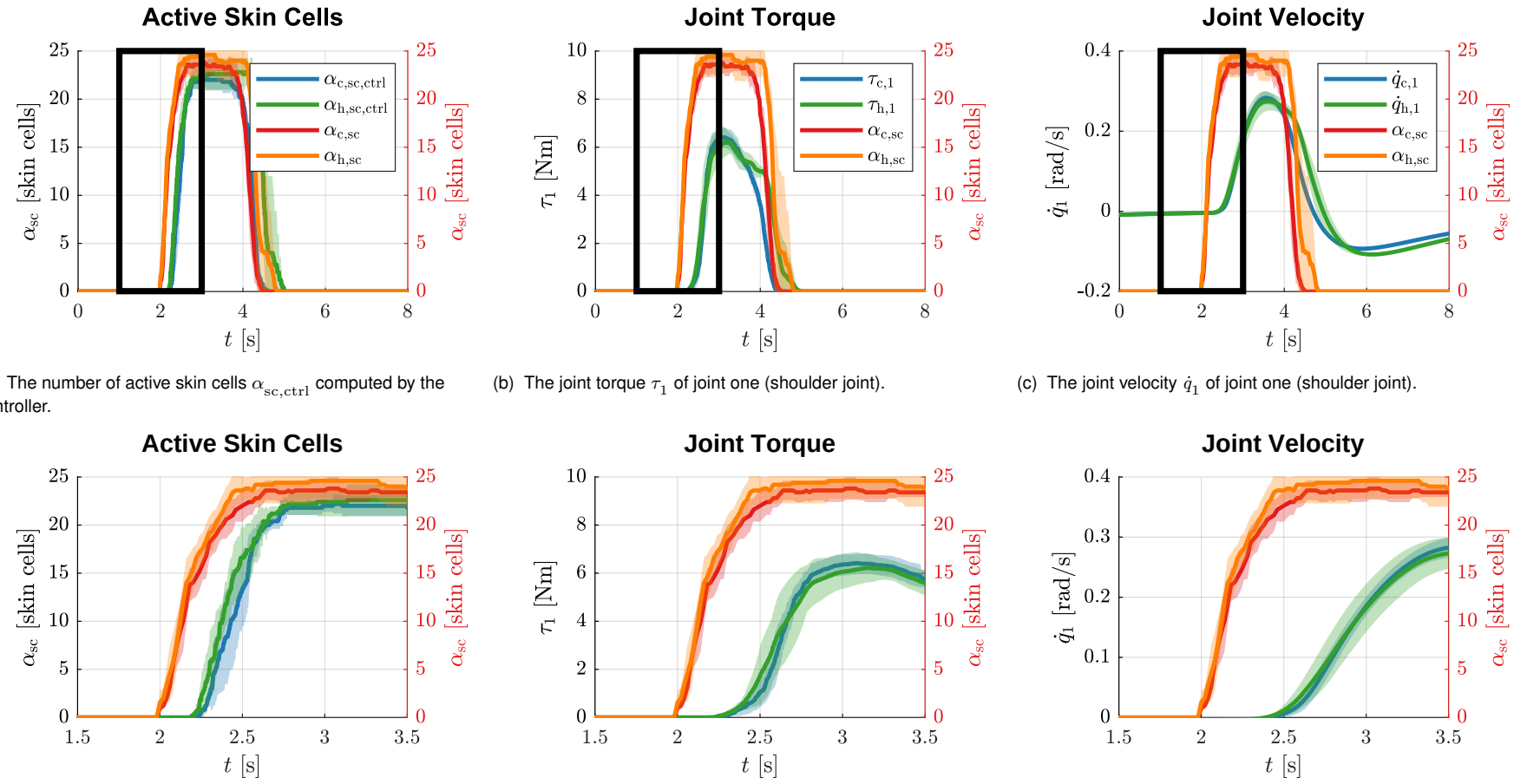


(e) Zoom in with the joint torque in the range $t \in [1.5,\ 3.5]$.



(f) Zoom in with the joint velocity in the range $t \in [1.5,\ 3.5]$.

Figure 57 The comparison of the number of active skin cells, the joint torque, and the resulting joint velocity for a skin sample rate of 62.5 Hz in the clock-driven and the hybrid event-driven system. The clock-driven system is denoted by the subscript c and the hybrid event-driven system respectively by h. Top row: Overview with the signals in the full range. Bottom row: Zoom in with the signals in the range $t \in [1.5,\ 3.5]$ s. All figures contain, as a reference, the number of active skin cells α_{sc} computed by the skin driver. The solid lines represent the mean, the shadows the standard deviation.



(a) The number of active skin cells $\alpha_{\mathrm{sc,ctrl}}$ computed by the controller.

(b) The joint torque τ_1 of joint one (shoulder joint).

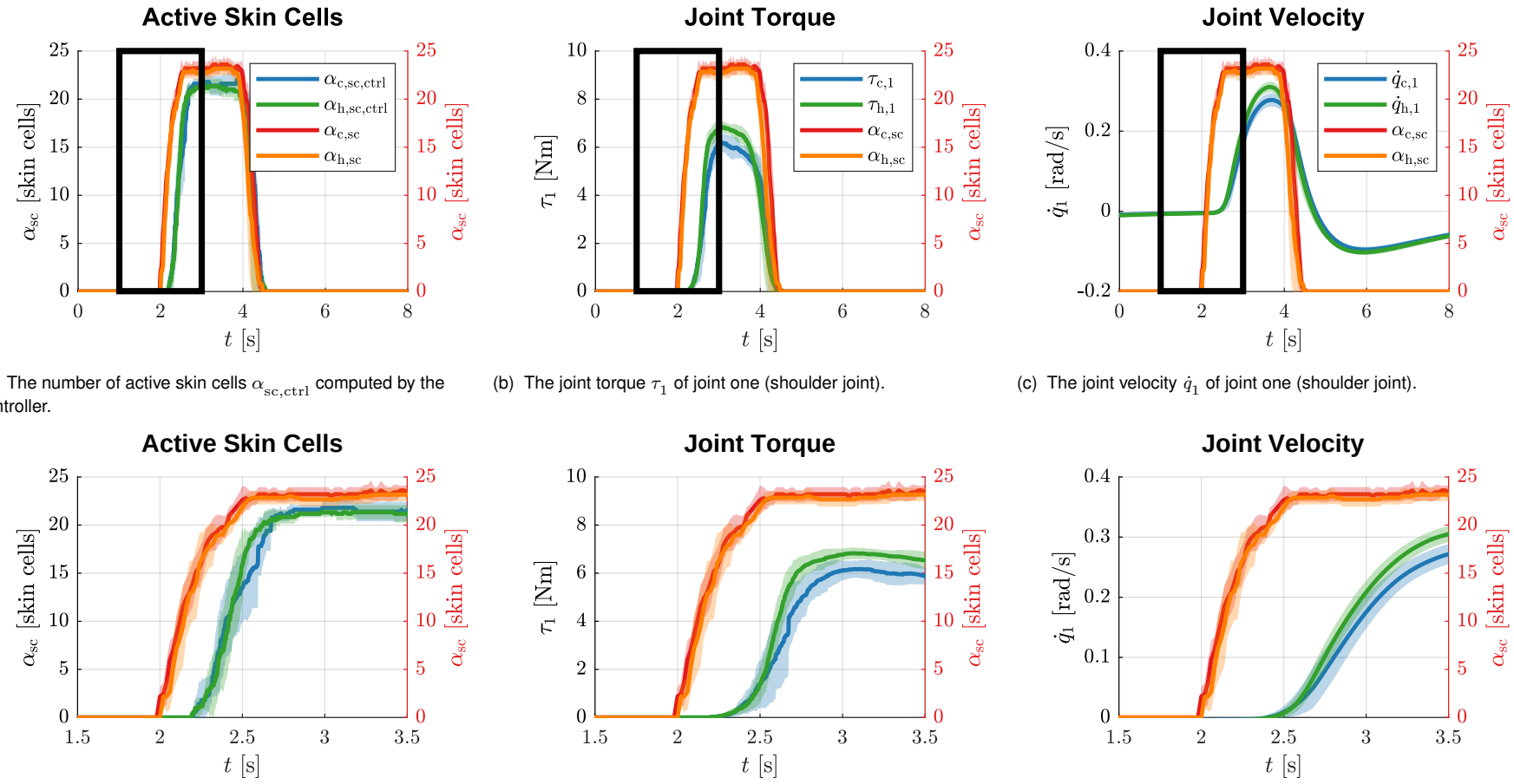
(c) The joint velocity $\dot{q}_1$ of joint one (shoulder joint).

(d) Zoom in with the number of active skin cells in the range $t \in [1.5,\ 3.5]$.

(e) Zoom in with the joint torque in the range $t \in [1.5,\ 3.5]$.

(f) Zoom in with the joint velocity in the range $t \in [1.5,\ 3.5]$.

Figure 58 The comparison of the number of active skin cells, the joint torque, and the resulting joint velocity for a skin sample rate of 125 Hz in the clock-driven and the hybrid event-driven system. The clock-driven system is denoted by the subscript c and the hybrid event-driven system respectively by h. Top row: Overview with the signals in the full range. Bottom row: Zoom in with the signals in the range $t \in [1.5,\ 3.5]$ s. All figures contain, as a reference, the number of active skin cells α_{sc} computed by the skin driver. The solid lines represent the mean, the shadows the standard deviation.



(a) The number of active skin cells $\alpha_{\mathrm{sc,ctrl}}$ computed by the controller.

(b) The joint torque τ_1 of joint one (shoulder joint).

(c) The joint velocity $\dot{q}_1$ of joint one (shoulder joint).

(d) Zoom in with the number of active skin cells in the range $t \in [1.5,\ 3.5]$.

(e) Zoom in with the joint torque in the range $t \in [1.5,\ 3.5]$.

(f) Zoom in with the joint velocity in the range $t \in [1.5,\ 3.5]$.

Figure 59 The comparison of the number of active skin cells, the joint torque, and the resulting joint velocity for a skin sample rate of 250 Hz in the clock-driven and the hybrid event-driven system. The clock-driven system is denoted by the subscript c and the hybrid event-driven system respectively by h. Top row: Overview with the signals in the full range. Bottom row: Zoom in with the signals in the range $t \in [1.5,\ 3.5]$ s. All figures contain, as a reference, the number of active skin cells α_{sc} computed by the skin driver. The solid lines represent the mean, the shadows the standard deviation.

Figures 57a, 58a, and 59a present the number of active skin cells α_{sc}. These numbers are either computed by the skin driver ($\alpha_{\mathrm{c,sc}}$ and $\alpha_{\mathrm{h,sc}}$) or by the controller ($\alpha_{\mathrm{c,sc,ctrl}}$ and $\alpha_{\mathrm{h,sc,ctrl}}$), in both cases, according to Algorithm 3. The subscript c denotes the clock-driven system and the subscript h respectively the hybrid event-driven system. The delay Δt_{α} between the flanks of α_{sc} and $\alpha_{\mathrm{sc,ctrl}}$ represents the delay of tactile information between the skin driver and the controller. Table 27 summarizes these delays for the different operation modes and sampling frequencies.

Delay [ms] @ 62.5 Hz	$\Delta t_{\alpha} = t_{\alpha_{\mathrm{sc,ctrl}}} - t_{\alpha_{\mathrm{sc}}}$	$\Delta t_{\mathrm{act,ctrl}} = t_{\dot{q}_1} - t_{\alpha_{\mathrm{sc,ctrl}}}$	$\Delta t_{\mathrm{act}} = t_{\dot{q}_1} - t_{\alpha_{\mathrm{sc}}}$
Clock-Driven System	330 (71)	501 (37)	831 (109)
Hybrid ED System	174 (130)	474 (95)	649 (225)

Delay [ms] @ 125 Hz	$\Delta t_{\alpha} = t_{\alpha_{\mathrm{sc,ctrl}}} - t_{\alpha_{\mathrm{sc}}}$	$\Delta t_{\mathrm{act,ctrl}} = t_{\dot{q}_1} - t_{\alpha_{\mathrm{sc,ctrl}}}$	$\Delta t_{\mathrm{act}} = t_{\dot{q}_1} - t_{\alpha_{\mathrm{sc}}}$
Clock-Driven System	247 (22)	536 (45)	782 (66)
Hybrid ED System	243 (52)	550 (112)	793 (164)

Delay [ms] @ 250 Hz	$\Delta t_{\alpha} = t_{\alpha_{\mathrm{sc,ctrl}}} - t_{\alpha_{\mathrm{sc}}}$	$\Delta t_{\mathrm{act,ctrl}} = t_{\dot{q}_1} - t_{\alpha_{\mathrm{sc,ctrl}}}$	$\Delta t_{\mathrm{act}} = t_{\dot{q}_1} - t_{\alpha_{\mathrm{sc}}}$
Clock-Driven System	244 (30)	536 (105)	780 (136)
Hybrid ED System	240 (77)	503 (54)	742 (131)

Table 27 The delays between the skin driver and the controller (Δt_{α}), between the active skin cell flank at the controller and the joint velocity flank ($\Delta t_{\mathrm{act,ctrl}}$), and between the skin driver and the controller reaction (Δt_{act}). All the respective flanks are depicted in Figures 57 to 59. The delays of the hybrid event-driven (ED) system are lower than in the clock-driven system for the sample rates 62.5 Hz and 250 Hz, and comparable for 125 Hz. The numbers in brackets denote the standard deviation of the presented results.

At first glance, the delays Δt_{α} between the skin driver and the controller seem to be, with a value of around 250 ms, rather large. These delays do not depend on the sampling frequency of the e-skin. Therefore, the saturation of the information handling system in the control process can not explain the absolute delay. The results of Section 5.1.3.1 only indicate saturation in the clock-driven system, but the delays are large for all operation modes and sampling rates. The reason for these large delays is the different active skin cell computation in the skin driver and the controller. The skin driver employs lower thresholds than the controller and thus its flanks start earlier. If there is indeed a constant delay between skin driver and controller then both, rising and falling flanks, would be shifted. This is clearly not the case in Figures 57a, 58a, and 59a. The falling flanks practically occur at the same time, thus indicating that the delay in the communication between the skin driver and the controller is practically zero.

In contrast to the absolute delays, the assessment of relative delays between different operation modes is more significant. The relative delays Δt_{α} are similar for the different operation modes and sample rates. The only exception is the hybrid event-driven system with the skin operating at 62.5 Hz. Under these conditions, the hybrid event-driven system shows a sig-

nificantly lower delay. Overall, the hybrid event-driven system shows a slightly smaller delay, and the combination of the event-driven e-skin system with a clock-driven controller has no negative effects on the system delay.

The joint torques and velocities are presented in Figures 57b, 58b, and 59b, and receptively in Figures 57c, 58c, and 59c. Throughout all operation modes and sample rates, the measurements prove a good similarity, ensuring the comparability between all operation modes and sample rates. All small differences in the shapes of the signals lie within the stochastic certainty and result from the uncertainties in the human robot interaction (Section 5.1.1). The delay $\Delta t_{\mathrm{act,ctrl}}$ between the active skin cell flank and the joint velocity flank is comparable for all experiments with a slight advantage for the hybrid event-driven system. The slow reaction of the robot is chosen by design with the user defined parameters of the virtual dynamic model. For capturing comparable measurements, slow robot reactions leading to deterministically controllable human robot interactions were targeted. When the robot is too fast for the human, the human would have to practice to keep up with the fast reactions of the robot. Nevertheless, the robot motion can be tuned to faster reactions and has proven its performance in other experimental setups [44, 29].

Here, the presented results prove that the hybrid event-driven system has no negative impacts on control performance. Actually, the hybrid system slightly improves the control performance with lower latencies that result in faster control reactions. The systems are comparable in all operation modes while hybrid event-driven systems significantly lower the demands on CPU usage and network traffic.

5.2. Contact-Driven Distributed Control Computation

This section presents a novel method to reduce the high computational demand in clock-driven reactive control algorithms that depend on large-area tactile feedback (Section 5.1). As presented in the previous section, reactive contact controllers transform tactile interaction sensed by the e-skin to joint torques, that is, to robot arm reactions. Therefore, control algorithms that feasibly scale with the tactile feedback are essential for realizing large-area interactions with efficient LASSs.

The importance of reactive torque control for contact-driven kinesthetic robot behaviors has first been pointed out by the works of [59, 92]. These works employ force/torque sensors to sense contacts. Later, to overcome the contact localization problem in multi-contact scenarios, reactive contact controllers using e-skin have been introduced in the works of [142, 50, 23, 116, 109, 40]. Reactive contact controllers processing tactile information of e-skin have a high demand for computational power that increases with the number of active skin cells (Algorithm 4, Section 5.1.2.3) and the Degrees Of Freedom (DOF) of the robot. To deal with the high computation load in systems with many DOF, the work of [141] introduced an efficient asynchronous algorithm for computing their forward kinematics.

To overcome the limited scalability of tactile feedback in clock-driven control, this thesis proposes to distribute the computation of skin joint torques (Algorithm 4) required for reactive contact control to the smart skin cells of the e-skin system. The proposed distribution immediately removes the expensive computation of skin joint torques from the control cycle, relaxing the limit on the maximum number of active skin cells before violating real-time constraints. To our best knowledge, the distributed computation of skin joint torques has not been fully addressed in other works.

The realization of the distributed skin joint torque computation is *contact-driven*, that is the joint torques are computed and forwarded to the central control system when the skin cells are *active* (Algorithm 3, Section 5.1.2.1).

This section is structured as follows. First, Section 5.2.1 introduces the experimental setup. Then, Section 5.2.2 details the realization of the distributed skin joint torque computation. Section 5.2.3 presents our experimental results.

The work presented in Section 5.2 was in part published in:
Bergner, F., Dean-Leon, E., Cheng, G., “Efficient Distributed Torque Computation for Large Scale Robot Skin”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, 2018, pp. 1593–1599.

5.2.1. Experimental Setup – Distributed Control Computation

The experimental platform is the same as in Sections 4.4.1, 4.5.1, and 5.1.1. The UR5 robot arm is covered with 253 skin cells, see Figure 60. The e-skin is integrated in a perception-action loop with a clock-driven controller, similar to the setup presented in Section 5.1.1, Figure 49. But in contrast to the previous setup, the reactive contact control presented in Algorithm 4 is now distributedly implemented in the skin cells. The local microcontrollers of the skin cells perform the fusion of tactile and proprioceptive information to motor commands τ_i. Together with the multi-modal tactile information, the skin cells of this setup also send their computed control contribution τ_i, that the controller process on the computer eventually fuses to the final commanded joint torque τ_{cmd}.

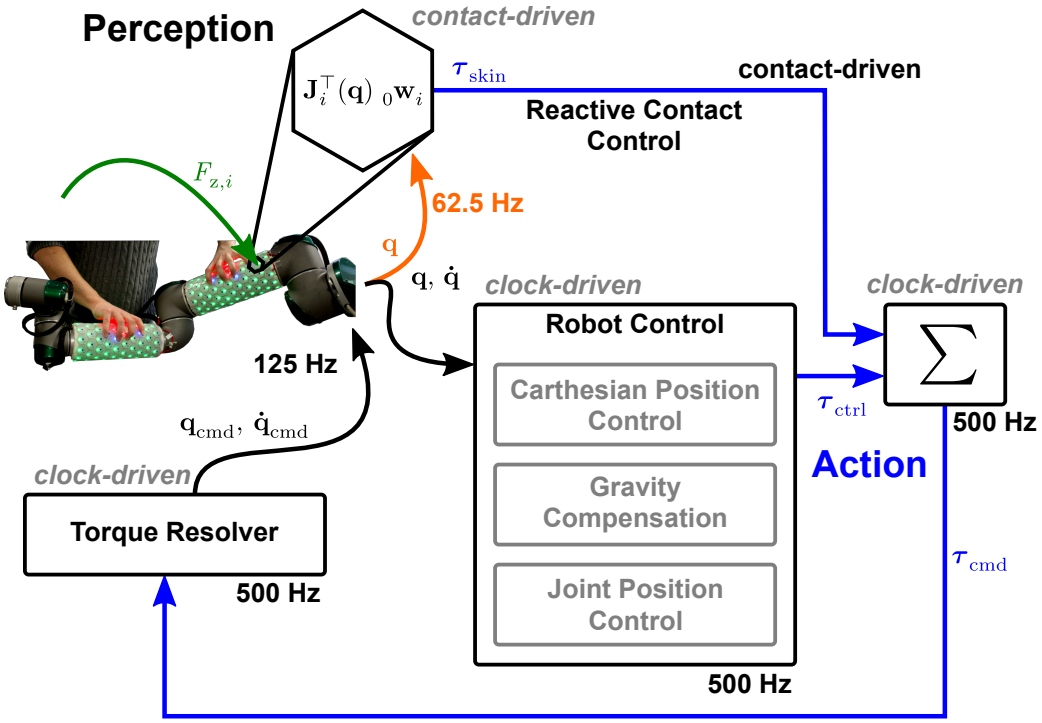


Figure 60 The reactive contact controller that fuses the tactile $F_{z,i}$ with proprioceptive information $\mathbf{q}$ to motor commands τ_i. This controller is distributedly implemented in the skin cells rather than in the controller process (Figure 49).

The experiments performed to validate the effectiveness of the approach does not need to be repeatable. Instead, the joint torques $\tau_{\text{skin,dstb}}$ computed by the e-skin on-line can be directly compared with the reference joint torques τ_{skin} computed in the controller process. For the reference computation of τ_{skin}, the reactive contact controller presented in Section 5.1 is used.

The controller is configured to realize a kinesthetic behavior in the joint space. Therefore, it combines the gravity compensation calculated on the computer with the reactive contact control realized in the skin cells. The robot arm is touched by the human and moves away from the detected contacts. Thus, the robot arm can be kinesthetically guided to desired

positions. The controller sends joint positions $\mathbf{q}$ to the skin cells with an update rate of $f_{\mathrm{q}} = 62.5\,\mathrm{Hz}$ and receives the sum of all skin joint torques $\boldsymbol{\tau}_{\mathrm{skin,dstb}}$. The computation of the skin joint torques is contact-driven and the results of active skin cells are sent with the same rate. The reactive contact controller supports two modes:

1. Use $\boldsymbol{\tau}_{\mathrm{skin,dstb}}$ and compute $\boldsymbol{\tau}_{\mathrm{skin}}$ in parallel, and
2. Use $\boldsymbol{\tau}_{\mathrm{skin,dstb}}$.

The first mode was used to compare the joint torque $\boldsymbol{\tau}_{\mathrm{skin,dstb}}$ computed by the e-skin with the reference τ_{skin} calculated in the computer. The second mode is used to measure and evaluate the reduction of computation time in the control loop when $\boldsymbol{\tau}_{\mathrm{skin}}$ is not computed. In both modes, $\boldsymbol{\tau}_{\mathrm{skin,dstb}}$ is used to control the robot. During the experiments, a touch on the robot arm would kinesthetically move it in the joint space.

5.2.2. Distributed Skin Joint Torque Computation

The following sections explain step by step the distribution and realization of Algorithm 4 in the microcontrollers of the skin cells. The algorithm is adjusted to increase computation efficiency and meet the requirements of the microcontrollers' limited computation capabilities.

5.2.2.1 The General Computation Principle

Each skin cell exploits its microcontroller and computes its own skin joint torque contribution $\boldsymbol{\tau}_i$. This decentralized computation of skin joint torques $\boldsymbol{\tau}_i$ on skin cells i requires the following steps:

1. Provide the static transformation ${}^{i}_{k}\mathbf{T}$ between the skin cell i and the joint k
2. Receive the most recent joint positions $\mathbf{q} \in \mathbb{R}^{\mathrm{DOF}}$
3. Compute the forward kinematics to the skin cell i, thus ${}^{l}_{0}\mathbf{T}(\mathbf{q})\ \forall\, l \leq k$
4. Compute the virtual force $F_{\mathrm{z},i}$ (Algorithm 3)
5. Compute the components $\tau_{l,i}$ of $\boldsymbol{\tau}_i$ (Equation (5.10))
6. Send the joint torques $\boldsymbol{\tau}_i$ from the skin cells to the reactive contact controller.

This distributed computation of skin joint torques was realized for the UR5 robot arm with six DOF (Section 5.2.1). The computational power of the skin cell's microcontrollers is very limited (16 Mega Instructions Per Second (MIPS), 16 bit architecture, no Floating Point Unit (FPU)), thus, the optimization of the implementation for this specific robot arm was needed. Nevertheless, this optimization does not limit the generality of the approach. The same or similar optimizations apply to other robot arms.

5.2.2.2 Fixed Point Arithmetic

The skin cells' microcontrollers use a 16 bit architecture without an FPU. Using floating-point arithmetic on this microcontroller is infeasible because even simple floating-point operations, such as sums and products, translate to hundreds of assembly instructions causing large delays. In order to achieve fast computations on this microcontroller, all arithmetic operations are realized with fixed point numbers with 32 bit and 16 bit precision. Fixed point numbers reserve one bit to realize negative numbers with the two's complement and employ the remaining bits for integer and fraction. The Q notation for fixed point numbers is used. A $Q1.14$ number uses bit 15 as sign bit, bit 14 for the integer part, and bits 13 to 0 for the fractional part. A $Q1.14$ number can represent numbers in $[-2, 2[$ with a constant precision of $2^{-14} = 6.1035 \cdot 10^{-5}$. Additions and multiplications of fixed point numbers map to integer additions and multiplications. To add fixed point numbers, both numbers must have the same number of fractional bits. While additions preserve the number of fractional bits, multiplications change it. The number of fractional bits can easily be adjusted by using shift operations. As long as arithmetic operations have a limited dynamic range, fixed point arithmetic can provide good accuracy and enable fast operations on architectures without an FPU. Additions and multiplications of fixed point numbers only require few assembly instructions.

5.2.2.3 Storing the Skin Cell Poses into the Skin Cells

The distributed computation of the skin joint torque $\boldsymbol{\tau}_i$ of a skin cell i requires the static transformation ${}^{i}_{k}\mathbf{T}$ (Section 5.2.2.1, Algorithm 4). The skin cell firmware is extended (Section 4.3.1.3) with an interface to store the transformation ${}^{i}_{k}\mathbf{T}$ with $Q1.14$ into the microcontroller's non-volatile flash memory. Since the locations of all skin cells are different, the transformations ${}^{i}_{k}\mathbf{T}$ are also different for all skin cells i. The skin cells also store the ID of the robot joint k, which is the robot joint directly connected to the limb where skin cell i is mounted (Figure 52). The transformation ${}^{i}_{k}\mathbf{T}$, which is the static transformation from skin cell i to joint k, is determined using a 3D self-calibration algorithm given by [102, 44, 29]. This process is done only once and the values are stored in the skin cell until a re-calibration process is needed, for instance, when a skin cell changes its location.

5.2.2.4 Fast Trigonometric Functions

Fast and precise sine and cosine functions are essential for updating the forward kinematics fast and with sufficient precision. The homogeneous transformation matrix ${}_{l-1}^{l}\mathbf{T}(q_l)$ of joint l with respect to joint $l-1$ contains a rotation matrix ${}_{l-1}^{l}\mathbf{R}(q_l) \in \mathbb{R}^{3\times3}$ and a translation vector ${}_{l-1}\mathbf{t}_l \in \mathbb{R}^3$:

$$ {}_{l-1}^{l}\mathbf{T}(q_l) = \begin{bmatrix} {}_{l-1}^{l}\mathbf{R}(q_l) & {}_{l-1}\mathbf{t}_l \\ \mathbf{0}^\top & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \tag{5.13} $$

All the transformations ${}_{l-1}^{l}\mathbf{T}(q_l)$ between joints of the kinematic chain contain $\sin(q_l) = s_l$ and $\cos(q_l) = c_l$ in ${}_{l-1}^{l}\mathbf{R}$ [3]. In the worst case, for a skin cell i actuated by joint $k = 6$, the microcontroller has to compute six different sines and cosines.

Sine/cosine functions can be approximated by polynomials, Look-Up-Tables (LUTs) with interpolation, Taylor series, or the Coordinate Rotation Digital Computer (CORDIC) algorithm [140]. LUTs require memory and are slow since memory access is slow. Taylor series approximations are inaccurate since an infinite series is truncated and the special properties of the sine/cosine functions are lost. The CORDIC algorithm uses only shift and addition operations but is slow in architectures where shift, addition, and multiplication operations have the same cost since the algorithm only converges linearly.

Thus, to get fast and precise fixed point results for sine/cosine operations, the approximation of the $\sin(x)$ is done with a 5-th order polynomial in the interval $x \in [-\frac{\pi}{2}, \frac{\pi}{2}[$. Simplifying the computation and notation, we transform coordinates and substitute x by z:

$$ z := \frac{x}{0.5\ \pi}. \tag{5.14} $$

The transformation normalizes angular input arguments x in radians. The sine function $\sin(z)$ has a period of $z = 2\ k$:

$$ \sin(z) = \sin(z \pm 2\ k) \qquad \forall k \in \mathbb{Z}. \tag{5.15} $$

The sine function is an odd function, thus all parameters of the 5-th order polynomial connected to even powers are zero:

$$ p_5(z) = a\,z - b\,z^3 + c\,z^5 \tag{5.16} $$

$$ \dot{p}_5(z) = a - 3\,b\,z^2 + 5\,c\,z^4 \tag{5.17} $$

The appropriate parameters are determined with three conditionals that define the behavior of the polynomial on its boundaries and preserve the unique properties of a sine function:

$$ p_5(z=1) \stackrel{!}{=} 1 \qquad \dot{p}_5(z=1) \stackrel{!}{=} 0 \qquad \dot{p}_5(z=0) \stackrel{!}{=} \frac{\pi}{2} \tag{5.18} $$

where the symbol $\stackrel{!}{=}$ denotes the condition and means that the left side has to be equal to the right. Solving this equation system with three unknown parameters and three conditions

[3] By definition ${}_{l-1}^{l}\mathbf{R}$ are relative orientations represented by basic rotations around the axes $\mathbf{x}$, $\mathbf{y}$, or $\mathbf{z}$. Therefore, each ${}_{l-1}^{l}\mathbf{R}$ contains one sine and one cosine.

results into the following parameters:

$$a = \frac{\pi}{2}, \qquad b = \pi - \frac{5}{2}, \qquad \text{and } c = \frac{\pi}{2} - \frac{3}{2}. \tag{5.19}$$

The simplified approximation of $\sin(z)$ for $z \in [-1, 1[$ is thus:

$$\begin{aligned} \sin(z) &\approx \sin_{5,p}(z) \qquad \forall z \in [-1, 1[\\ \sin_{5,p}(z) &= 0.5z \left(\pi - z^2 \left[(2\pi - 5) - z^2 (\pi - 3)\right]\right). \end{aligned} \tag{5.20}$$

The sine function is a periodic function. Thus, the approximation of Equation (5.20) can be extended for general $z \in \mathbb{R}$. Changing the fixed point precision of z from $Qm.f$ to $Q1.14$ or respectively to $Q1.30$, maps, in a modulo-4-like operation, all $z \in \mathbb{R}$ to the interval $z \in [-2, 2[$. For this interval, the sine function can be approximated in the following way:

$$\sin_5(z) = \begin{cases} \sin_{5,p}(2 - z) & \textbf{if } z \in [-2, -1[\\ \sin_{5,p}(z) & \textbf{if } z \in [-1, 1[\\ \sin_{5,p}(2 - z) & \textbf{if } z \in [1, 2[\end{cases}. \tag{5.21}$$

This approximation of the sine function is fast and only uses 20 assembly instructions. Its realization on the microcontroller takes fixed point numbers of $Q1.14$ as argument and returns fixed point numbers in $Q1.14$. The approximation of the cosine is then:

$$\cos_5(z) = \sin_5(z + 1). \tag{5.22}$$

The mean error for $y = \sin_5(z)$ with z and y in $Q3.12$ is $6.7 \cdot 10^{-4}$ which is acceptably low for the application domain.

5.2.2.5 Fast Forward Kinematics

The forward kinematics for a skin cell i actuated by joint k computes as follows:

$${}^{k}_{0}\mathbf{T}(q_1, q_2, \ldots, q_k) = {}^{1}_{0}\mathbf{T}(q_1)\, {}^{2}_{1}\mathbf{T}(q_2) \cdots {}^{k}_{k-1}\mathbf{T}(q_k). \tag{5.23}$$

Multiplying two 4×4 matrices $\mathbf{AB} = \mathbf{C}$ results in:

$$c_{ij} = \sum_{k=1}^{4} a_{ik}\, b_{kj}, \tag{5.24}$$

an operation that has an arithmetic complexity of 64 `mul` $+ 48$ `add`. In the worst case, for $k = 6$, the arithmetic complexity of updating the forward kinematics is five times a 4×4 matrix multiplication resulting in 320 `mul` $+ 240$ `add`. Only computing the rotation matrix ${}^{k}_{0}\mathbf{R}$, the arithmetic complexity reduces to five times a 3×3 matrix multiplication, which is

135 `mul` $+ 90$ `add`. However, only computing ${}^{k}_{0}\mathbf{R}$ is not sufficient to compute the skin joint torque $\boldsymbol{\tau}_i$. Equation (5.10) also requires the projections $\mathbf{p}_{i,l-1,0}$, that is the projection of ${}_0\mathbf{t}_i$ over ${}_0\mathbf{t}_{l-1}$ with respect to the base coordinate frame 0:

$$\mathbf{p}_{i,l-1,0} = {}_0\mathbf{t}_i - {}_0\mathbf{t}_{l-1}. \tag{5.25}$$

The first index i denotes the coordinate frame of ${}_0\mathbf{t}_i$, the second index $l-1$ the coordinate frame of ${}_0\mathbf{t}_{l-1}$, and the last index the common base coordinate frame 0.

At first glance, this requires the computation of ${}^{i}_{0}\mathbf{T}$ which induces one additional 4×4 matrix multiplication resulting in 384 `mul` $+ 288$ `add`. However, the projection $\mathbf{p}_{i,l-1,0}$ can be computed much more efficiently by

$$\begin{aligned}
\mathbf{p}_{i,l-1,0} &= {}^{l-1}_{0}\mathbf{R}\ {}_{l-1}\mathbf{t}_i \\
&= {}^{l-1}_{0}\mathbf{R}\left({}^{k}_{l-1}\mathbf{R}\ {}_{k}\mathbf{t}_i + {}_{l-1}\mathbf{t}_k\right) \\
&= {}^{k}_{0}\mathbf{R}\ {}_{k}\mathbf{t}_i + {}^{l-1}_{0}\mathbf{R}\ {}_{l-1}\mathbf{t}_k \\
&= \mathbf{p}_{i,k,0} + \mathbf{p}_{k,l-1,0}.
\end{aligned} \tag{5.26}$$

The projection $\boldsymbol{p}_{i,k,0}$ is needed for the computation of all the components $\tau_{l,i}$ of $\boldsymbol{\tau}_i$ and the projection $\boldsymbol{p}_{k,l-1,0}$ can be computed by

$$\mathbf{p}_{k,l-1,0} = \sum_{m=l-1}^{k-1} {}^{m}_{0}\mathbf{R}\ {}_{m}\mathbf{t}_{m+1} \qquad \forall l \leq k. \tag{5.27}$$

All the required rotation matrices ${}^{m}_{0}\mathbf{R}$ are determined when computing ${}^{k}_{0}\mathbf{R}$. In the worst case for $k = 6$, the computation of $\mathbf{p}_{k,l-1,0}$ requires six times a matrix-vector multiplication (9 `mul`$+6$ `add`) and five times a vector addition (0 `mul`$+3$ `add`). This results in 54 `mul`$+51$ `add`. In summary, the computation of

$$\begin{aligned}
{}^{k}_{0}\mathbf{R} &\rightarrow 135\ \texttt{mul} + 90\ \texttt{add} \\
\mathbf{p}_{i,0,0} &\rightarrow 63\ \texttt{mul} + 78\ \texttt{add} \\
{}^{k}_{0}\mathbf{R}\,{}_{k}\mathbf{z}_i &\rightarrow 9\ \texttt{mul} + 6\ \texttt{add}
\end{aligned} \tag{5.28}$$

results in 207 `mul`$+174$ `add`. In comparison to computing ${}^{i}_{0}\mathbf{T}$ which needs 384 `mul`$+288$ `add`, the exploitation of the projection properties reduce the number of multiplications by 48 % and the number of additions by 40 %.

The number of multiplications and additions in the final realization on the microcontroller can be further reduced by reusing sums and products in the different processing steps. The kinematics for $k = 6$ using the formulas of Equation (5.28) results in 66 `mul`$+26$ `add`, reducing the final number of multiplications by 80 % and the number of additions by 85 %.

5.2.2.6 Joint Torque Computation and Propagation

The skin cells receive the joint positions of the robot $\mathbf{q} \in \mathbb{R}^6$ as a broadcast message with a constant update rate f_{q}. $\mathbf{q}$ has six components q_i, $i = 1, \ldots, 6$ which are trimmed and transformed into the z representation (Equation (5.14)), that is

$$\mathbf{q} = (q_i) \in \mathbb{R}^6 \qquad \text{with } q_i \in [-2, 2[. \tag{5.29}$$

These six joint positions q_i are encoded in $Q1.14$ and fit into one single skin packet. The ability to use broadcast messages and to pack $\mathbf{q}$ into one packet limits the transmission overhead in the down-link (computer-to-skin cell) enabling high update rates f_{q}.

When a skin cell i receives a new $\mathbf{q}$, it immediately starts to update $J_{l,i}(\mathbf{q})$ using Equations (5.10), (5.26), and (5.27). The realization of these equations on the skin cell is optimized such that only what is necessary is computed and, if possible, intermediate results are reused. In the worst case, for $k = 6$, these computations enfold six computations of $J_{l,i}(\mathbf{q})$ with a total arithmetic complexity of

$$(J_{l,i}) \rightarrow 103\ \texttt{mul} + 57\ \texttt{add} + 6\ \texttt{sin} + 6\ \texttt{cos} \tag{5.30}$$

and require around 1096 assembly instructions. Thus, with 16 MIPS, updating $(J_{l,i}) \in \mathbb{R}^6$ takes in worst case 68.5 µs.

The skin cells compute the virtual force $F_{\mathrm{z},i}$ implementing Algorithm 3. The final computation of $\boldsymbol{\tau}_i = (J_{l,i})\, F_{\mathrm{z},i}$ is only performed when the skin cell is active. The skin joint torque $\boldsymbol{\tau}_i \in \mathbb{R}^6$ of a skin cell i uses a fixed point precision of $Q3.12$ and can be packed into one single skin packet. The skin joint torque $\boldsymbol{\tau}_i$ is only sent to the computer (up-link) when the skin cell is active. This limits the transmission overhead in the up-link.

Joint torques $\boldsymbol{\tau}_i$ do not have a frame of reference and can be added up as shown in Equation (5.2). To exploit this important property, each skin cell i adds the skin joint torques $\boldsymbol{\tau}_{\mathrm{skin},n}$ of its neighbors to its own skin joint torque $\boldsymbol{\tau}_i$ before passing the result $\boldsymbol{\tau}_{\mathrm{skin},i}$ to the next skin cell as depicted in Figure 61, that is

$$\boldsymbol{\tau}_{\mathrm{skin},i} = \boldsymbol{\tau}_i + \sum_n \boldsymbol{\tau}_{\mathrm{skin},n} \in \mathbb{R}^6$$
$$n \in \{\text{neighbors of skin cell } i\}. \tag{5.31}$$

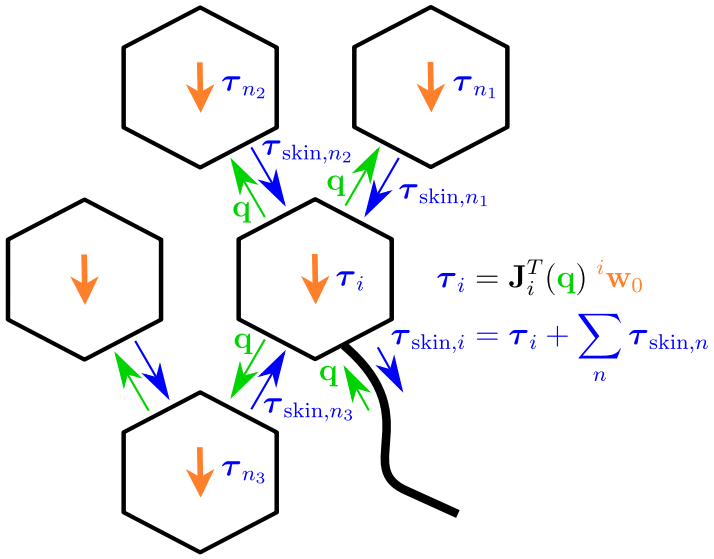


Figure 61 Propagating the sums of skin joint torques.

In order to allow for the higher numbers of the sums, the fixed point precision for $\boldsymbol{\tau}_{\text{skin},i}$ was reduced to $Q7.8$. Naturally, the propagation and summing of joint torques reduce the transmission rate in the up-link further at the cost of a slight decrease of precision in the final skin joint torque $\boldsymbol{\tau}_{\text{skin}}$.

The presented efficient approach for computing skin joint torques demonstrated the feasibility to distribute these computations to the skin cell's microcontrollers with very limited computational power. While the presentation focused on the optimal realization for the given hardware, the presented principles and optimizations are not limited to this specific hardware. The efficient approximation of sines and cosines, the optimized computation of the kinematics and joint torques, and the strategy to reuse computations can be applied to any hardware platform where they would immediately improve computational efficiency. For instance, more powerful computational hardware could then gain computation time for other tasks, or could compute the joint torques for more sensors.

5.2.3. Experimental Validation

The following sections compare the accuracy of the distributedly computed skin joint torque with the one computed in the controller process on the computer, and analyze the obtained efficiency gain of the new approach in the control loop.

5.2.3.1 Accuracy of the Skin Joint Torque

Figure 62 compares the skin joint torques $\boldsymbol{\tau}_{\text{skin}} = (\tau_l)$ computed by the controller process on the computer with the skin joint torques $\boldsymbol{\tau}_{\text{skin,dstb}} = (\tau_{\text{dstb},l})$ computed by the skin cells. The resulting reaction of the controller can be seen in Figure 63.

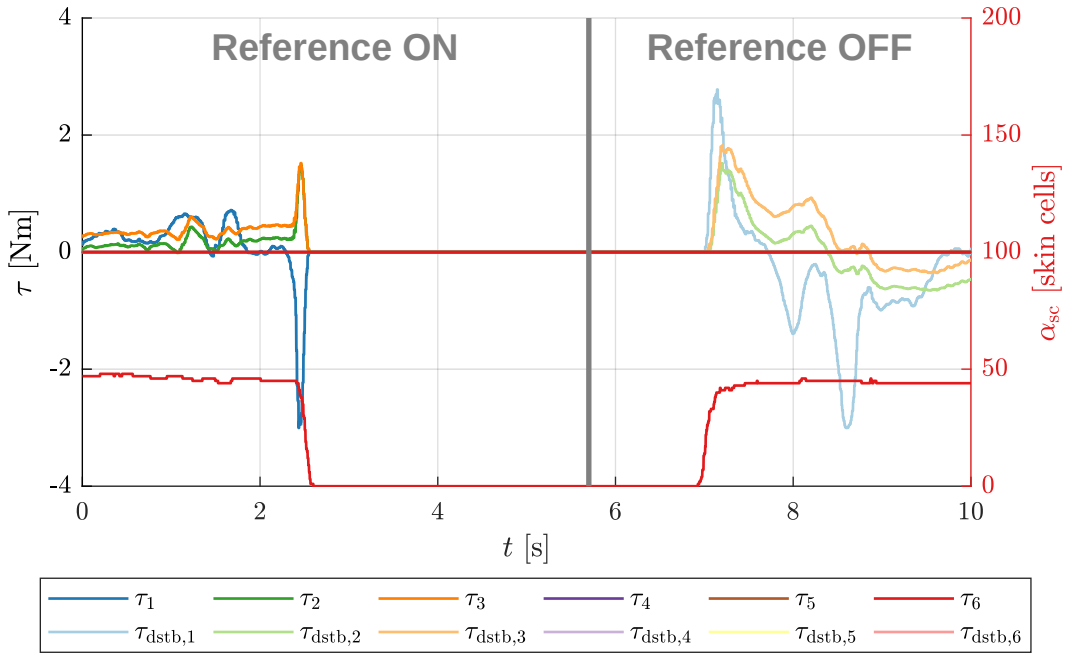


Figure 62 The comparison of the skin joint torque $\boldsymbol{\tau}_{\mathrm{skin,dstb}} = (\tau_{\mathrm{dstb},l})$ computed on the skin cells with joint torques $\boldsymbol{\tau}_{\mathrm{skin}} = (\tau_l)$ computed in the real-time control loop on the computer. The reference computation of $\boldsymbol{\tau}_{\mathrm{skin}}$ is performed on the left side (before $t = 5.7$ s), and switched off on the right side. The joint torques depicted on the left are practically identical. The mean absolute error of $\boldsymbol{\tau}_{\mathrm{dstb}}$ with respect to $\boldsymbol{\tau}$ is 0.0054 N m and the error of $\boldsymbol{\tau}_{\mathrm{dstb,add}}$ is 0.0356 N m. The plot additionally depicts the number of active skin cells α_{sc}.

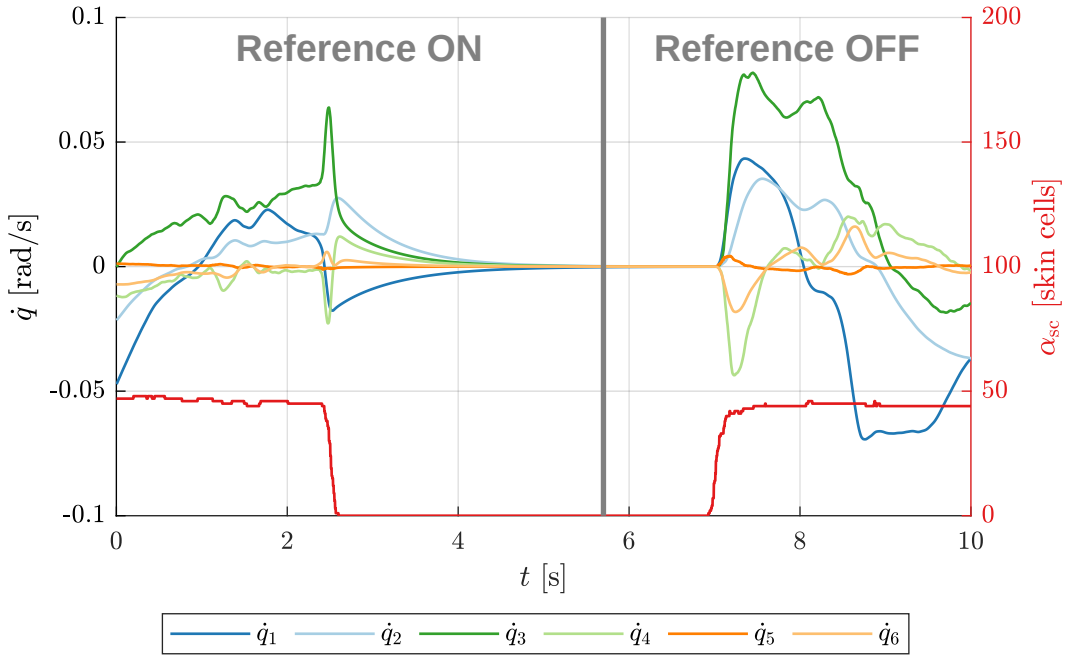


Figure 63 The joint velocities $\dot{\mathbf{q}}$ resulting from the skin joint torque $\boldsymbol{\tau}_{\mathrm{skin,dstb}}$ computed on the skin cells, and the number of active skin cells α_{sc}.

Up to the time $t = 5.7\,\text{s}$, the controller computes $\boldsymbol{\tau}_{\text{skin}}$ and receives $\boldsymbol{\tau}_{\text{skin,dstb}}$ (left side in Figures 62 and 63). Visually, there is no difference between these two joint torques. The average error for all experiments is 0.0054 N m for the non-summed up $\boldsymbol{\tau}_i$ in $Q3.12$ and 0.0356 N m for the summed up $\boldsymbol{\tau}_i$ in $Q7.8$. This error is neglectable for standard industrial robots, such as the one used in the experiments.

5.2.3.2 Computational Cost in the Real-Time Control Loop

Figure 64 presents the total cycle time T_{total} of the real-time control loop. The control loop runs at 500 Hz (Section 5.1.1), therefore, if the total cycle time crosses the border of 2 ms, then the controller breaks its real-time constraint.

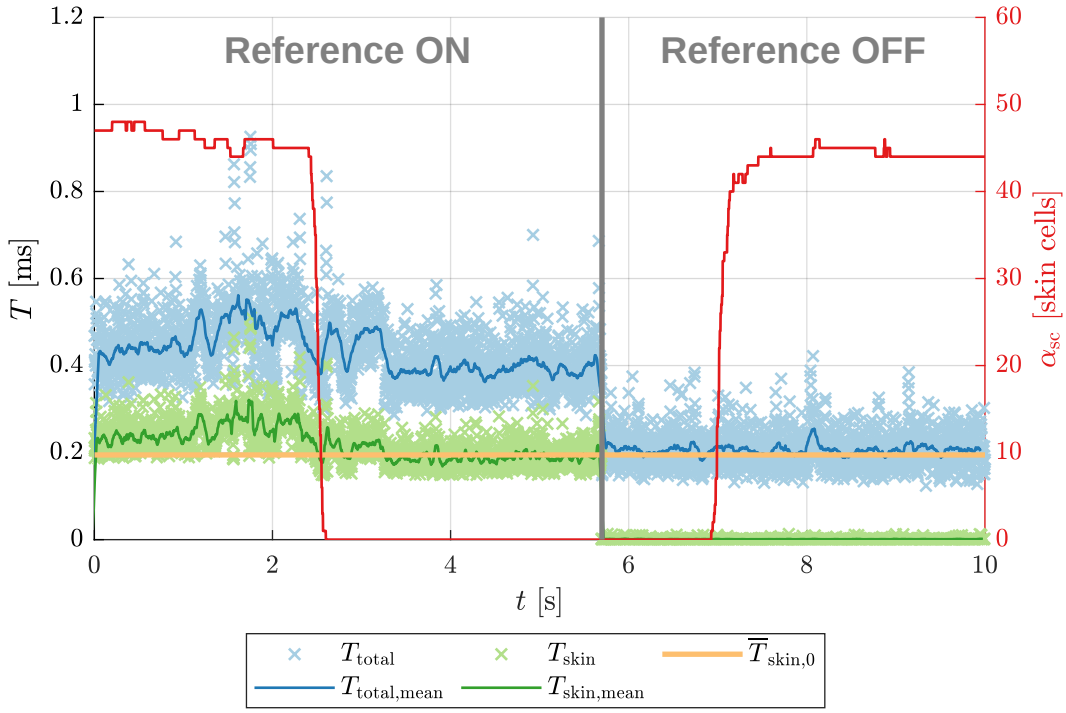


Figure 64 The cycle times in ms when using the distributedly computed skin joint torque $\boldsymbol{\tau}_{\text{skin,dstb}}$. The reference computation of $\boldsymbol{\tau}_{\text{skin}}$ is performed on the left side (before $t = 5.7\,\text{s}$), and switched off on the right side. T_{skin} is the delay in the real-time control loop when computing the skin joint torque $\boldsymbol{\tau}_{\text{skin}}$. The cycle time T_{total} of the controller drops when switching to the distributed computation in the skin cells (switching off the reference computation of $\boldsymbol{\tau}_{\text{skin}}$). $\overline{T}_{\text{skin,0}}$ is the average time for computing $\boldsymbol{\tau}_{\text{skin}}$ when the number of active skin cells α_{sc} is zero (for a total number of $n_{\text{sc}} = 253$ skin cells).

The cycle time of the reactive skin controller T_{skin} is part of the total cycle time T_{total} and $\overline{T}_{\text{skin,0}}$ is the average of T_{skin} under the condition that the number of active skin cells α_{sc} is zero. The drop of both, T_{total} and T_{skin} at $t = 5.7\,\text{s}$ clearly indicates the switching to the distributed computation of the skin joint torques $\boldsymbol{\tau}_{\text{skin,dstb}}$. T_{skin} drops to zero and the total cycle time T_{total} reduces to more or less 0.2 ms. Most notably T_{total} is now independent of the number of active skin cells α_{sc}, which has been not possible if the computing of $\boldsymbol{\tau}_{\text{skin}}$ is within the control loop. This is because, whenever the $\boldsymbol{\tau}_{\text{skin}}$ is computed within the control

loop, T_{skin} will depend on the number of skin cells n_{sc} and also on the number of active skin cells α_{sc}.

5.2.3.3 Relationship between the Number of Skin Cells and the Cycle Time

The cycle time T_{skin} of a reactive skin controller, that computes $\boldsymbol{\tau}_{\text{skin}}$ in the real-time loop, depends on the total number of skin cells n_{sc}. Even if there are no active skin cells, T_{skin} increases linearly with the total number of skin cells n_{sc}, see Figure 65.

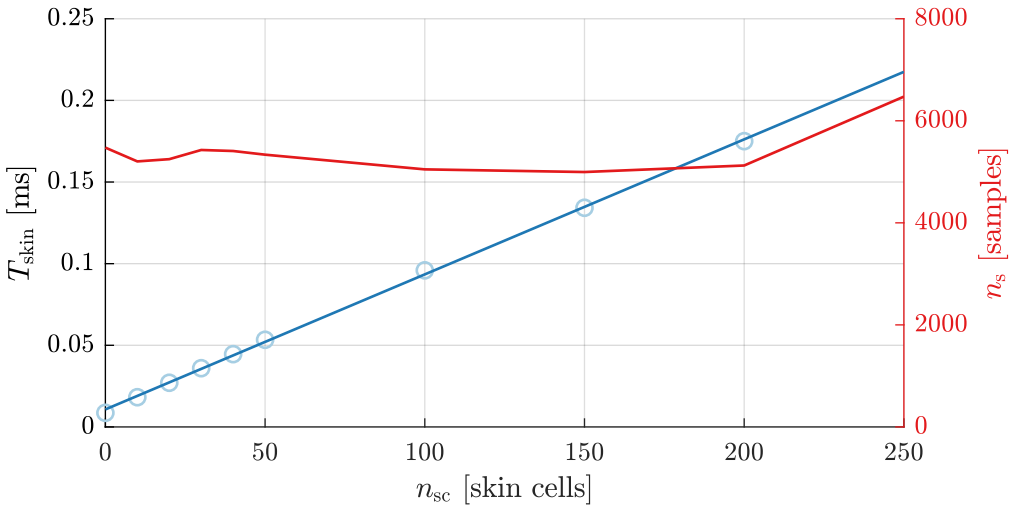


Figure 65 The cycle time T_{skin} of the reactive contact controller computing $\boldsymbol{\tau}_{\text{skin}}$ in the case that the number of active skin cells α_{sc} is zero. The linear approximation takes the number of samples n_{s} as weight.

The linear approximation for this relationship is

$$T_{\text{skin}}(n_{\text{sc}}) = 0.827 \cdot 10^{-6}\,\text{s}\; n_{\text{sc}} + 0.01073 \cdot 10^{-3}\,\text{s} \tag{5.32}$$

with a fitting accuracy of 99.95 %. The linear approximation is weighted with the number of samples. The dependency of T_{skin} on the number of cells is the result of traversing the sensor values of the skin cells in memory in order to detect active cells. For a control rate of 500 Hz and a cycle time of 0.2 ms for the other parts of the controller (Figure 64), the theoretical limit for the total number of skin cells n_{sc} is 2160 skin cells. However, activating only one of these 2160 skin cells would then break the real-time constraint for 500 Hz.

5.2.3.4 Relationship between Number of Active Cells and the Cycle Time

The cycle time T_{skin} of a reactive contact controller that computes $\boldsymbol{\tau}_{\text{skin}}$ in the real-time loop additionally depends on the number of active skin cells α_{sc} as depicted in Figure 66.

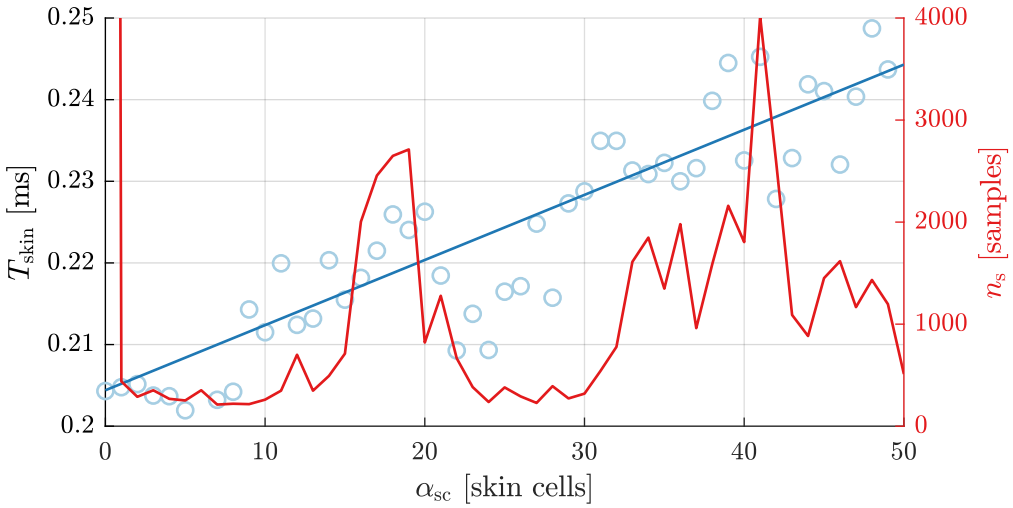


Figure 66 The dependency between the number of active skin cells α_{sc} and the cycle time T_{skin} for skin joint torques computed by the controller in the real-time loop for a total of 253 skin cells. The linear approximation takes the number of samples n_{s} as weight.

The linear approximation for the relationship between cycle time and number of active skin cells with the number of samples as weight is

$$T_{\text{skin}}(\alpha_{\text{sc}}) = 0.798 \cdot 10^{-6}\,\text{s}\;\alpha_{\text{sc}} + 0.2 \cdot 10^{-3}\,\text{s} \tag{5.33}$$

with a fitting accuracy of 93.9 %. Combining Equations (5.32) and (5.33) results in

$$\begin{aligned} T_{\text{skin}}(\alpha_{\text{sc}}, n_{\text{sc}}) = {} & 0.798 \cdot 10^{-6}\,\text{s}\;\alpha_{\text{sc}} \\ & + 0.827 \cdot 10^{-6}\,\text{s}\;n_{\text{sc}} \\ & + 0.01073 \cdot 10^{-3}\,\text{s} \end{aligned} \tag{5.34}$$

Under the assumption that the maximum number of active skin cells is $\max(\alpha_{\text{sc}}) \leq 100$, the limit for the total number of skin cells when computing $\boldsymbol{\tau}_{\text{skin}}$ in the control loop, is $n_{\text{sc}} = 2067$ skin cells. The real-time constraint would be broken for more skin cells, or respectively, for more active skin cells.

The distributed computation of skin joint torques at the skin cells presented here is efficient, feasible, accurate, and applicable on a real robotic system. This approach successfully removes the delay of fusing tactile with proprioceptive information to motor actions from the control loop. Removing this delay not only relaxes the demand to fulfill real-time constraints. More importantly, it also removes any upper bounds for the maximum number of active skin cells or the total number of skin cells in reactive contact control.

5.3. Event-Driven Reactive Contact Control

Sections 5.1 and 5.2 introduced the clock-driven reactive contact control. That is a controller that fuses tactile with proprioceptive information to motor actions. Both presented approaches successfully improve the overall performance of the control system. First by improving the system that provides tactile information to the controller. Second by decentralizing the fusion of tactile and proprioceptive to motor commands. These two contributions do not yet exploit the efficient event-driven representation of tactile information, or the event-driven approach in the reactive control algorithm itself. However, the successful exploitation of both is expected to contribute to more efficient clock- and event-driven control algorithms with lower demands on computational power, improving the feasibility of large-area interaction, which is one objective of this thesis. This section confirms these expectations. It contributes effective methods for clock-driven algorithms that utilize event-driven tactile information to avoid the inefficient search for tactile saliency in large data sets, and a fully event-driven reactive control algorithm.

To this end, Section 5.3.2 presents the analytical approach that splits the standard reactive control algorithm of Section 5.1 according to its proprioceptive and tactile input modalities. This approach additionally removes all redundant calculations. The resulting improved calculations contribute to the novel and more efficient clock-driven algorithm presented in Section 5.3.3, and the novel event-driven algorithm presented in Section 5.3.4. Only tactile and proprioceptive events drive the computation in the event-driven algorithm. The effectiveness of both of these novel reactive control algorithms is validated and evaluated by an experimental study in Section 5.3.5. The results evidence a significant improvement in the system's performance.

The work presented in Section 5.3 was in part published in:
Bergner, F., Dean-Leon, E., Cheng, G., "Efficient Event-Driven Reactive Control for Large Scale Robot Skin". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Singapore, Singapore, 2017, pp. 394–400.

5.3.1. Experimental Setup – Event-Driven Reactive Contact Control

The reactive contact controller presented in this section realizes two operation modes, the clock-driven and the event-driven mode. Similar to the previous sections, the controller is evaluated with the UR5 robot arm that is covered with 253 skin cells. Figure 67 depicts the implemented action-perception loop with the reactive contact control either in event-driven or clock-driven operation mode.

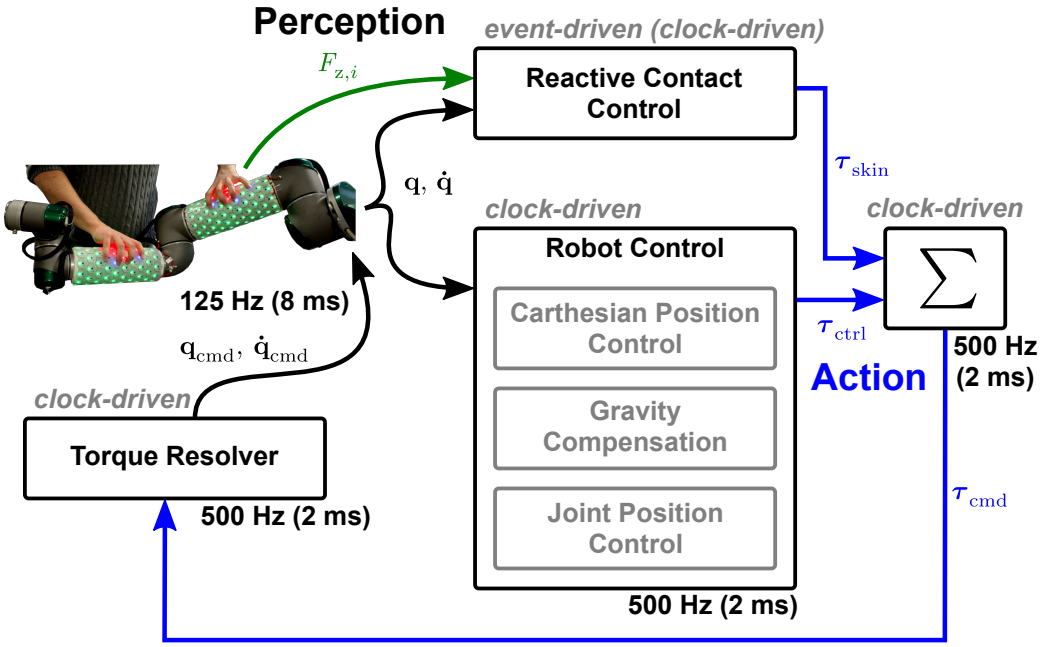


Figure 67 The controller block fuses tactile with proprioceptive information. The event-driven reactive contact controller is combined with several other controllers to achieve a desired control behavior, e.g. a compliant behavior. The motor actions are generated with joint torques τ. The torque resolver [40, 44] transforms the resulting joint torque τ_{cmd} to joint position $\mathbf{q}_{cmd}$ or joint velocity commands $\dot{\mathbf{q}}_{cmd}$.

In contrast to the previous evaluations, the detailed and comprehensive performance evaluation of the novel event-driven controller requires a tactile interaction that is exactly repeatable. Then, the controller processes similar tactile and proprioceptive input information and thus computes similar reactions in the clock-driven operation mode with clock-driven algorithms and in the event-driven operation mode with event-driven algorithms. Thus, repeatable tactile interactions allow for the direct comparison of the event-driven controller with its clock-driven reference.

The stochastic evaluation of similar tactile human-robot interactions, as performed in the previous Section 5.1, is not feasible. Since the evaluated algorithms are different, it would not be possible to separate the differences caused by the uncertain human interaction from the differences caused by the clock- and event-driven algorithms. Therefore, the tactile interaction was controlled by employing robot-robot interaction, see Figure 68.

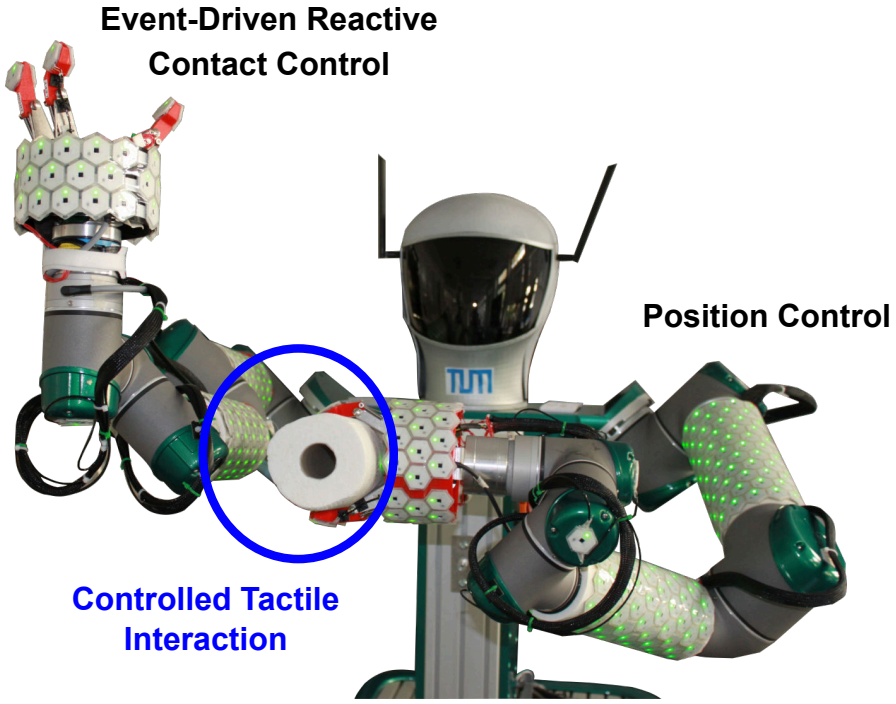
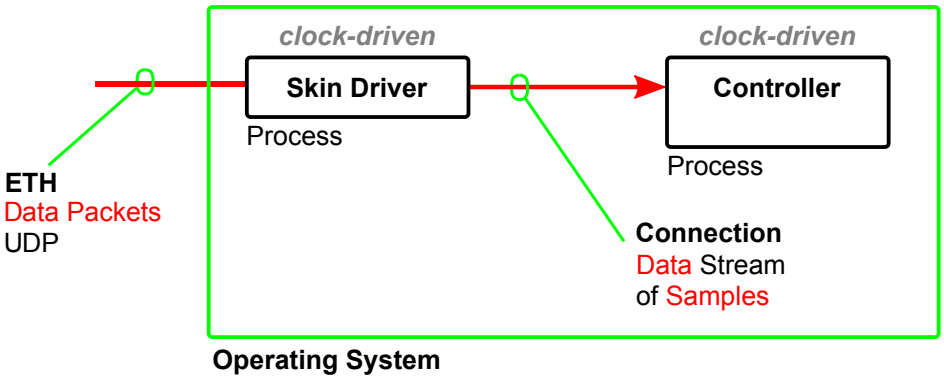


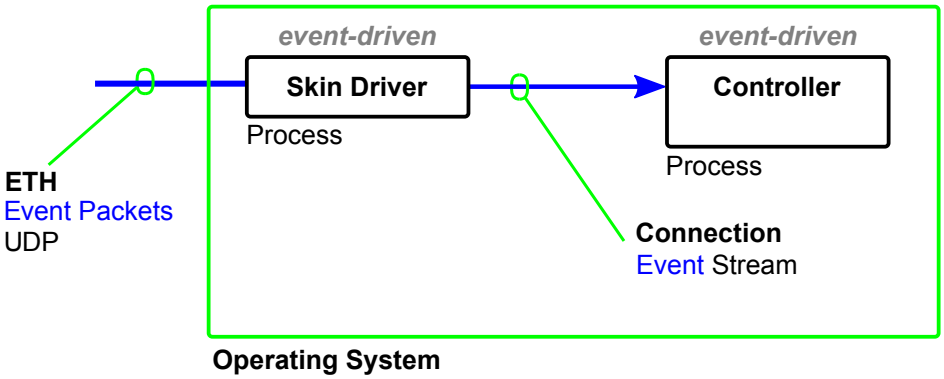
Figure 68 The novel event-driven controller is evaluated with a repeatable robot-robot interaction where one UR5 arm is position-controlled, and the other one reacts to contacts. Here, the controlled interaction is executed by the two UR5 robot arms of the robot TOMM [42, 29]. The left UR5 robot arm holds a paper towel in a gripper, which is used in the experiment to push the right UR5 robot arm that reacts to contacts – it tries to avoid the contact and moves to the right.

In the robot-robot interaction, another UR5 robot arm moves along a predefined trajectory and touches the lower arm of the reactive UR5 robot arm with a paper towel. The high movement precision of the UR5 arms allow for the exact repeatability of the tactile interaction.

For the experiments, the e-skin and the reactive controller were either executed in the clock-driven mode or the event-driven mode (Figure 69). In event-driven mode, the e-skin uses the event thresholds with improved noise canceling (Table 17). The sample rate of the e-skin is 62.5 Hz in both operation modes.



(a) The clock-driven reference system.



(b) The event-driven reactive contact controller with the event-driven skin driver.

Figure 69 The clock-driven reference and the event-driven system for the evaluated reactive contact control. ETH denotes the Ethernet connection of the communication interface to the e-skin.

5.3.2. Modality Separated Skin Joint Torque Computation

As presented in Section 5.1.2, the joint torque contribution $\boldsymbol{\tau}_i$ of a skin cell i either computes by a matrix vector multiplication (Equation (5.5)) or component-wise (Equation (5.10)). This section now focuses on the optimization of Equation (5.10) by removing redundant computations and splitting the computations according to their dependency on joint positions $\mathbf{q}$, static transformations, and the virtual forces $F_{\mathrm{z},i}$ of the skin cells (Algorithm 3).

According to Equation (5.9), the l-th component of $\boldsymbol{\tau}_i$ for a skin cell i on joint k (Figure 52) computes by

$$\begin{aligned}\tau_{l,i}(\mathbf{q}) &= \mathbf{j}_{l,i}^{\top}(\mathbf{q})\ {}_0\mathbf{w}_i \\ &= \left[{}_0\mathbf{z}_{l-1} \times \left({}_0\mathbf{t}_i - {}_0\mathbf{t}_{l-1}\right)\right]^{\top}\ {}_0^k\mathbf{R}\ {}_k\mathbf{z}_i\ F_{\mathrm{z},i}\end{aligned} \tag{5.35}$$

and can be rearranged to

$$\begin{aligned}\tau_{l,i}(\mathbf{q}) &= \left(\left[{}_0\mathbf{t}_i - {}_0\mathbf{t}_{l-1}\right]^{\top}\ \left[{}_0\mathbf{z}_{l-1}\right]_{\times}^{\top}\right) {}_0^k\mathbf{R}\ {}_k\mathbf{z}_i\ F_{\mathrm{z},i} \\ &= \left[A_{l,i}(\mathbf{q}) - B_{l,i}(\mathbf{q})\right]\ F_{\mathrm{z},i}\end{aligned} \tag{5.36}$$

with

$$\begin{aligned} A_{l,i}(\mathbf{q}) &= {}_0\mathbf{t}_i^\top \ \left[{}_0\mathbf{z}_{l-1}\right]_\times^\top \ {}_0^k\mathbf{R} \ {}_k\mathbf{z}_i \\ &= {}_k\bar{\mathbf{t}}_i^\top \ {}_0^k\mathbf{T}^\top \ \left[{}_0\bar{\mathbf{z}}_{l-1}\right]_\times^\top \ {}_0^k\mathbf{T} \ {}_k\hat{\mathbf{z}}_i \\ &= {}_k\bar{\mathbf{t}}_i^\top \ \mathbf{S}_{l,k}(\mathbf{q}) \ {}_k\hat{\mathbf{z}}_i \end{aligned} \tag{5.37}$$

and

$$\begin{aligned} B_{l,i}(\mathbf{q}) &= {}_0\mathbf{t}_{l-1}^\top \ \left[{}_0\mathbf{z}_{l-1}\right]_\times^\top \ {}_0^k\mathbf{R} \ {}_k\mathbf{z}_i \\ &= {}_k\bar{\mathbf{t}}_{l-1}^\top \ \left[{}_0\bar{\mathbf{z}}_{l-1}\right]_\times^\top \ {}_0^k\mathbf{T} \ {}_k\hat{\mathbf{z}}_i \\ &= \mathbf{t}_{l,k}^\top(\mathbf{q}) \ {}_k\hat{\mathbf{z}}_i. \end{aligned} \tag{5.38}$$

The formulas $\mathbf{S}_{l,k}(\mathbf{q}) \in \mathbb{R}^{4\times4}$ and $\mathbf{t}_{l,k}^\top(\mathbf{q}) \in \mathbb{R}^4$ of Equations (5.37) and (5.38) depend on the active joint k and the joint position $\mathbf{q}$. Since these formulas do not depend on the skin cell i they only have to be computed once on the update of $\mathbf{q}$ for each $k \leq k_{\max}$. $k_{\max}$ is the joint that has an active skin cell with the longest kinematic chain to the base coordinate frame 0 (Figure 52). ${}_k\hat{\mathbf{z}}_i \in \mathbb{R}^4$ and ${}_k\bar{\mathbf{t}}_i \in \mathbb{R}^4$ are extracted from the static transformation ${}_k^i\mathbf{T}$ (analog Equation (5.12)) and are different for each skin cell i. Nevertheless, since they are static, they only need to be acquired once during the 3D surface calibration of the skin [102, 107] (Section 5.1.2.2). The bar above vectors indicate that these vectors have been extended to their homogeneous representation and the hat respectively that a zero has been appended for example:

$${}_k\bar{\mathbf{t}}_i = \begin{bmatrix} {}_k\mathbf{t}_i \\ 1 \end{bmatrix} \tag{5.39}$$

and

$${}_k\hat{\mathbf{z}}_i = \begin{bmatrix} {}_k\mathbf{z}_i \\ 0 \end{bmatrix}. \tag{5.40}$$

The cross-product matrix operator $[\cdot]_\times$ is defined by

$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times3} \tag{5.41}$$

with $\mathbf{a} \in \mathbb{R}^3$, or respectively, for $\bar{\mathbf{a}} \in \mathbb{R}^4$ by

$$[\bar{\mathbf{a}}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 & 0 \\ a_3 & 0 & -a_1 & 0 \\ -a_2 & a_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times4}. \tag{5.42}$$

The cross-product matrix operator $[\cdot]_\times$ produces a skew symmetric matrix. Thus, the calculation of $\mathbf{S}_{l,k}(\mathbf{q})$ and $\mathbf{t}_{l,k}^\top(\mathbf{q})$ further simplifies to

$$\mathbf{S}_{l,k}(\mathbf{q}) = {}^k_0\mathbf{T}^\top \left[{}_0\bar{\mathbf{z}}_{l-1}\right]_\times^\top {}^k_0\mathbf{T} \tag{5.43}$$
$$= \begin{bmatrix} 0 & a & b & c \\ -a & 0 & d & e \\ -b & -d & 0 & f \\ -c & -e & -f & 1 \end{bmatrix}$$

$$a = ({}_0\mathbf{x}_k \times {}_0\mathbf{y}_k)^\top \; {}_0\mathbf{z}_{l-1} \qquad b = ({}_0\mathbf{x}_k \times {}_0\mathbf{z}_k)^\top \; {}_0\mathbf{z}_{l-1}$$
$$c = ({}_0\mathbf{x}_k \times {}_0\mathbf{t}_k)^\top \; {}_0\mathbf{z}_{l-1} \qquad d = ({}_0\mathbf{y}_k \times {}_0\mathbf{z}_k)^\top \; {}_0\mathbf{z}_{l-1}$$
$$e = ({}_0\mathbf{y}_k \times {}_0\mathbf{t}_k)^\top \; {}_0\mathbf{z}_{l-1} \qquad f = ({}_0\mathbf{z}_k \times {}_0\mathbf{t}_k)^\top \; {}_0\mathbf{z}_{l-1}$$

and

$$\mathbf{t}_{l,k}^\top(\mathbf{q}) = {}_k\bar{\mathbf{t}}_{l-1}^\top \left[{}_0\bar{\mathbf{z}}_{l-1}\right]_\times^\top {}^k_0\mathbf{T} \tag{5.44}$$
$$= \begin{pmatrix} \bar{a} & \bar{b} & \bar{c} & \bar{d} \end{pmatrix}$$

$$\bar{a} = {}_0\mathbf{x}_k^\top \left({}_0\mathbf{z}_{l-1} \times {}_0\mathbf{t}_{l-1}\right) \qquad \bar{b} = {}_0\mathbf{y}_k^\top \left({}_0\mathbf{z}_{l-1} \times {}_0\mathbf{t}_{l-1}\right)$$
$$\bar{c} = {}_0\mathbf{z}_k^\top \left({}_0\mathbf{z}_{l-1} \times {}_0\mathbf{t}_{l-1}\right) \qquad \bar{d} = {}_0\mathbf{t}_k^\top \left({}_0\mathbf{z}_{l-1} \times {}_0\mathbf{t}_{l-1}\right) + 1$$

where ${}_0\mathbf{x}_k$, ${}_0\mathbf{y}_k$, and ${}_0\mathbf{z}_k$ are extracted from the transformation ${}^k_0\mathbf{T}$ (Equation (5.12)). Analog, ${}_0\mathbf{z}_{l-1}$ and ${}_0\mathbf{t}_{l-1}$ are extracted from ${}^{l-1}_0\mathbf{T}$. Taking Equations (5.43) and (5.44), $A_{l,i}(\mathbf{q})$ and $B_{l,i}(\mathbf{q})$ of Equations (5.37) and (5.44) can be analytically expanded. Then, their results can be ordered and re-factorized employing the permutation rules of the triple product expansion of the cross-product. All the entries of $\mathbf{S}_{l,k}(\mathbf{q})$ and $\mathbf{t}_{l,k}^\top(\mathbf{q})$ are such triple products. The ordering and re-factorization reduce the redundant calculations caused by the extended skew symmetric matrix $\mathbf{S}_{l,k}(\mathbf{q})$ with the following results:

$$A_{l,i}(\mathbf{q}) = ({}_k\mathbf{z}_i \times {}_k\mathbf{t}_i)^\top \begin{bmatrix} d \\ -b \\ a \end{bmatrix} - {}_k\mathbf{z}_i^\top \begin{bmatrix} c \\ e \\ f \end{bmatrix} \tag{5.45}$$
$$= {}_0\mathbf{z}_{l-1}^\top \left[{}^k_0\mathbf{A}(\mathbf{q})\,({}_k\mathbf{z}_i \times {}_k\mathbf{t}_i) - {}^k_0\mathbf{B}(\mathbf{q})\,{}_k\mathbf{z}_i\right]$$
$$= {}_0\mathbf{z}_{l-1}^\top \left[{}^k_0\mathbf{A}(\mathbf{q})\;{}_k\mathbf{c}_i - {}^k_0\mathbf{B}(\mathbf{q})\;{}_k\mathbf{z}_i\right]$$
$$= {}_0\mathbf{z}_{l-1}^\top\;{}_0\mathbf{c}_i(\mathbf{q})$$

and

$$
\begin{aligned}
B_{l,i}(\mathbf{q}) &= {}_k\mathbf{z}_i^\top \; {}^k_0\mathbf{R}^\top \left({}_k\mathbf{z}_{l-1} \times {}_k\mathbf{t}_{l-1} \right) \\
&= {}_0\mathbf{z}_{\mathrm{r},i}^\top(\mathbf{q}) \; {}_0\mathbf{c}_{l-1}(\mathbf{q})
\end{aligned}
\tag{5.46}
$$

with

$$
{}_0\mathbf{c}_i(\mathbf{q}) = {}^k_0\mathbf{A} \; {}_k\mathbf{c}_i - {}^k_0\mathbf{B} \; {}_k\mathbf{z}_i \qquad \in \mathbb{R}^3 \tag{5.47}
$$

$$
{}^k_0\mathbf{A}(\mathbf{q}) = \begin{pmatrix} {}_0\mathbf{z}_k \times {}_0\mathbf{y}_k & {}_0\mathbf{x}_k \times {}_0\mathbf{z}_k & {}_0\mathbf{y}_k \times {}_0\mathbf{x}_k \end{pmatrix} \qquad \in \mathbb{R}^{3\times3} \tag{5.48}
$$

$$
{}^k_0\mathbf{B}(\mathbf{q}) = \begin{pmatrix} {}_0\mathbf{x}_k \times {}_0\mathbf{t}_k & {}_0\mathbf{y}_k \times {}_0\mathbf{t}_k & {}_0\mathbf{z}_k \times {}_0\mathbf{t}_k \end{pmatrix} \qquad \in \mathbb{R}^{3\times3} \tag{5.49}
$$

$$
{}_k\mathbf{c}_i = {}_k\mathbf{z}_i \times {}_k\mathbf{t}_i \qquad \in \mathbb{R}^3 \tag{5.50}
$$

$$
{}_0\mathbf{z}_{\mathrm{r},i}(\mathbf{q}) = {}^k_0\mathbf{R} \; {}_k\mathbf{z}_i \qquad \in \mathbb{R}^3 \tag{5.51}
$$

$$
{}^k_0\mathbf{R}(\mathbf{q}) = \begin{pmatrix} {}_0\mathbf{x}_k & {}_0\mathbf{y}_k & {}_0\mathbf{z}_k \end{pmatrix} \qquad \in \mathbb{R}^{3\times3} \tag{5.52}
$$

$$
{}_0\mathbf{c}_{l-1}(\mathbf{q}) = {}_0\mathbf{z}_{l-1} \times {}_0\mathbf{t}_{l-1} \qquad \in \mathbb{R}^3. \tag{5.53}
$$

Thus, the components $\tau_{l,i}$ of $\boldsymbol{\tau}_i$ of a skin cell i compute by

$$
\begin{aligned}
\tau_{l,i}(\mathbf{q}) &= J_{l,i}(\mathbf{q}) \; F_{\mathrm{z},i} \\
&= \left[A_{l,i}(\mathbf{q}) - B_{l,i}(\mathbf{q}) \right] \; F_{\mathrm{z},i} \\
&= \left[{}_0\mathbf{z}_{l-1}^\top(\mathbf{q}) \; {}_0\mathbf{c}_i(\mathbf{q}) - {}_0\mathbf{z}_{\mathrm{r},i}^\top(\mathbf{q}) \; {}_0\mathbf{c}_{l-1}(\mathbf{q}) \right] \; F_{\mathrm{z},i}.
\end{aligned}
\tag{5.54}
$$

From Equations (5.47) to (5.53) we observe that computing Equation (5.54) splits into three types of equations:

1. Equations (5.48), (5.49), (5.52), and (5.53) colored in red, and depending on k and $\mathbf{q}$ have to be calculated only once per active joint k,
2. Equation (5.50), colored in green and depending on i, is static and does not change, but has to be provided for each skin cell i, and
3. Equations (5.47) and (5.51), colored in blue and depending on the former two types (1. and 2.), must be computed for each active skin cell and for each change of $\mathbf{q}$.

The Equations of type 1 only depend on terms of the robot kinematics, namely the axes ${}_0\mathbf{x}_l$, ${}_0\mathbf{y}_l$, and ${}_0\mathbf{z}_l$ and the translations ${}_0\mathbf{t}_l$ of the joints' coordinate frames l up to joint k. Thus, these equations are only updated when the robot state changes, that is when the joint positions $\mathbf{q}$ change. The Equations of type 3 combine joint information with skin cell specific information and are updated when the skin cell i is active and when the joint positions $\mathbf{q}$ change.

The presented component-wise computation of skin joint torques for the reactive contact control serves two purposes:

1. The reduction of redundant computations enhances the efficiency of the expensive computation of skin joint torques for both operations modes of the controller (clock-driven and event-driven).

2. The separation of computations according to their dependency on joint position updates and the current skin cell i prepares the algorithm for the realization of event-driven computations.

5.3.3. Optimized Reactive Contact Control – Clock-Driven Mode

Exploiting the improvements on the computation of $\boldsymbol{\tau}_i$ introduced in Section 5.3.2, an optimized reactive contact controller operating in clock-driven mode is developed. Rather than looping in each control cycle through all skin cells, as in the skin cell-wise joint torque computation (Algorithm 4), Algorithm 5 is employed.

Algorithm 5 Update $F_{\mathrm{z},i}$ for a skin cell data packet

1: **get** skin cell ID, F_{c}, and F_{p} **from** data packet (Figure 29a)
2: **map** ID **to** index i
3: **calculate** $F_{\mathrm{z},i}$ according to Algorithm 3
4: **if** $F_{\mathrm{z},i} > 0$ **then**
5: **mark** skin cell i as *active*
6: **end if**

Algorithm 5 (update-driven virtual force computation) is executed whenever the controller receives a new skin cell data packet containing the normalized sensor values and the skin cell ID. When executed, the update-driven virtual force computation first maps the skin cell ID to the corresponding skin cell index i and then updates the virtual force $F_{\mathrm{z},i}$ in the controller memory. When $F_{\mathrm{z},i}$ is greater than zero, then the skin cell is marked as active. The update-driven virtual force computation (Algorithm 5) reduces the number of executions of the virtual force computation (Algorithm 3) to the sample rate/update rate of the skin cells rather than the control loop rate. Furthermore, the update-driven virtual force computation reduces the complexity of searching active skin cells in the control loop. Thus, the update-driven virtual force computation alone is expected to significantly increase the performance of the controller in comparison to the skin cell-wise joint torque computation (Algorithm 4).

Algorithm 6 presents the optimized computation of the skin joint torque $\boldsymbol{\tau}_{\mathrm{skin}}$, that is the sum of all contributions $\boldsymbol{\tau}_i$ of active skin cells i.

Algorithm 6 Update $\boldsymbol{\tau}_{\mathrm{skin}}$ for a joint position $\mathbf{q}$

1: $\boldsymbol{\tau}_{\mathrm{skin}} := 0$
2: **update** ${}_0^l\mathbf{T}(\mathbf{q}) \quad \forall l \in 1, 2, \ldots, \mathrm{DOF}$
3: **calculate** ${}_0\mathbf{c}_{l-1}(\mathbf{q}) \quad \forall l \in 1, 2, \ldots, \mathrm{DOF}$
4: **calculate** ${}_0^k\mathbf{A}(\mathbf{q}) \quad \forall k \in 1, 2, \ldots, \mathrm{DOF}$
5: **calculate** ${}_0^k\mathbf{B}(\mathbf{q}) \quad \forall k \in 1, 2, \ldots, \mathrm{DOF}$
6: **for all** active skin cells i **do**
7: **get** joint k for skin cell i
8: **calculate** $\tau_{l,i}(\mathbf{q}) := J_{l,i}(\mathbf{q})\ F_{\mathrm{z},i} \quad \forall l \leq k$
9: **compose** $\tau_{l,i}$ **to** $\boldsymbol{\tau}_i$
10: $\boldsymbol{\tau}_{\mathrm{skin}} := \boldsymbol{\tau} + \boldsymbol{\tau}_i$
11: **end for**

The optimized skin joint torque computation (Algorithm 6) is executed in the control loop of the reactive contact controller. First, the optimized skin joint torque computation updates the kinematic chain ${}_0^l\mathbf{T}(\mathbf{q})$ according to the new joint position $\mathbf{q}$. Then, it calculates the joint torque $\boldsymbol{\tau}_i$ for each active skin cell. The optimized skin joint torque computation is a major improvement in comparison to the skin cell-wise joint torque computation (Algorithm 4) since:

1. The algorithm pre-computes common terms once in Line 3 to Line 5 and then re-uses them in Line 9.
2. The algorithm loops (Line 6) only over the set of active skin cells rather than over all skin cells, $\alpha_{\mathrm{sc}} \leq n_{\mathrm{sc}}$.

The update-driven virtual force computation (Algorithm 5) and the optimized skin joint torque computation (Algorithm 6) are expected to contribute a significant performance increase in comparison to the skin cell-wise joint torque computation (Algorithm 4), which will be validated in Section 5.3.5.

5.3.4. Event-Driven Reactive Contact Control

The realization of event-driven reactive contact control utilizes two algorithms that are driven by different information sources and compute skin joint torques $\boldsymbol{\tau}_i$:

1. Algorithm 7 is driven by the tactile events of the event-driven e-skin system. This algorithm computes the virtual force, creates virtual force events, determines the joints with active skin cells, and computes the skin joint torque of active skin cells.

2. Algorithm 8 is driven by proprioceptive information. Therefore, this algorithm creates joint position events, and subsequently computes the skin joint torque of active skin cells.

The tactile-driven skin joint torque computation (Algorithm 7) is executed on the arrival of a force event or a proximity event. Then, it computes the virtual force $F_{\mathrm{z},i}$ employing Algorithm 3. After that, the tactile-driven skin joint torque computation implements an event generator (Equation (3.14)) for virtual force events. When a virtual force event is generated, then the algorithm proceeds with its computations. When the joint k of the skin cell i becomes *active*, that is when the active skin cell is on a joint k that up to now has not had any active skin cells, then the algorithm updates the active joint list K, the joint $k_{\max}$ that is active and has the longest kinematic chain, and the common pre-computations ${}^{k}_{0}\mathbf{A}(\mathbf{q})$, ${}^{k}_{0}\mathbf{B}(\mathbf{q})$, and ${}_{0}\mathbf{c}_{l-1}(\mathbf{q})$. When the joint k becomes *inactive*, then k is removed from the list of active joints. Eventually, the tactile-driven skin joint torque computation updates $\boldsymbol{\tau}_i$ and $\boldsymbol{\tau}_{\mathrm{skin}}$ exploiting the shared pre-computations.

The tactile-driven skin joint torque computation (Algorithm 7) is a significant improvement to the skin cell-wise joint torque computation (Algorithm 4). Furthermore, the tactile-driven skin joint torque computation substantially improves its optimized clock-driven counterparts (the update-driven virtual force computation (Algorithm 5), and the optimized skin joint torque computation (Algorithm 6)) since:

1. Algorithm 7 is only active and computes virtual forces $F_{\mathrm{z},i}$ when there is novel tactile information (force/proximity events).

2. Algorithm 7 only triggers the computation of joint torques $\boldsymbol{\tau}_i$ when $F_{\mathrm{z},i}$ changes significantly (virtual force events).

3. Algorithm 7 only triggers the pre-computation of the shared terms when they are required (active joints and active skin cells).

4. Algorithm 7 only triggers the pre-computation of the shared terms on the arrival of virtual force events (and joint position events (Algorithm 8)).

5. Algorithm 7 exploits the pre-computed shared terms for the optimized computation of $\boldsymbol{\tau}_i$.

Algorithm 7 Update $F_{\mathrm{z},i}$ and $\boldsymbol{\tau}_{\mathrm{skin}}$ for a skin cell event

if not force event **and not** proximity event **then**
  return
end if
get skin cell ID **from** event
map ID **to** i
if force event **then**
  $F_{\mathrm{c},i} :=$ value of event
end if
if proximity event **then**
  $F_{\mathrm{p},i} :=$ value of event
end if
calculate $F_{\mathrm{z,temp},i}$ according to Algorithm 3
if $\left|F_{\mathrm{z},i} - F_{\mathrm{z,temp},i}\right| < F_{\mathrm{z,e,th}}$ **then**
  return
end if
$F_{\mathrm{z},i} := F_{\mathrm{z,temp},i}$
get joint k for skin cell i
if joint k becomes *active* **then**
  add k **to** active joint list K
  $k_{\mathrm{temp}} := k_{\mathrm{max}}$
  update k_{max}
  for l **in** k_{temp} **to** k_{max} **do**
    update ${}_0\mathbf{c}_{l-1}(\mathbf{q})$
  end for
  update ${}_0^k\mathbf{A}(\mathbf{q})$, ${}_0^k\mathbf{B}(\mathbf{q})$
end if
if joint k becomes *inactive* **then**
  update k_{max}
  remove k **from** active joint list K
end if
$\boldsymbol{\tau}_{\mathrm{skin}} := \boldsymbol{\tau}_{\mathrm{skin}} - \boldsymbol{\tau}_i$
$\boldsymbol{\tau}_i := 0$
if $F_{\mathrm{z},i} > 0$ **then**
  calculate $\tau_{l,i}(\mathbf{q}) := J_{l,i}(\mathbf{q})\ F_{\mathrm{z},i} \quad \forall l \leq k$
  compose $\tau_{l,i}$ **to** $\boldsymbol{\tau}_i$
end if
$\boldsymbol{\tau}_{\mathrm{skin}} := \boldsymbol{\tau}_{\mathrm{skin}} + \boldsymbol{\tau}_i$

The proprioceptive-driven skin joint torque computation (Algorithm 8) is executed in the control loop of the reactive contact controller. It only executes the joint position event generator (Equation (3.14)) with the update rate of the control loop. The joint position event generator does not generate events for the components of $\mathbf{q}$, it rather generates one event containing all the components of $\mathbf{q}$ as soon as one of its components q_l surpasses the scalar joint position threshold $q_{\mathrm{e,th}}$. In the case of a joint position event, the algorithm updates the kinematic chain

${}^{l}_{0}\mathbf{T}(\mathbf{q})$, the shared terms ${}^{k}_{0}\mathbf{A}(\mathbf{q})$, ${}^{k}_{0}\mathbf{B}(\mathbf{q})$, and ${}_{0}\mathbf{c}_{l-1}(\mathbf{q})$, and the joint torques $\boldsymbol{\tau}_i$ of all active skin cells i.

Algorithm 8 Update $\boldsymbol{\tau}_{\mathrm{skin}}$ for a joint position $\mathbf{q}$

flag := **false**
for $l \in 1,\, 2,\, \ldots,\, \mathrm{DOF}$ **do**
  if $\left| q_{\mathrm{mem},l} - q_l \right| > q_{\mathrm{e,th}}$ **then**
    flag := **true**
    break
  end if
end for
if not *flag* **then**
  return $\boldsymbol{\tau}$
end if
$\mathbf{q}_{\mathrm{mem}} := \mathbf{q}$
update ${}^{l}_{0}\mathbf{T}(\mathbf{q}) \quad \forall l \in 1,\, \ldots,\, \mathrm{DOF}$
calculate ${}_{0}\mathbf{c}_{l-1}(\mathbf{q}) \quad \forall l \in 1,\, \ldots,\, k_{\max}$
calculate ${}^{k}_{0}\mathbf{A}(\mathbf{q}) \quad \forall k \in K$
calculate ${}^{k}_{0}\mathbf{B}(\mathbf{q}) \quad \forall k \in K$
for all active skin cells i **do**
  $\boldsymbol{\tau}_{\mathrm{skin}} := \boldsymbol{\tau}_{\mathrm{skin}} - \boldsymbol{\tau}_i$
  calculate $\tau_{l,i}(\mathbf{q}) := J_{l,i}(\mathbf{q})\, F_{\mathrm{z},i} \quad \forall l \leq k$
  compose $\tau_{l,i}$ **to** $\boldsymbol{\tau}_i$
  $\boldsymbol{\tau}_{\mathrm{skin}} := \boldsymbol{\tau}_{\mathrm{skin}} + \boldsymbol{\tau}_i$
end for

The proprioceptive-driven skin joint torque computation (Algorithm 8) is a significant improvement to the skin cell-wise joint torque computation (Algorithm 4). Furthermore, the proprioceptive-driven skin joint torque computation substantially improves its optimized clock-driven counterparts (the update-driven virtual force computation (Algorithm 5), and the optimized skin joint torque computation (Algorithm 6)) since:

1. Algorithm 8 is only active and computes the kinematics and the shared terms when there is novel proprioceptive information (joint position events).
2. Algorithm 8 only triggers the pre-computation of the shared terms when they are required (active joints and active skin cells).
3. Algorithm 8 only triggers the computation of joint torques $\boldsymbol{\tau}_i$ when $\mathbf{q}$ changes significantly (joint position events).
4. Algorithm 8 only triggers the computation of joint torques $\boldsymbol{\tau}_i$ for active skin cells.
5. Algorithm 8 exploits the pre-computed shared terms for the optimized computation of $\boldsymbol{\tau}_i$.

Overall, the presented event-driven algorithms for reactive contact control (the tactile- and proprioceptive-driven skin joint torque computations, Algorithms 7 and 8) constitute a significant performance increase in comparison to the previously presented clock-driven approaches (the skin cell-wise joint torque computation (Algorithm 4) and the optimized skin joint torque computation (Algorithms 5 and 6). The event-driven reactive contact controller's improved performance emerges from its novelty-driven computations (tactile- and proprioceptive-driven) and its avoidance of redundant computations, both reducing the total number of computations. The validation follows in Section 5.3.5.

The event-driven reactive contact controller extends this thesis efficient event-driven approach for large-area e-skin. It takes the approach from purely perceptive tactile systems to reactive contact control systems, whereby it removes the bottlenecks of clock-driven large-area contact control (Section 5.1) and allows for large-area tactile interactions. Furthermore, the presented strategies, for instance the update-driven virtual force computation, constitute a basis for improving the performance of complex clock-driven control algorithms depending on large-area tactile feedback (Section 5.4).

5.3.5. Experimental Validation

The following sections present the experimental results and evaluate the performance of the event-driven system with the e-skin and the controller in event-driven operation mode in comparison with its clock-driven counter part (e-skin and controller in clock-driven operation mode).

5.3.5.1 Comparability of Experiments and Controllers

The comprehensive performance evaluation of the new event-driven control algorithm requires a tactile interaction experiment that is reliably comparable and repeatable. The performance improvements of the event-driven system are only valid if the controller, or respectively, the system performance does not deteriorate. As presented in Section 5.3.1, the control of tactile interaction with a robot-robot interaction is used for the evaluation. This interaction consists in a total of six phases (Figures 70 to 72, and Table 28):

(1) The controller is idle, that is the right arm robot is not moving and not touched. The joint positions and tactile sensations are constant (do not change). Besides noise, there are no tactile and no joint position events.

(2) The tactile interaction with the robot arm starts. The tactile stimulation has not crossed its thresholds yet. There are no active skin cells, no skin joint torques are computed, and the skin joint torque is zero. The robot arm is not moving.

(3) The tactile interaction has enough intensity and there are active skin cells. The controller computes skin joint torques and the robot arm starts to respond to the interaction. The robot arm starts to move away from the contact area and avoids the contact.

(4) The tactile interaction's intensity decreases. The number of active skin cells decreases as does the number of skin joint torques the controller computes. The robot arm continues to respond to the contact by avoiding it.

(5) The tactile interaction with the robot arm is about to finish. The tactile stimulation falls below the thresholds and the number of active skin cells is zero. No skin joint torques are computed and the skin joint torque is zero. The joint velocities decrease (friction). The robot arm still moves because of its dynamics (inertia).

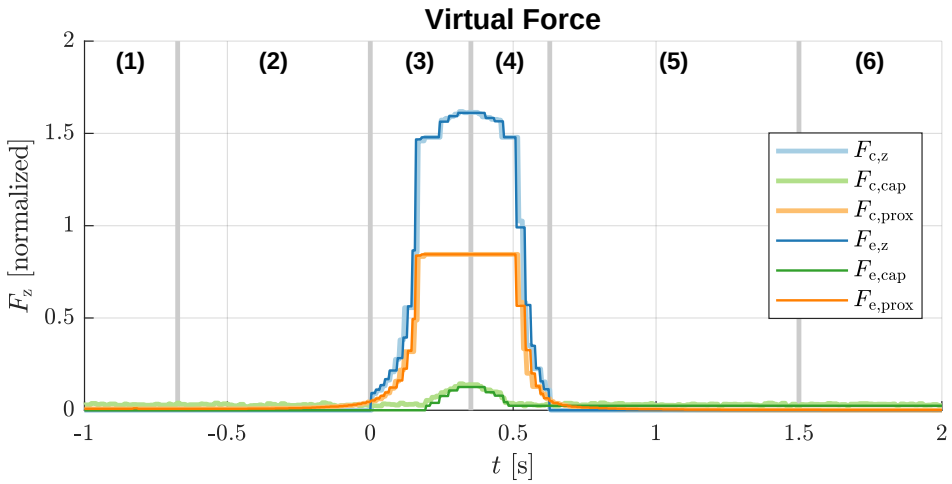
(6) The robot arm is no longer touched and the skin is idle. The movement of the robot arm slows down (friction).

Table 28 summarizes the system's states throughout these six phases.

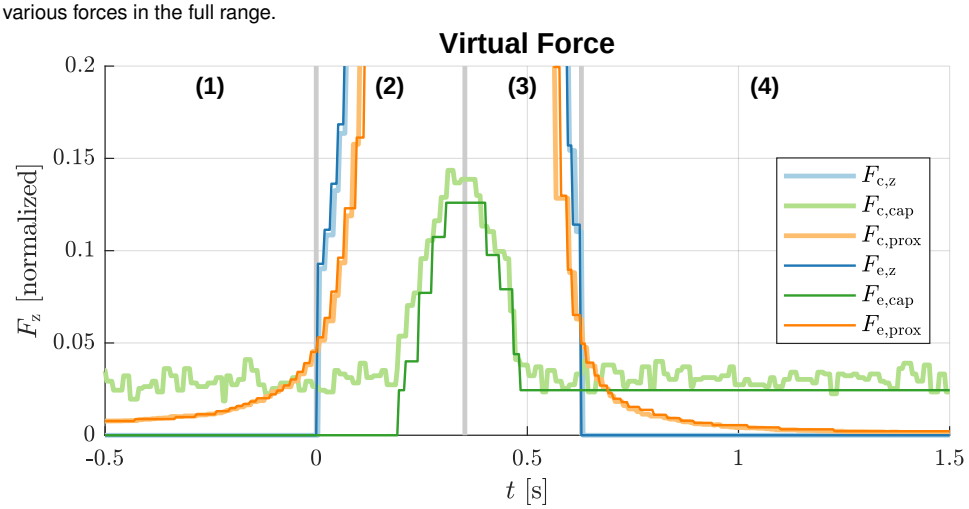
Phase	Mode	Network Traffic	Active Skin Cells	Active Joints
(1)	Clock-Driven	yes	no	no
(2)	Clock-Driven	yes	no	yes
(3)	Clock-Driven	yes	yes	yes
(4)	Clock-Driven	yes	yes	yes
(5)	Clock-Driven	yes	no	yes
(6)	Clock-Driven	yes	no	yes
(1)	Event-Driven	no	no	no
(2)	Event-Driven	yes	no	no
(3)	Event-Driven	yes	yes	yes
(4)	Event-Driven	yes	yes	yes
(5)	Event-Driven	yes	no	yes
(6)	Event-Driven	no	no	yes

Table 28 The system's state throughout the different experimental phases for the clock-driven and the event-driven operation mode. The network traffic addresses the traffic between e-skin and skin driver, and between skin-driver and controller. The number of active skin cells is computed according to Algorithm 3, and the joints are active when the robot arm moves (there are joint position events in the event-driven mode).

Figure 70 presents the total virtual force $F_{\mathrm{z}} = \sum_i F_{\mathrm{z},i}$ for all skin cells i and its sub-components of the skin cells' capacitive force and proximity sensors for both operation modes of the system. The similarity of the respective forces readings in clock- and event-driven mode prove, 1) the similarity of the tactile interaction, and 2) the similarity of the encoded tactile information. Thus, the experimental setup is valid since it ensures the comparability of the measurements when performing the experiment in clock- and event-driven modes. Furthermore, the similarity of the tactile information proves that the encoding error in event-driven mode is neglectable despite the mode's large reduction of network traffic.



(a) The various forces in the full range.



(b) The zoom-in to the interval $t \in [0.5, 1.5]$ s.

Figure 70 The total virtual force F_z, the capacitive force F_{cap}, and the proximity force F_{prox}. The subscripts c and e denote the clock-driven, or respectively, the event-driven operation mode. Thanks to the repeatable interaction and the neglectable encoding error of the event-driven system, the traces of the respective forces are practically identical. All forces are normalized (Section 5.1.2.1).

Figure 71 presents the joint velocities of the robot arm when it reacts to the tactile interaction. The respective velocities are practically identical for both operation modes. Thus, the clock- and event-driven controllers deliver the same result allowing for the comparison of their performance with respect to the CPU usage and network traffic indicators (Section 4.3.5).

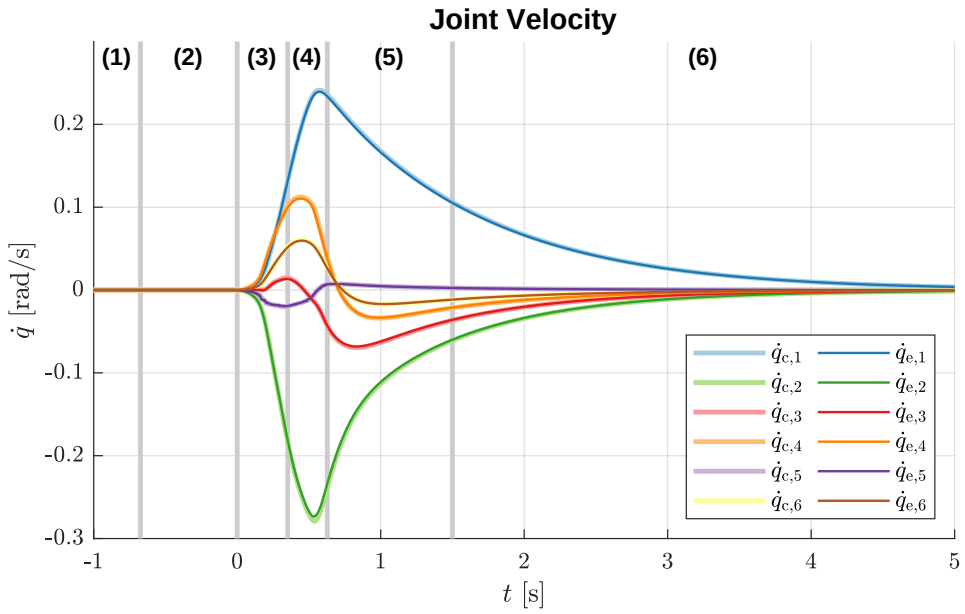


Figure 71 The joint velocities $\dot{\mathbf{q}}$ of the robot arm. The subscripts c and e denote the clock-driven, or respectively, the event-driven operation mode. The trajectories are practically identical for both operation modes proving the comparability of the controllers.

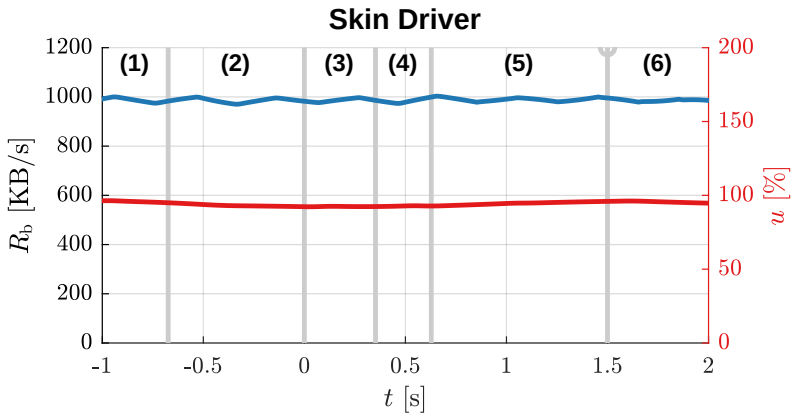
In summary, the controllers operate on identical tactile information and deliver identical reactions in both operation modes. The clock- and event-driven systems are comparable without the restriction of any kind. This direct comparability allows for the subsequent evaluation of the different controllers.

5.3.5.2 Controller Performance

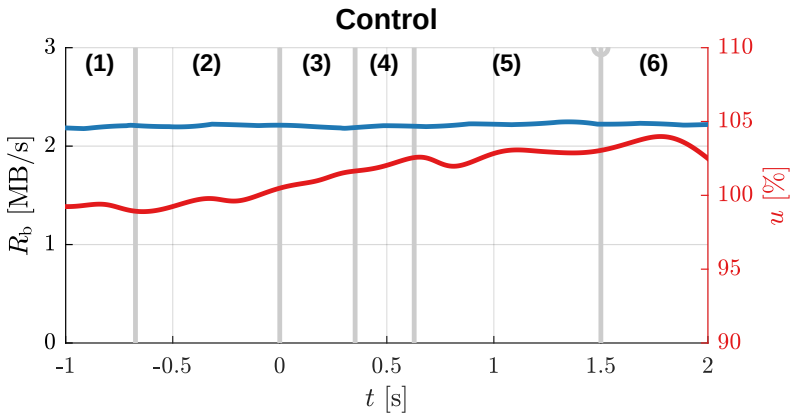
Figure 72 depicts the CPU usage and the network traffic for the skin driver and the controller. The figure's first row presents measurements for the clock-driven system. The second row presents the measurements for the event-driven system. The measurements for the idle system (experiment phase (1)) and the peak measurements are summarized in Tables 29 and 30.

Network Traffic	CD Idle	CD Peak	ED Idle	ED Peak
Skin Driver	1 MB/s	1 MB/s	2.5 kB/s	80 kB/s
Controller	2.25 MB/s	2.25 MB/s	5.6 kB/s	180 kB/s
Ratio	100 %	100 %	0.24 %	7.8 %

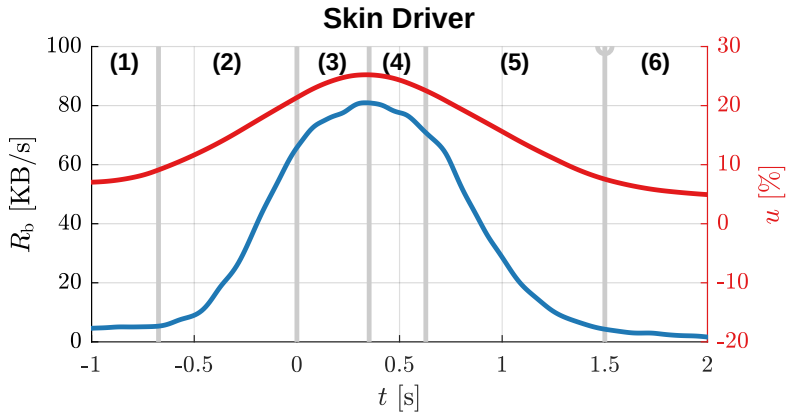
Table 29 The network traffic of the skin information for the system and its components in the clock-driven (CD) and the event-driven (ED) operation mode. The network traffic of the controller is 2.25 times larger than for the skin driver because of the messaging overhead in ROS (Appendix C).



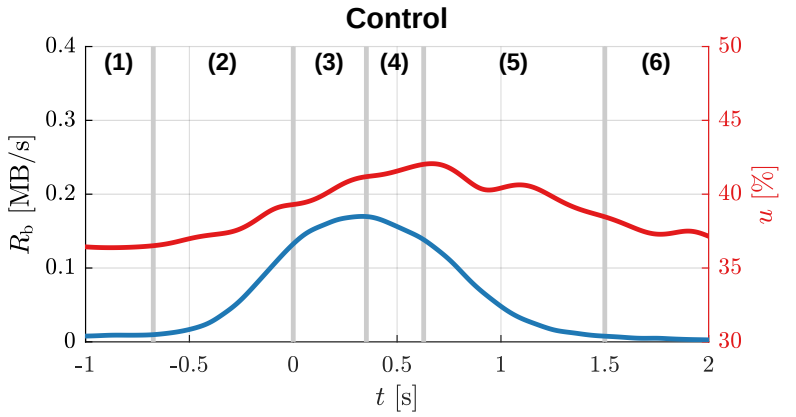
(a) The network traffic R_b (blue) and the CPU usage u (red) of the skin driver in clock-driven mode.



(b) The network traffic R_b (blue) and the CPU usage u (red) of the controller in clock-driven mode.



(c) The network traffic R_b (blue) and the CPU usage u (red) of the skin driver in event-driven mode.



(d) The network traffic R_b (blue) and the CPU usage u (red) of the controller in event-driven mode.

Figure 72 The network traffic R_b and the CPU usage u for the skin driver and the controller. The first row presents the measurements for the clock-driven system and the second row for the event-driven system. The network traffic of the skin driver is the network traffic in the Ethernet connection to the e-skin, and the network traffic of the controller is the network traffic in the ROS connection to the skin driver (Appendix C). The CPU usage of the skin driver and the controller are measured in the same way as in Section 5.1.1. Thus, they are comparable with Figures 53b and 53f, and respectively with Figures 53d and 53h.

CPU Usage	CD Idle	CD Peak	ED Idle	ED Peak
Skin Driver	95 %	95 %	5.5 %	25 %
Controller	99 %	103 %	36.5 %	41.6 %
Total	194 %	198 %	42 %	66.6 %
Ratio	100 %	100 %	**22 %**	**34 %**

Table 30 The CPU usage for the system and its components in the clock-driven (CD) and the event-driven (ED) operation modes.

Overall, the event-driven controller demonstrates a significant reduction of network traffic and CPU usage. The total reduction ratio of the CPU usage of the event-driven system is 78 % in the best case and 66 % in the worst case.

Table 31 summarizes the results for the standard controller introduced in Section 5.1 with the optimized controller of this section. In the clock-driven mode, the optimized controller significantly requires less CPU usage when it is idle, this is only 99 % in comparison to 180 %. This result demonstrates the effectiveness of Algorithm 5 (update-driven virtual force computation) since when it is idle, the controller mainly updates skin information in its memory and computes and monitors the number of active skin cells.

System	CPU Usage	CD Idle	CD Peak	ED/H Idle	ED/H Peak
Standard	**Controller**	180 %	200 %	82 %	122 %
	Total	260 %	280 %	99 %	150 %
	Ratio	100 %	100 %	**38 %**	**54 %**
Optimized	**Controller**	99 %	103 %	36.5 %	41.6 %
	Total	194 %	198 %	42 %	66.6 %
	Ratio	100 %	100 %	**22 %**	**34 %**

Table 31 The total CPU usage for the standard system of Section 5.1 in clock-driven (CD) and hybrid (H) operation mode, and the total CPU usage for the optimized system of this section in clock-driven and event-driven (ED) operation mode. Despite their different experimental setups, the measurements of both systems are roughly comparable. The number of active skin cells is the same. Thus the number of skin joint torques $\boldsymbol{\tau}_i$ to compute are the same and with that the increase in the peak measurements are comparable. The idle measurements of the controllers in clock-driven mode indicate the efficiency of monitoring the number of active skin cells.

Furthermore, comparing the peak CPU usages of both controllers in clock-driven mode, which is 103 % versus 200 %, proves the effectiveness of Algorithm 6 (optimized skin joint torque computation) and the optimized controller. The peak measurements indicate an increase of the number of active skin cells and thus in the number of skin joint torques $\boldsymbol{\tau}_i$ to compute. Since the number of active skin cells is the same in both experimental setups, a higher increase towards the peaks of CPU usage indicates that more CPU usage is demanded per active skin cell. With an increase of 20 % of the standard controller versus the

4 % of the optimized controller, the optimized controller consumes significantly less CPU usage per active skin cell.

Comparing the total CPU usages of all systems, the completely event-driven system presented in this section demonstrates its superiority with significant improvements. Not only does it show better ratios than the hybrid system, but the absolute CPU usages are also significantly lower, both for idle and peak readings. The reduction is more than two times in comparison to the hybrid event-driven system and more than four times in comparison to the standard clock-driven controller.

In summary, the presented optimized algorithms for computing skin joint torques in clock- and event-driven mode result in major performance improvements. The event-driven approach not only proves its effectiveness in processing tactile information in the stages before control, but it also underlines its potential in control algorithms. Event-driven control algorithms effectively reduce the number of computationally expensive computations, thus hugely reduce their CPU usage. Thus, the event-driven approach of this thesis is not only effective for large-area tactile perception, but it also allows for the efficiency in reactive large-area contact control, whereby it significantly mitigates the limitations of clock-driven large-area contact control.

5.4. Event-Driven Large-Area Skin System & Whole Body Control of an Autonomous Humanoid Robot

This section demonstrates that this thesis' event-driven LASS can scale-up to provide effective multi-modal tactile feedback for even highly complex control systems, such as the control of autonomous humanoid robots. As already pointed out in Section 4.6, a LASS such as deployed on the humanoid robot, H1, can only effectively work in event-driven operation mode. The clock-driven mode is not feasible. Furthermore, Section 5.1 showed that combining event-driven e-skin with standard clock-driven control is effective, and Section 5.3 demonstrated that clock-driven control algorithms can be optimized for event-driven tactile feedback. Altogether, these outcomes significantly contributed to the feasible utilization of LASSs in many advanced applications with very complex control systems. Section 5.4.2 briefly describes these applications and emphasizes the contribution of the event-driven LASS. All the presented systems have been published and used the autonomous humanoid robot, H1, as the experimental platform (Section 5.4.1). These all demonstrate the scalability of the approach presented in this thesis.

5.4.1. Event-Driven Large-Area E-Skin and the Humanoid Robot H1

This section briefly introduces the experimental platform, the humanoid robot, H1, for the works presented in Section 5.4.2. Since the following presentations focus more on H1 as a whole, rather than only on its large-area e-skin (Sections 4.3.4 and 4.6), this section summarizes its complete system.

Figure 73 depicts the humanoid robot H1 [29, 20] with the event-driven LASS. H1, embeds two computers, a battery, and has 30 actuators, which is 30 DOF. H1 is self-sufficient, that is, all computations are performed on the robot in its embedded computer systems, and all its systems are powered by its onboard battery. One computer is dedicated to acquiring the e-skin's tactile information and one for controlling the 30 joints. The robot's battery powers its actuators, its computers, and the deployed e-skin.

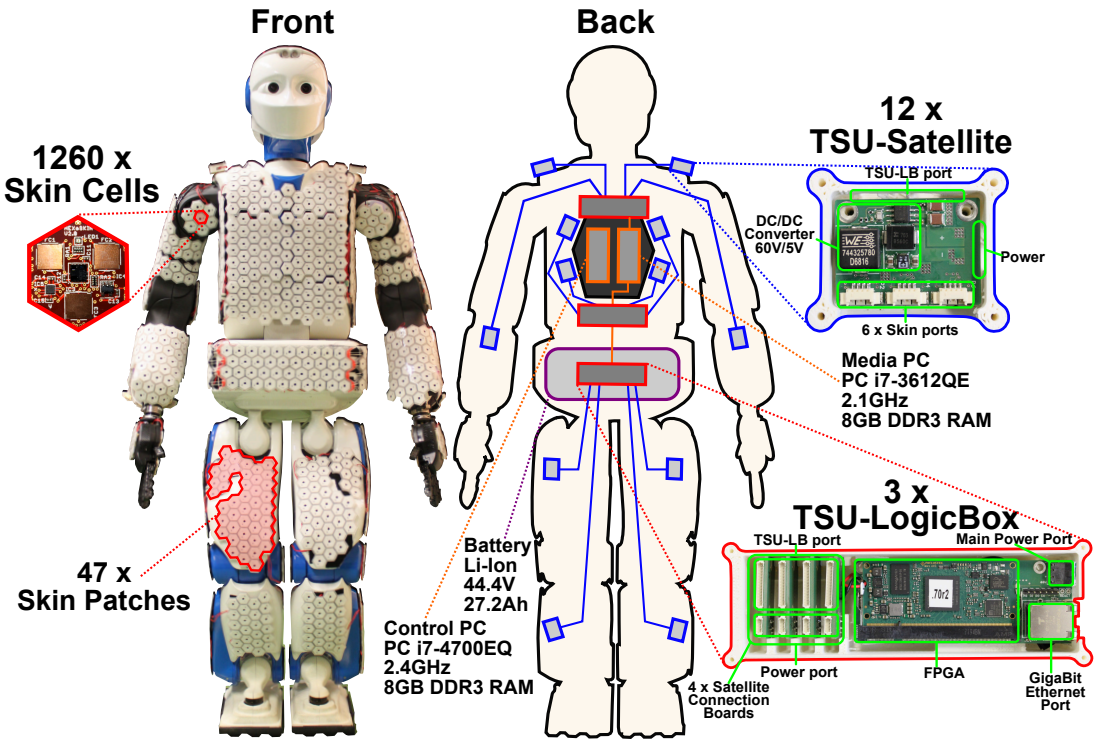


Figure 73 The humanoid robot, H1, and the deployed event-driven e-skin system [29, 20]. The self-sufficient robot employs two computers, 30 actuators, and a battery. Its battery powers all systems, and all computations are performed on the robot's computers. The battery has a capacity of 1.2 kW h. The e-skin system with all its components consumes around 86 W. More details on the e-skin system can be found in Section 4.3.4.

Realizing effective controllers for H1 is a challenging task, in particular when a controller requires tactile feedback. Both, controllers and e-skin, have a high demand for computational power, which is a limited resource in self-sufficient robots such as H1. The works presented in Section 5.4.2 not only demonstrate the effectiveness of event-driven e-skin systems in complex control systems such as H1. These works actually provide strong evidence that the contributions of this thesis, in particular the efficiency of the event-driven information handling for e-skin (Section 4.6) and the optimal exploitation of event-driven information in control (Section 5.3), deliver for the first time feasible large-area tactile feedback in very complex control architectures.

5.4.2. Event-Driven Large-Area E-Skin and Complex Control Systems

Figures 74 and 75 summarize the work this thesis contributed to [41]. The complex control system realizes whole-body active compliance control with a hierarchical control regime. The hierarchical controller fuses possibly antagonistic control goals resolving their priority to generate a consistent control behavior. For instance, the humanoid robot's first priority is to balance such that it does not fall. When the redundancy in actuation allows additional movements (e.g. some joints are currently not required for balancing) the controller concurrently executes controllers with lower priority, for instance a tactile-driven active compliance controller.

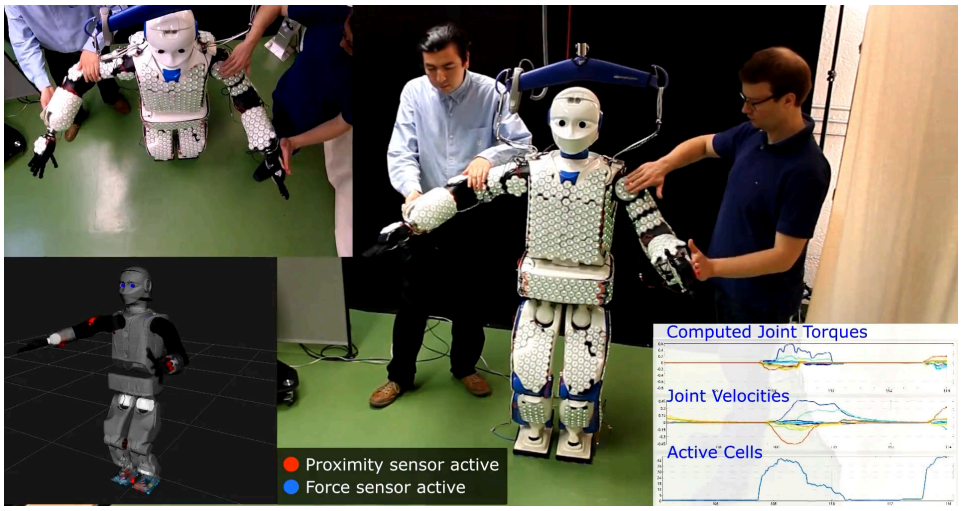


Figure 74 The whole-body active compliance controller [41]. The primary control task is to balance the robot such that it does not fall. The hierarchical controller concurrently executes control tasks with lower priority, e.g. the tactile-driven active compliance controller. As a result the robot balances and at the same time reacts to tactile interactions.

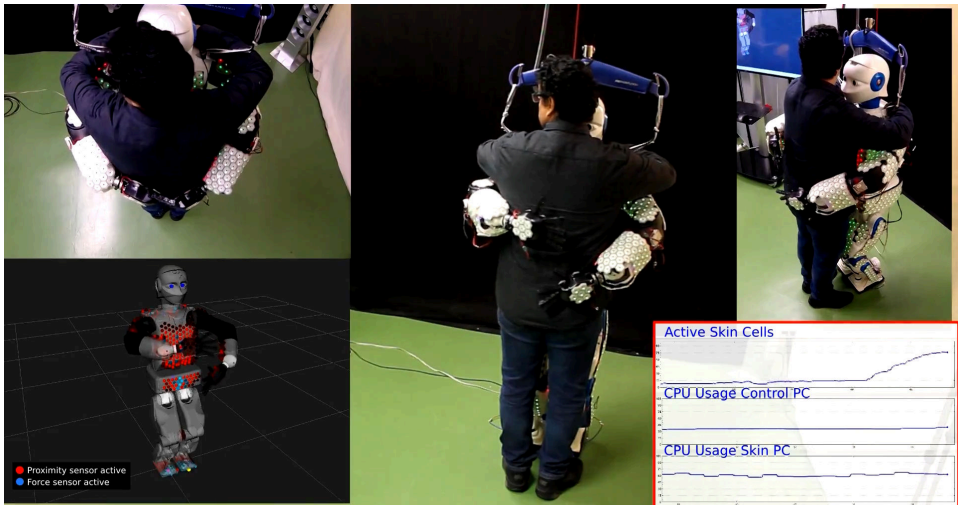


Figure 75 The whole-body active compliance controller [41]. The event-driven skin system and the optimized tactile-driven active compliance controller can handle a large number of active skin cells.

The work [62] depicted in Figure 76 extends the whole-body control hierarchy with a pressure-driven compliance controller. The pressure-driven compliance controller not only considers contact forces but also contact areas. That is, the robot reacts faster and stronger to large forces in small areas than it reacts to small forces in large-areas.

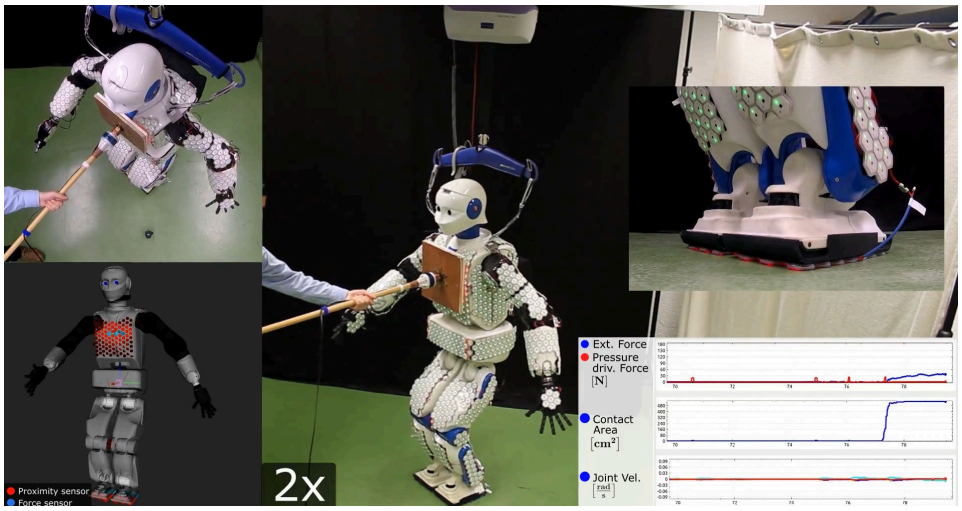


Figure 76 The pressure-driven compliance controller [62]. The event-driven skin system and the optimized tactile-driven active compliance controller can handle a large number of active skin cells.

The following up works of [61, 60] and [77, 78], depicted in Figures 77 and 78, exploit the event-driven LASS in the complex field of humanoid locomotion and the fusion of humanoid locomotion with interaction, for instance in obstacle discrimination and avoidance in the footfall [61, 60] and dance with a human [77, 78].

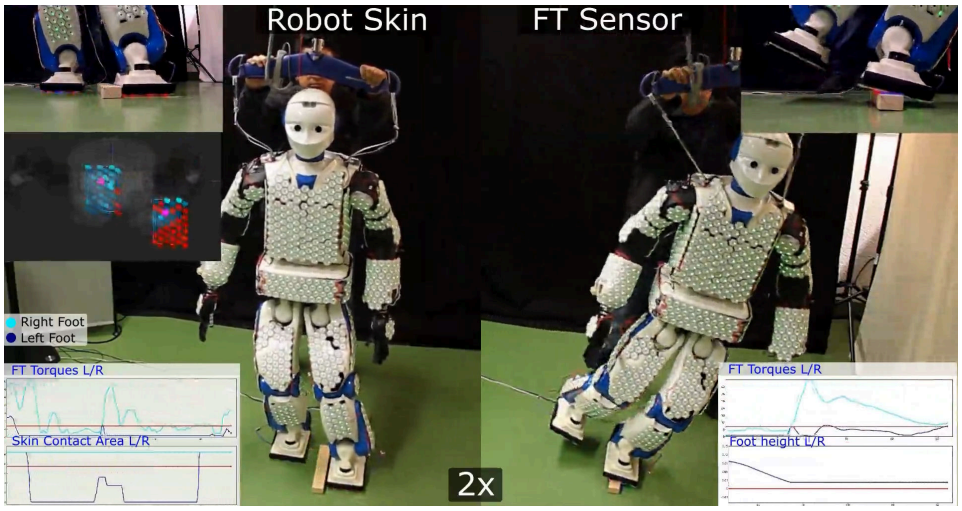


Figure 77 The skin-driven obstacle avoidance in humanoid locomotion [61, 60]. The skin enables the robot to detect obstacles in the footfall that can not be detected by other sensors, for instance, Force-Torque (FT) sensors.

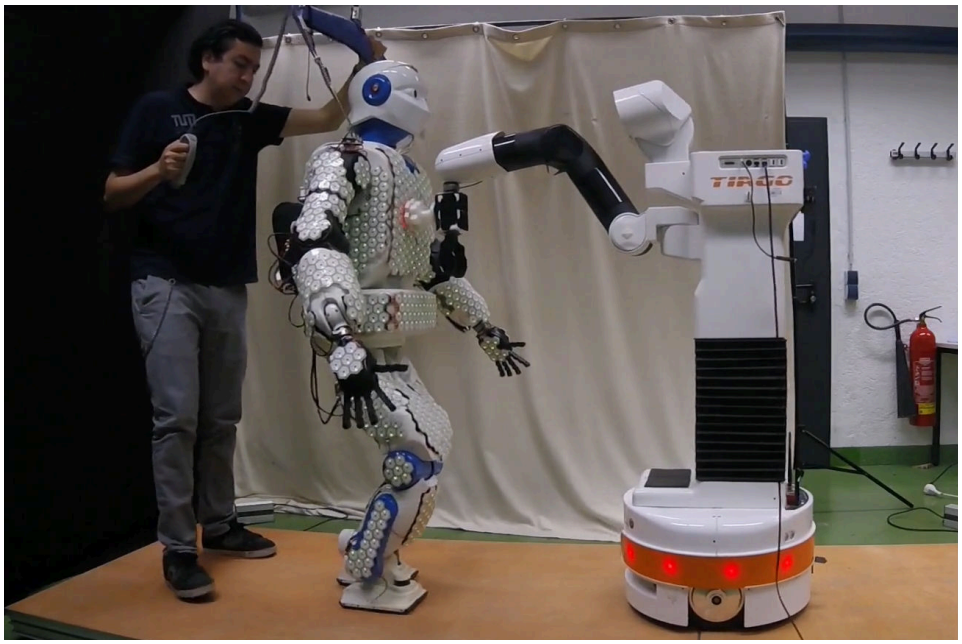


Figure 78 The leader-follower control framework for complex tactile interactions during humanoid locomotion [77, 78]. The LASS supports the framework with feature reach whole-body contact information.

This thesis' contributions, in the form of the efficient large-area e-skin system (Section 4.3), and the optimized processing of event-driven skin information in clock-driven control (Section 5.3.3) had a significant impact on the feasibility of tactile large-area feedback in all these presented works [41, 62, 61, 60, 77, 78]. The standard clock-driven approach for LASSs is not effective for perceptional tasks and fails to provide tactile feedback in control. Furthermore, early experiments in [41] revealed that the inefficient looping through all 1260 skin cells in clock-driven control (without the optimization provided by Algorithm 5) is not feasible, even in a hybrid event-driven system with an event-driven LASS. The controller continuously broke its real-time constraints and became unstable. Optimizations exploiting the event-driven tactile feedback were strictly required to solve this issue.

In summary, this thesis significantly contributed to the previously infeasible and very demanding applications of LASS in whole-body controllers.

5.5. Summary

This chapter presented the realization and validation of applications that exploit the tactile information of LASSs and generate appropriate reactions and behaviors. The presented systems implement a reactive behavior to tactile interactions and form a perception-action loop. The fundamental principle of event-driven systems, which is to drive the handling of information on demand (Section 3.3), plays also a central role in computationally expensive control algorithms. As evidenced with the results of Section 5.2, the supported number of active skin cells and the number of all skin cells is limited by the efficiency of the control algorithm and the real-time constraints of the controller. A more efficient control algorithm can support larger skin systems and larger contact areas. The comparative results of Sections 5.1 and 5.3, and the experience collected in realizing tactile whole-body control (Section 5.4) evidence the effectiveness of this thesis' event-driven approach. They even demonstrate that clock-driven controllers implementing large-area tactile interaction have to exploit the nature of the event-driven tactile information to increase efficiency by avoiding unnecessary computations and to eventually meet the real-time constraints. Optimized hybrid event-driven systems significantly boost performance as far as by a factor of two and enable previously infeasible tactile control architectures. Purely event-driven tactile control systems require the deep analysis of control algorithms to transform them from clock-driven to event-driven computations. Nevertheless, the analysis of an optimized event-driven reactive contact controller (Section 5.3) demonstrates a tremendous performance improvement up to a factor of four.

Overall, this chapter presented approaches for integrating and utilizing the tactile feedback of event-driven large-area e-skin in very complex control systems that, for the first time, led to tactile-driven whole-body interactions with large contact areas. This thesis' event-driven approach for large-area e-skins, together with the presented efficient integration methods, contribute the required boost in computational performance. Both, this thesis' efficient event-driven approach for e-skins, and its effective, efficient, and flexible integration methods in complex systems, allow systems that rely on tactile feedback to scale, whereby previous approaches failed.

6. Conclusion

6.1. Summary and Conclusions

This thesis presented novel methods based on biologically plausible principles to efficiently handle the large amount of tactile information arising in large-area e-skin. Chapter 1 introduced the motivations and challenges of realizing efficient e-skins that can eventually endow machines with human-like whole-body tactile interaction capabilities, along with the contributions of this thesis. Then, Chapter 2 provided the background and connection to related works. This chapter introduced the neuroscientific view of the sense of touch, and the engineering challenges e-skin systems needed to mitigate or solve. Furthermore, it presented the information handling system from the perspectives of neuromorphic engineering, signal processing, and information theory, and novelty-driven systems. The eventual comparative study of existing event representations and protocols determined the most flexible and applicable approach for realizing the novelty-driven principle for large-area e-skins. Chapter 3 drew from this background and homogenized the fundamentals of novelty-driven systems. These fundamentals delivered the theory for the correct design and optimal parameterization of novelty detectors for the e-skin's sensors to optimize for sensitivity, transmission rate, and to minimize encoding errors and noise. The theory actually delivered a novel tuning guideline for the correct parameterization of tactile event generators. Another outcome of these fundamentals was a model for the computational demand of novelty-driven e-skin systems. This model later allowed for the extrapolation of evaluation results to larger e-skin systems. Furthermore, the fundamentals provided the foundations for comparing the performance of novelty-driven e-skin systems with standard clock-driven reference systems. This comparability has been essential in the evaluations of the approaches taken in this thesis. Then, Chapter 4 presented the design methods developed for the systematic realization of the novelty-driven approach in existing e-skins and computer systems. This chapter subsequently demonstrated the realization of the designs in an e-skin system with more than 10 000 multi-modal tactile sensors. The realization validated the feasibility of the designs. The experimental evaluation that followed additionally demonstrated the efficiency of the novelty-driven approach in e-skin systems. The evaluation evidenced a significant boost in performance that constituted a large reduction of transmission rates and computational demands. This improvement demonstrated, for the first time, the effective handling of tactile information in a large-area e-skin with more than 10 000 sensors. Eventually, Chapter 5 presented very complex systems that utilize the feedback of large-area contacts. The successful integration of the novelty-driven large-area e-skin in robotic systems demonstrated its efficiency and effectiveness in complex reactive control algorithms for interactions. These integrations additionally delivered general methods for the efficient connection of novelty-driven e-skin with standard clock-driven control algorithms (hybrid event-driven systems), the efficient exploitation of novelty-driven information in clock-driven algorithms, and the transformation of clock-driven to event-driven control algo-

rithms. Altogether, these methods provided the foundations for experiments with a humanoid robot utilizing whole-body tactile feedback. The experiments demonstrated the system's efficiency to scale, whereby previous approaches failed, allowing very challenging utilization of large-area e-skin in whole-body controllers.

This thesis will have a long lasting impact on the design, integration, and information handling of large-area e-skin systems. Three key contributions achieve this impact.

A Systematic Approach for Efficient Information Handling in Large-Area E-Skin This thesis contributes a systematic approach for realizing efficient information handling in large-area e-skin. The contributed approach is biologically plausible and enables novelty-driven e-skin systems. The approach improves the information handling efficiency in large-area e-skins significantly. This thesis also contributes the fundamentals for its approach. These fundamentals provide the theoretical basis for efficient novelty-driven designs, comprehensive evaluations, and well-founded extrapolations for larger e-skin systems. Furthermore, this thesis contributes homogeneous fundamentals for novelty-driven systems that will hopefully foster the exchange between the field of neuromorphic engineering and the field of signal and control theory. This exchange will lead to faster novelty-driven algorithms for standard computer systems, and to more applications and better integrations of neuromorphic systems in complex systems, for instant, humanoid robots.

A Realization of Flexible Information Handling for Large-Area E-Skin Systems This thesis contributes flexible, novelty-driven, and hardware-independent designs that lead to rapid integrations in existing e-skin systems and connections to complex calculations that rely on tactile feedback. These designs immediately boost performance by lowering demands on communication bandwidth and computational power. Furthermore, this thesis contributes the realization of its designs in a large-area e-skin system with more than 10 000 multi-modal tactile sensors. This e-skin system is capable of providing effective tactile feedback at this scale, where all previous approaches failed. With this contribution, this thesis will impact the design of future large-area e-skins that will provide higher spatio-temporal resolutions, larger sensing areas, and higher coverage factors.

A Realization of Efficient Tactile Information Handling in Complex Control Systems This thesis contributes efficient methods for integrating the tactile feedback of large-area e-skin systems in complex control algorithms, for instance, in reactive contact control. Thereby this thesis significantly contributes to the realization of previously infeasible tactile-driven whole-body interactions. The integration methods, this thesis contributes, remove previous limitations for contact areas and temporal resolutions in the feedback of large-area e-skin. With the successful realization of these methods, this thesis demonstrates the feasibility of realizing whole-body control in a humanoid robot covered with large-area e-skin. This whole-body control is capable to react to the tactile feedback of an e-skin with more than 10 000 sensors, whereby all previous approaches failed.

Overall, this thesis contributes, for the first time, efficient large-area tactile feedback allowing for novel applications of large-area e-skin whereby all previous approaches failed. Thereby, this thesis will hopefully positively impact a novel trend, where increasingly more autonomous machines successfully exploit the sense of touch and thus improve their physical interaction capabilities with their surroundings.

6.2. Outlook

This thesis contributed to the advancement of large-area e-skin systems for human-like whole-body tactile interaction capabilities, which opens up new perspectives with the potential to further elevate the scalability of large-area e-skin systems. Three directions are particularly promising, these are: 1) Improving the robustness of the event representation can contribute to counter the rise of communication uncertainties in increasingly large systems, 2) The shaping of tactile densities to fit the requirements of tactile interaction in specific regions can contribute to increase the size of covered areas and improve interaction performance, 3) Shape changing e-skin can improve conformity to complex surfaces with high curvatures or non-rigid support contributing to increase the overall covered surface area.

Event Shaping targets to improve the robustness of event representations. Improved robustness can contribute to improving the scalability of very large e-skin systems that inherently have higher communication uncertainties caused by longer wires, more interfering components, and more points of failure. The idea for realizing event shaping emerges from observations in nature. We, humans, register the absence of sensory information by a feeling of numbness. However, then how can we differentiate between an absence of novel information and absence caused by a loss in communication? A probable answer, therefore, could be controlled redundancy. For instance, the information consumer could expect or predict specific patterns, and if these expectations or predictions are not met then information has been lost. Drawing from the interpretation of novelty detection in Section 3.1.1.3 as supervisors of predictors, more robust event creation schemes could be devised to add controlled redundancy to the stream of events.

Tactile Density Shaping targets to improve the scalability of e-skin by deployment patterns that reflect the spatial acuity required in different skin regions. Human skin realizes this idea and shapes the deployment density of its tactile sensors according to the requirements to distinguish the location and shape of contact points in different areas [55]. The density is highest in the fingertips and the lips, and lowest on our back. An e-skin's skin cells (Section 4.3.1.1) could be shrunk to better fit varying applications. Smaller skin cells increase the spatial acuity and allow for covering surfaces with higher curvature. Thus, the sensing density would improve and more surfaces would be feasible to be covered with e-skin. The vision is a network of skin cells (Section 4.3.2) with interchangeable skin cells of different sizes. So far, tactile density shaping has not been feasible due to the lack of an approach for efficiently

handling information in e-skin with larger numbers of sensors. This thesis' efficient novelty-driven approach for large-area e-skin now renders the investigation of tactile density shaping possible.

Shape Changing E-Skin addresses the challenges to advance from flexible to stretchable large-area e-skin without obstructing the approaches and strategies that mitigate and solve the challenges of large-area e-skins. Among these, the approach that enables the self-localization of tactile sensors is of particular importance [100, 102]. A stretchable large-area e-skin with the capability to self-localize its tactile sensors would not only increase the e-skin's conformability, but it would also, for the first time, allow to directly differ between contact forces and stretch and measure them at the same time in large-areas without the need to solve complex computational models for materials and sensing. The sensing and localization of large-area contacts on deformable surfaces would enable many novel applications, for instance in soft robotics, wearables, and smart objects. Investigations in this direction have so far been limited due to the larger number of sensors involved. This thesis' approach for efficient information handling in large-area e-skin now supports investigations in this direction.

A. E-Skin Interfaces

The e-skin system presented in this thesis (Section 4.3) uses interfaces that bridge the skin cell network to Ethernet networks and convert skin packets to standard UDP packets and vice versa.

The first generation of interfaces are referred to as the Tactile Section Units (TSUs) [101, 109], see Figure 79. One TSU interfaces between 5 UART connections and one 1 Gbit/s Ethernet connection and can power up to 300 skin cells (8.3 kpacket/s). Thus, instead of 5 UART connection one Ethernet cable plus power can support 300 skin cells. Multiple TSUs can be connected to the same 1 Gbit/s Ethernet network via standard Ethernet switches allowing for great flexibility when scaling up to large-area networks with several hundreds of skin cells.

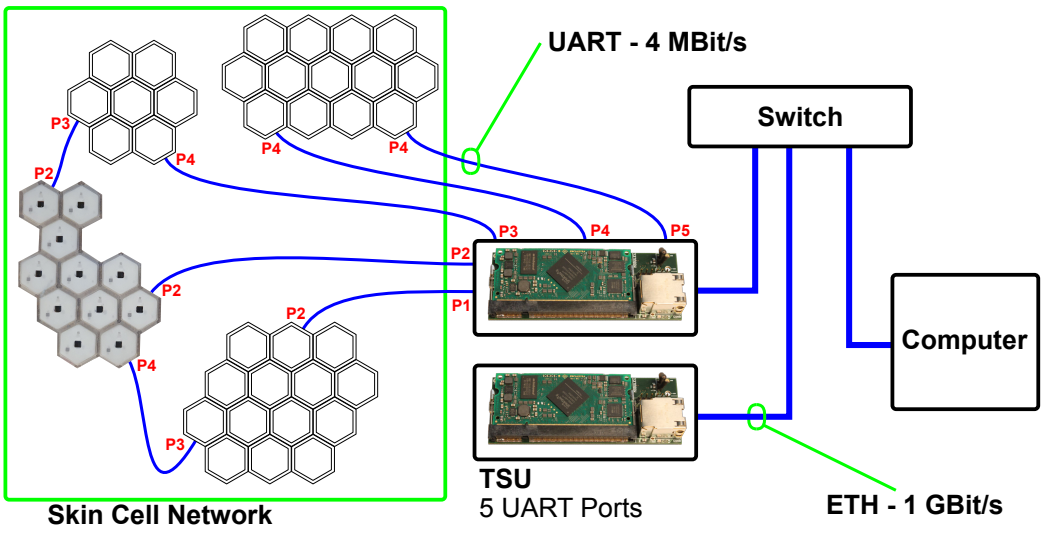


Figure 79 Tactile Section Unit (TSU) with network of skin cells.

The second generation of interfaces enhances the TSUs with focus on improving modularity, merging connections, and providing power closer to the skin patches. The TSUs are too bulky to be placed close to the skin patches. The most practical setup demands long connections between skin patches and TSUs since the TSUs are placed close to the computer system that acquires and consumes the information of the e-skin. This setup exhibits two major drawbacks. First, the setup requires many long cables that are susceptible to damages and require a lot of space. Second, the power lines in the cables demand low impedance to avoid voltage drops in the low voltage (5 V) power distribution system of the skin cell network. Often setups with long cables and large networks lead to instabilities due to large voltage drops and insufficiently powered skin cells. Solving these issues, the second generation of interfaces divide the functionalities of a TSU to a Tactile Section Unit Logic Box (TSU-LB) that is shared by up to four Tactile Section Unit Satellites (TSU-Ss) [20], see Figure 80.

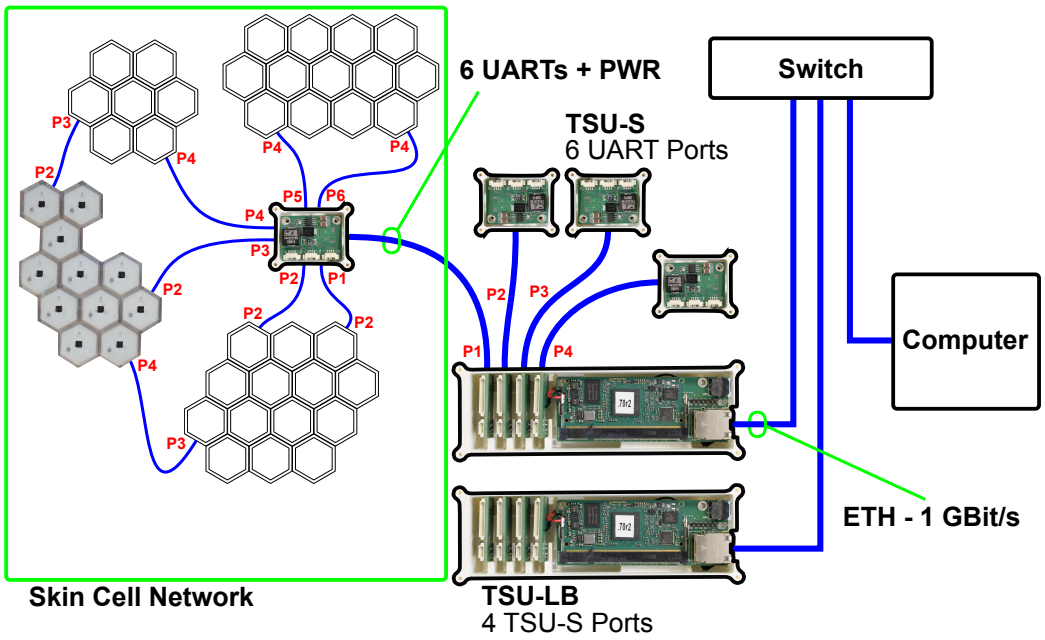


Figure 80 Tactile Section Unit Satellite (TSU-S) and Tactile Section Unit Logic Box (TSU-LB) with network of skin cells.

A TSU-S locally converts a high supply voltage of up to 60 V to the lower supply voltage of the skin cell network. Furthermore, the TSU-S bundles the high voltage power line and six UART connections to one single cable. The TSU-Ss are slim enough to be placed next to the skin patches, effectively reducing the number of cables from the distributed skin patches to the centralized computer by a factor of six. The higher supply voltage mitigates voltage drops in long cables and increases the stability of large skin cell networks. The Tactile Section Unit Logic Box (TSU-LB) provides the logic to interface between four TSU-Ss and a 1 Gbit/s Ethernet connection. A single TSU-LB effectively provides a interface to 24 UART connections, or respectively to at least 1.400 skin cells (400 kpacket/s).

Ethernet networks and UDP packets are standards and supported by many computer systems, easing the access to information provided by the e-skin system. The price of providing information in a generalized protocol is overhead. A UDP packet containing the 20 B of a packet in the skin cell network occupies 62 B in the Ethernet frame, and 86 B (24 B in addition for the preamble, the start of frame delimiter, the CRC checksum, and the IPG) on the Ethernet bus. A 1 Gbit/s Ethernet connection can thus theoretically provide the communication bandwidth for up to 6242 skin cells (1.56 Mpacket/s).

B. Requesting and Yielding Computation Time

Programs, whether they are clock- or event-driven, often run in conditions where they have to wait. They need to wait until information is available or can be forwarded to the next stage. Clock-driven programs may need to wait to synchronize to a desired sampling rate, while event-driven programs may need to sleep until novel information arrives. Ideally, all programs should wait in such a way that the operating system can switch context, that is, pause the current task and schedule another process/thread as soon as a program enters a waiting condition. In this way no computation time is wasted while waiting. Programs can usually wait in three different ways, by

1. Busy-Waiting,
2. Timed-Waiting, and
3. Signaled-Wakeup.

Busy-Waiting, often also called active waiting or polling, describes a program that waits for a condition/flag, by repeatedly reading it until it is true, see Algorithm 9.

Algorithm 9 Busy-Waiting

```
1: while flag == false do
2:     # do nothing, waste time
3: end while
```

The thread executing this program never notifies the operating system that it is waiting and that it could yield its allotted timeslice for other, more important tasks. Actually, a thread that never yields is considered as *greedy*, since the operating system assigns this thread as much computation time as possible and the thread consumes all of it, if needed or not. A greedy thread never stops before it consumed its timeslice. Busy-Waiting is wasting the limited resource computation time, is thus extremely inefficient, and should be avoided.

Timed-Waiting is more efficient, since the thread notifies the operating system how long it can manage without computation time. In contrast to Busy-Waiting, Timed-Waiting checks the flag/condition with a rate defined by the sleep time T_s, rather than checking it as fast as possible, see Algorithm 10.

The work presented in Appendix B was in part published in:
Bergner, F., Dean-Leon, E., Cheng, G., "Design and Realization of an Efficient Large-Area Event-Driven E-Skin". In: *Sensors* 20.7 (2020), p. 1965.

Algorithm 10 Timed-Waiting

```
1: while flag == false do
2:     sleep(T_s)                          # yield and reschedule after T_s
3: end while
```

The sleep function yields the timeslice of the thread and defines the time when it is scheduled next. Timed-Waiting is not wasting computation time and is an efficient waiting principle for clock-driven programs with a defined sampling frequency.

Signaled-Wakeup is realized in programs that wait for signals of the operating system rather than for flags. These signals can be defined and triggered by other processes/threads, or are provided by the operating system. The operating system can provide signals that are triggered by hardware interrupts, for example, the arrival of network packets, and so forth. Exploiting the Signaled-Wakeup principle in programs is even more efficient than Timed-Waiting. Employing Signaled-Wakeup, a thread yields and notifies the operating system that it is waiting for a signal, see Algorithm 11.

Algorithm 11 Signaled-Wakeup

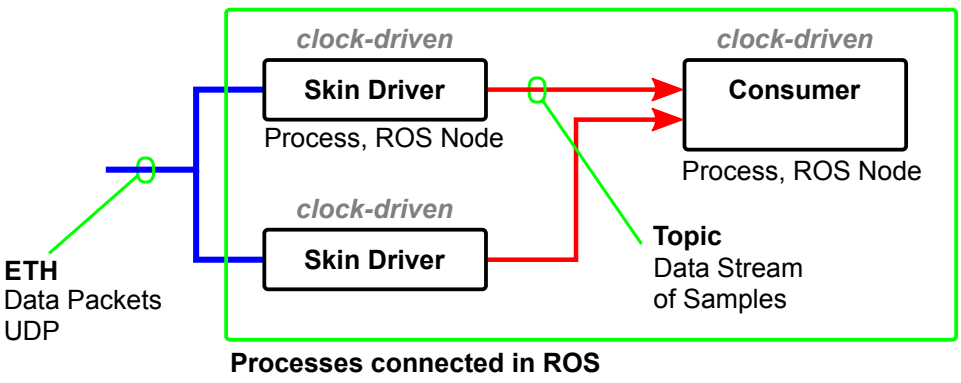
```
1: # do something
2: select(signal)                          # yield and reschedule on signal
3: # continue
```

Rather than repeatedly reassigning timeslices for the program to read a flag, or check a condition, as in Timed-Waiting, the operating system only returns to the thread on the occurrence of the signal. Obviously, the proper exploitation of the Signaled-Wakeup principle enables us to realize fast event-driven information handling in standard computing systems with low waiting overhead.

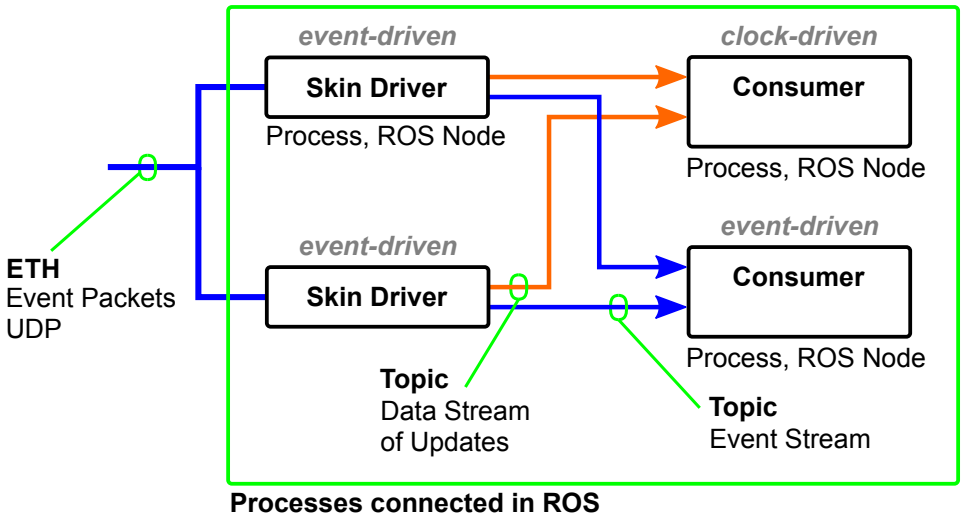
C. Event-Driven E-Skin Software Framework

This thesis' implementation of a flexible and modular information handling system for large-area e-skins follows the design descriptions of Section 4.2.1. The software architecture builds upon the ROS middleware and the Qt software framework. ROS not only eases the integration of e-skin in robotic systems, ROS also provides a standardized communication protocol that eases information exchange between processes (nodes in ROS terminology). The communication protocol bases on socket connections which match well with the signaling principles required in event-driven communication as depicted in Section 4.2.1. Thus, exploiting ROS provides the software system for the e-skin with modularity, an interface to robots, and event capable inter-process communication. Additionally, the Qt software framework provides a sophisticated and high performance event messaging system for inter-thread communication with event dispatchers. These features of Qt provide the necessary support for implementing event-driven programs within the processes of our e-skin software framework.

The e-skin software architecture implements and provides the modules for the event-driven information handling system as depicted in Figure 24 of Section 4.2.1. Each module is a separate process (ROS node) and all modules communicate via the ROS communication system (topics, services). The system incorporates *skin driver* processes, and clock-driven or event-driven *information consumer* processes, see Figure 81. Information consumer processes are visualization programs, control programs, information extraction programs for higher-level information, etc.



(a) E-Skin in clock-driven operation mode.



(b) E-Skin in event-driven operation mode. Hybrid and event-driven setups.

Figure 81 The e-skin information handling system on the computer. The processes (ROS nodes) are connected utilizing the inter-process messaging system of ROS.

The skin driver processes bridge between the peripheral e-skin system with its skin cells and communication interfaces (TSU,TSU-S,TSU-LB) and the information handling system on the computer. The e-skin system employs one skin driver for each TSU or TSU-S. For example, the e-skin system on H1 presented in Section 4.3.4.1 employs 12 skin driver processes. The assignment of one skin driver per TSU or TSU-S not only improves modularity (on H1, a TSU-S is associated to a body part, see Table 5), it also distributes computational load, avoiding the saturation of threads.

A skin driver processes contains several functional blocks, see Figure 82. The packet unpacker block unpacks the data packets or, respectively, the event packets sent by the skin cells. For event packets the packet unpacker implements Algorithm 2. Data packets are decompressed and the sensor values are stored along with the ID of the skin cell into a skin cell values data structure. Event packets are unpacked to event data structures that contain the ID of the skin cell, the value of the sensor, and the type of the sensor.

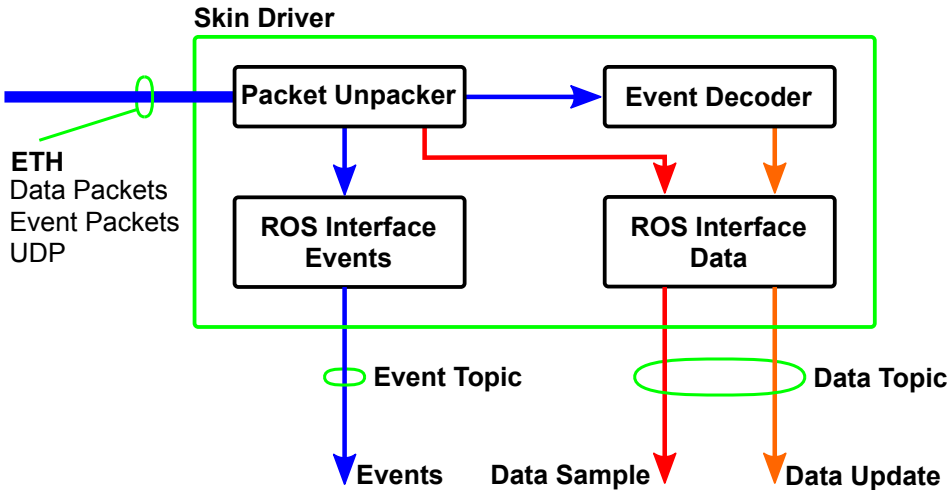


Figure 82 The functional block of a skin driver process. The skin driver bridges between the e-skin and the consumer processes and operates either in clock-driven or event-driven mode. The skin driver provides a communication link for hybrid systems where the skin driver is event-driven and the consumer is clock-driven.

The event decoder block converts events to skin cell value data structures for clock-driven data consumers, following the design of the event decoder presented in Figure 26 of Section 4.2.2. The implemented event decoder keeps a skin cell value data structure of each skin cell in memory. An arriving event updates the sensor value in memory according to the skin cell ID and the sensor type of the event. The event decoder block forwards a complete skin cell value data structure whenever at least one of its values is updated by an event.

The ROS interface blocks provide access to the information stream of events and skin cell data. The event stream contains the event data structures provided by the packet unpacker. The data stream contains the skin cell value data structures either originating directly from the packet unpacker or, when the e-skin is operating in event-driven mode, from the event decoder. In clock-driven mode, the data stream is clock-driven and in event-driven mode the data stream is update-driven, that is, it only contains the skin cell value data structures that need to be updated (sample rate/event update) in the memory of the clock-driven consumers. Through the update-driven data stream, hybrid event-driven systems can partially profit from the event-driven approach up to the clock-driven consumer, as demonstrated in Section 5.1.

D. Copyright Permissions

Rightslink® by Copyright Clearance Center https://s100.copyright.com/AppDispatchServl...

Copyright Clearance Center RightsLink®

Home | Help | Email Support | Sign in | Create Account

IEEE Requesting permission to reuse content from an IEEE publication

Event-based signaling for reducing required data rates and processing power in a large-scale artificial robotic skin

Conference Proceedings:
2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

Author: Florian Bergner

Publisher: IEEE

Date: Sept. 2015

Copyright © 2015, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

© 2020 Copyright - All Rights Reserved | Copyright Clearance Center, Inc. | Privacy statement | Terms and Conditions
Comments? We would like to hear from you. E-mail us at customercare@copyright.com

1 of 1 08/07/2020, 16:09

Figure 83 Copyright permissions for [21], Bergner et al. 2015.

Figure 84 Copyright permissions for [16], Bergner et al. 2016.

Figure 85 Copyright permissions for [17], Bergner et al. 2017.

Figure 86 Copyright permissions for [18], Bergner et al. 2018.

Figure 87 Copyright permissions for [20], Bergner et al. 2019.

MDPI | Rights & Permissions

Copyright and Licensing

For all articles published in MDPI journals, copyright is retained by the authors. Articles are licensed under an open access Creative Commons CC BY 4.0 license, meaning that anyone may download and read the paper for free. In addition, the article may be reused and quoted provided that the original published version is cited. These conditions allow for maximum use and exposure of the work, while ensuring that the authors receive proper credit.

In exceptional circumstances articles may be licensed differently. If you have specific condition (such as one linked to funding) that does not allow this license, please mention this to the editorial office of the journal at submission. Exceptions will be granted at the discretion of the publisher.

Reproducing Published Material from other Publishers

It is absolutely essential that authors obtain permission to reproduce any published material (figures, schemes, tables or any extract of a text) which does not fall into the public domain, or for which they do not hold the copyright. Permission should be requested by the authors from the copyrightholder (usually the Publisher, please refer to the imprint of the individual publications to identify the copyrightholder).

Permission **is required** for:

1. Your own works published by other Publishers and for which you did not retain copyright.
2. Substantial extracts from anyones' works or a series of works.
3. Use of Tables, Graphs, Charts, Schemes and Artworks if they are unaltered or slightly modified.
4. Photographs for which you do not hold copyright.

Permission **is not required** for:

1. Reconstruction of your *own* table with data already published elsewhere. Please notice that in this case you must cite the source of the data in the form of either "Data from..." or "Adapted from...".

Figure 88 Copyright permissions for [19], Bergner et al. 2020. Page 1.

2. Reasonably short quotes are considered *fair use* and therefore do not require permission.
3. Graphs, Charts, Schemes and Artworks that are completely redrawn by the authors and significantly changed beyond recognition do not require permission.

Obtaining Permission

In order to avoid unnecessary delays in the publication process, you should start obtaining permissions as early as possible. If in any doubt about the copyright, apply for permission. MDPI cannot publish material from other publications without permission.

The copyright holder may give you instructions on the form of acknowledgement to be followed; otherwise follow the style: "Reproduced with permission from [author], [book/journal title]; published by [publisher], [year].' at the end of the caption of the Table, Figure or Scheme.

Figure 89 Copyright permissions for [19], Bergner et al. 2020. Page 2.

Bibliography

[1] Abraira, V. E., Ginty, D. D., “The Sensory Neurons of Touch”. In: *Cell* 79 (2013), pp. 618–639.

[2] Albini, A., Denei, S., Cannata, G., “Enabling natural human-robot physical interaction using a robotic skin feedback and a prioritized tasks robot control architecture”. In: *IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. 2017, pp. 99–106.

[3] Albini, A., Denei, S., Cannata, G., “Towards autonomous robotic skin spatial calibration: A framework based on vision and self-touch”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 153–159.

[4] Amaral, D. G. “Principles of Neural Science, Fifth Edition”. In: ed. by Eric R. Kandel, James H. Schwartz, Thomas M. Jassell, Steven A. Siegelbaum, and A. J. Hudspeth. The MacGraw-Hill Companies: New York City, United States, 2013. Chap. 16: The Functional Organization of Perception and Movement, pp. 356–369.

[5] Anderson, J. D., Lee, D.-J., Edwards, B., Archibald, J., Greco, C. R., “Real-time feature tracking on an embedded vision sensor for small vision-guided unmanned vehicles”. In: *International Symposium on Computational Intelligence in Robotics and Automation*. Jacksonville, United States, 2007, pp. 55–60.

[6] Asfour, T., Regenstein, K., Azad, P., Schroder, J., Bierbaum, A., Vahrenkamp, N., Dillmann, R., “ARMAR-III: An integrated humanoid platform for sensory-motor control”. In: *IEEE-RAS International Conference on Humanoid Robots*. Genova, Italy, 2006, pp. 169–175.

[7] Bader, C. “Design of a Robust Dynamic Network Protocol for Robot Skin”. MA thesis. Technische Universität München, 2017.

[8] Bader, C., **Bergner, F.**, Cheng, G., “A Robust and Efficient Dynamic Network Protocol for a large-scale artificial robotic skin”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, 2018, pp. 1600–1605.

[9] Baglini, E., Cannata, G., Mastrogiovanni, F., “Design of an Embedded Networking Infrastructure for whole-Body Tactile Sensing in Humanoid Robots”. In: *10th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE. Nashville, United States, 2010, pp. 671–676.

[10] Baglini, E., Youssefi, S., Mastrogiovanni, F., Cannata, G., “A Real-Time Distributed Architecture for Large-Scale Tactile Sensing”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2014, pp. 1663–1669.

[11] Bartolozzi, C., Benosman, R., Boahen, K., Cauwenberghs, G., Delbrück, T., Indiveri, G., Liu, S.-C., Furber, S., Imam, N., Linares-Barranco, B., Serrano-Gotarredona, T., Meier, K., Posch, C., Valle, M., “Neuromorphic Systems”. In: *Wiley Encyclopedia of Electrical and Electronics Engineering* (2016), pp. 1–22.

[12] Bartolozzi, C., Ros, P. M., Diotalevi, F., Jamali, N., Natale, L., Crepaldi, M., Demarchi, D., “Event-driven encoding of off-the-shelf tactile sensors for compression and latency optimisation for robotic skin”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, Canada, 2017, pp. 166–173.

[13] Benarroch, E. E., Daube, J. R., Flemming, K. D., Westmoreland, B. F., “Mayo Clinic Medical Neurosciences”. In: ed. by Eduardo E. Benarroch, Jasper R. Daube, Kelly D. Flemming, and Barbara F. Westmoreland. Mayo Clinic Scientific Press and Informa Healthcare USA, Inc.: Rochester, United States, 2008. Chap. 7: The Sensory System, pp. 217–264.

[14] Benosman, R., Clercq, C., Lagorce, X., Ieng, S.-H., Bartolozzi, C., “Event-Based Visual Flow”. In: *IEEE Transactions on Neural Networks and Learning Systems* 25 (2014), pp. 407–417.

[15] **Bergner, F.**, Cheng, G., “Sensory Systems for Robotic Applications – Making sense of the world”. In: ed. by Ravinder S. Dahiya, Oliver Ozioko, and Gordon Cheng. The Institution of Engineering and Technology, 2020, Submitted. Chap. Neuromorphic Principles for Large-Scale Robot Skin.

[16] **Bergner, F.**, Dean-Leon, E., Cheng, G., “Event-based signaling for large-scale artificial robotic skin - Realization and performance evaluation”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, Korea, 2016, pp. 4918–4924.

[17] **Bergner, F.**, Dean-Leon, E., Cheng, G., “Efficient Event-Driven Reactive Control for Large Scale Robot Skin”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Singapore, Singapore, 2017, pp. 394–400.

[18] **Bergner, F.**, Dean-Leon, E., Cheng, G., “Efficient Distributed Torque Computation for Large Scale Robot Skin”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, 2018, pp. 1593–1599.

[19] **Bergner, F.**, Dean-Leon, E., Cheng, G., “Design and Realization of an Efficient Large-Area Event-Driven E-Skin”. In: *Sensors* 20.7 (2020), p. 1965.

[20] **Bergner, F.**, Dean-Leon, E., Guadarrama-Olvera, J. R., Cheng, G., “Evaluation of a Large Scale Event Driven Robot Skin”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4247–4254.

[21] **Bergner, F.**, Mittendorfer, P., Dean-Leon, E., Cheng, G., “Event-based signaling for reducing required data rates and processing power in a large-scale artificial robotic skin”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. Hamburg, Germany, 2015, pp. 2124–2129.

[22] Boahen, K. A. “Point-to-point connectivity between neuromorphic chips using address events”. In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 47.5 (2000), pp. 416–434.

[23] Bosch, R. *APAS Intelligent Systems for Man-Machine Collaboration*. Robert Bosch GmbH. Postfach 30 02 20, 70442 Stuttgart, Germany, 2016.

[24] Cannata, G., Denei, S., Mastrogiovanni, F., “Towards Automated Self-Calibration of Robot Skin”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2010, pp. 4849–4854.

[25] Cannata, G., Maggiali, M., Metta, G., Sandini, G., “An Embedded Artificial Skin for Humanoid Robots”. In: *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. IEEE. Seoul, Korea, 2008, pp. 434–438.

[26] Caviglia, S., Pinna, L., Bartolozzi, C., “An event-driven POSFET taxel for sustained and transient sensing”. In: *IEEE International Symposium on Circuits and Systems (ISCAS)*. Montreal, Canada, 2016, pp. 349–352.

[27] Caviglia, S., Pinna, L., Valle, M., Bartolozzi, C., “Spike-Based Readout of POSFET Tactile Sensors”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 64.6 (2016), pp. 1421–1431.

[28] Caviglia, S., Valle, M., Bartolozzi, C., “Asynchronous, event-driven readout of POSFET devices for tactile sensing”. In: *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. Melbourne, Australia, 2014, pp. 2648–2651.

[29] Cheng, G., Dean-Leon, E., **Bergner, F.**, Guadarrama-Olvera, J. R., Leboutet, Q., Mittendorfer, P., “A comprehensive realisation of Robot Skin: Sensors, Sensing, Control and Applications”. In: *Proceedings of the IEEE* 107.10 (2019), pp. 2034–2051.

[30] Cheung, E., Lumelsky, V. J., “Proximity sensing in robot manipulator motion planning: system and implementation issues”. In: *IEEE transactions on Robotics and Automation* 5.6 (1989), pp. 740–751.

[31] Conradt, J., Cook, M., Berner, R., Lichtsteiner, P., Douglas, R. J., Delbruck, T., “A pencil balancing robot using a pair of AER dynamic vision sensors”. In: *IEEE International Symposium on Circuits and Systems (ISCAS)*. Taipei, Taiwan, 2009.

[32] Cuevas, H. M., Velázquez, J., Dattel, A. R., *Human Factors in Practice - Concepts and Applications*. Ed. by Haydee M. Cuevas, Jonathan Velázquez, and Andrew R. Dattel. Taylor & Francis Group, LLC: Abingdon-on-Thames, United Kingdom, 2018.

[33] Dahiya, R. S., Cattin, D., Adami, A., Collini, C., Barboni, L., Valle, M., Lorenzelli, L., Oboe, R., Metta, G., Brunetti, F., “Towards tactile sensing system on chip for robotic applications”. In: *Sensors Journal* 11.12 (2011), pp. 3216–3226.

[34] Dahiya, R. S., Metta, G., Valle, M., Sandini, G., “Tactile sensing - from humans to humanoids”. In: *IEEE Transactions on Robotics* 26.1 (2010), pp. 1–20.

[35] Dahiya, R. S., Mittendorfer, P., Valle, M., Cheng, G., Lumelsky, V. J., “Directions Toward Effective Utilization of Tactile Skin: A Review”. In: *IEEE Sensors Journal* 13.11 (2013), pp. 4121–4138.

[36] Dahiya, R. S., Valle, M., *Robotic Tactile Sensing: Technologies and System*. Ed. by Ravinder S. Dahiya and Maurizio Valle. Springer: New York City, United States, 2013.

[37] Dahiya, R. “E-Skin: From Humanoids to Humans”. In: *Proceedings of the IEEE* 107.2 (2019), pp. 247–252.

[38] Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., Liao, Y., Lin, C.-K., Lines, A., Liu, R., Mathaikutty, D., McCoy, S., Paul, A., Tse, J., Venkataramanan, G., Weng, Y.-H., Wild, A., Yang, Y., Wang, H., "Loihi: A neuromorphic manycore processor with on-chip learning". In: *IEEE Micro* 38.1 (2018), pp. 82–99.

[39] Dayan, P., Abbott, L. F., *Theoretical neuroscience: computational and mathematical modeling of neural systems*. Ed. by Peter Dayan and Laurence F. Abbott. MIT press: Cambridge, United States, 2001.

[40] Dean-Leon, E., **Bergner, F.**, Ramirez-Amaro, K., Cheng, G., "From multi-modal tactile signals to a compliant control". In: *IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. 2016, pp. 892–898.

[41] Dean-Leon, E., Guadarrama-Olvera, J. R., **Bergner, F.**, Cheng, G., "Whole-Body Active Compliance Control for Humanoid Robots with Robot Skin". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Montreal, Canada, 2019, pp. 5404–1510.

[42] Dean-Leon, E., Pierce, B., Mittendorfer, P., **Bergner, F.**, Ramirez-Amaro, K., Burger, W., Cheng, G., "TOMM: Tactile Omnidirectional Mobile Manipulator". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Singapore, Singapore, 2017, pp. 2441–2447.

[43] Dean-Leon, E., Ramirez-Amaro, K., **Bergner, F.**, Cheng, G., "Robot Skin: Fully-Compliant Control Framework Using Multi-modal Tactile Events". In: *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI* 7 (2019), pp. 4–13.

[44] Dean-Leon, E., Ramirez-Amaro, K., **Bergner, F.**, Dianov, I., Cheng, G., "Integration of Robotic Technologies for Rapidly Deployable Robots". In: *IEEE Transactions on Industrial Informatics* 14.4 (2018), pp. 1691–1700.

[45] Dean-Leon, E., Ramirez-Amaro, K., **Bergner, F.**, Dianov, I., Lanillos, P., Cheng, G., "Robotic Technologies for Fast Deployment of Industrial Robot Systems". In: *The 42nd Annual Conference of IEEE Industrial Electronics Society (IECON)*. 2016, pp. 6900–6907.

[46] Deistler, M., Yener, Y., **Bergner, F.**, Lanillos, P., Cheng, G., "Tactile Hallucinations on Artificial Skin Induced by Homeostasis in a Deep Boltzmann Machine". In: *IEEE International Conference on Cyborg and Bionic Systems*. 2019.

[47] Del-Prete, A., Denei, S., Natale, L., Mastrogiovanni, F., Nori, F., Cannata, G., Metta, G., "Skin spatial calibration using force/torque measurements". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2011.

[48] Dianov, I., Ramirez-Amaro, K., Lanillos, P., Dean-Leon, E., **Bergner, F.**, Cheng, G., "Extracting general task structures to accelerate the learning of new tasks". In: *IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. 2016, pp. 802–807.

[49] Fishel, J. A., Santos, V. J., Loeb, G. E., “A robust micro-vibration sensor for biomimetic fingertips”. In: *2nd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*. Scottsdale, United States, 2008.

[50] Fritzsche, M., Elkmann, N., Schulenburg, E., “Tactile sensing: A key technology for safe physical human robot interaction”. In: *Proceedings of the 6th International Conference on Human-robot Interaction*. 2011, pp. 139–140.

[51] Furber, S. B., Lester, D. R., Plana, L. A., Garside, J. D., Painkras, E., Temple, S., Brown, A. D., “Overview of the spinnaker system architecture”. In: *IEEE Transactions on Computers* 62.12 (2013), pp. 2454–2467.

[52] Fuster, J. M. “Upper processing stages of the perception–action cycle”. In: *Trends in cognitive sciences* 8.4 (2004), pp. 143–145.

[53] Gardner, E. P. “Touch”. In: *Encyclopedia of Life Sciences*. John Wiley & Sons: New York City, United States, 2010.

[54] Gardner, E. P., Johnson, K. O., “Principles of Neural Science, Fifth Edition”. In: ed. by Eric R. Kandel, James H. Schwartz, Thomas M. Jassell, Steven A. Siegelbaum, and A. J. Hudspeth. The MacGraw-Hill Companies: New York City, United States, 2013. Chap. 22: The Somatosensory System: Receptors and Central Pathways, pp. 475–497.

[55] Gardner, E. P., Johnson, K. O., “Principles of Neural Science, Fifth Edition”. In: ed. by Eric R. Kandel, James H. Schwartz, Thomas M. Jassell, Steven A. Siegelbaum, and A. J. Hudspeth. The MacGraw-Hill Companies: New York City, United States, 2013. Chap. 23: Touch, pp. 498–529.

[56] Gardner, E. P., Johnson, K. O., “Principles of Neural Science, Fifth Edition”. In: ed. by Eric R. Kandel, James H. Schwartz, Thomas M. Jassell, Steven A. Siegelbaum, and A. J. Hudspeth. The MacGraw-Hill Companies, 2013. Chap. 21: Sensory Coding, pp. 449–474.

[57] Gehrig, S. K., Rabe, C., “Real-Time Semi-Global Matching on the CPU”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition–Workshops*. San Francisco, United States, 2010, pp. 85–92.

[58] Goldstein, E. B., Rockmole, J. R., “Sensation and Perception”. In: ed. by E. Bruce Goldstein and James R. Rockmole. Cengage Learning: Boston, United States, 2015. Chap. 14: The Cutaneous Senses, pp. 337–380.

[59] Grunwald, G., Schreiber, G., Albu-Schäffer, A., Hirzinger, G., “Programming by Touch: The Different Way of Human-Robot Interaction”. In: *IEEE Transactions on Industrial Electronics* 50.4 (2003), pp. 659–666.

[60] Guadarrama Olvera, J. R., Dean Leon, E., **Bergner, F.**, Cheng, G., “Plantar Tactile Feedback For Biped Balance and Locomotion on Unknown Terrain”. In: *International Journal of Humanoid Robotics* 17.1 (2019), p. 1950036.

[61] Guadarrama-Olvera, J. R., **Bergner, F.**, Dean-Leon, E., Cheng, G., “Enhancing Biped Locomotion on Unknown Terrain Using Tactile Feedback”. In: *International Conference on Humanoid Robots (Humanoids)*. Beijing, China, 2018, pp. 47–52.

[62] Guadarrama-Olvera, J. R., Dean-Leon, E., **Bergner, F.**, Cheng, G., “Pressure-Driven Body Compliance Using Robot Skin”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4418–4423.

[63] Hajimiri, A. “The future of high frequency circuit design”. In: *Proceedings of ESSCIRC*. 2009, pp. 44–51.

[64] Harmon, L. D. “Automated Tactile Sensing”. In: *The International Journal of Robotics Research* 1.2 (1982), pp. 3–32.

[65] Hirzinger, G., Sporer, N., Albu-Schaffer, A., Hahnle, M., Krenn, R., Pascucci, A., Schedl, M., “DLR’s torque-controlled light weight robot III-are we reaching the technological limits now?” In: *IEEE International Conference on Robotics and Automation*. Washington, United States, 2002, pp. 1710–1716.

[66] Hoxha, I., Del Duca, F., Ehrlich, S. K., **Bergner, F.**, Berberich, N., Cheng, G., “Improving user comfort in auditory-steady-state-response brain-computer interface by using a co-adaptive stimulus”. In: *Federation of European Neuroscience Societies (FENS)*. 2020.

[67] Hsu, J. “Ibm’s new brain”. In: *IEEE Spectrum* 51.10 (2014), pp. 17–19.

[68] Indiveri, G., Linares-Barranco, B., Hamilton, T. J., Schaik, A. V., Etienne-Cummings, R., Delbruck, T., Liu, S.-C., Dudek, P., Häfliger, P., Renaud, S., Schemmel, J., Cauwenberghs, G., Arthur, J., Hynna, K., Folowosele, F., Saighi, S., Serrano-Gotarredona, T., Wijekoon, J., Wang, Y., Boahen, K., “Neuromorphic silicon neuron circuits”. In: *Frontiers in Neuroscience* 5 (2011), pp. 1–22.

[69] Itti, L. “Real-time high-performance attention focusing in outdoors color video streams”. In: *Human Vision and Electronic Imaging VII*. Vol. 4662. 2002, pp. 235–243.

[70] Iwata, H., Sugano, S., “Design of human symbiotic robot TWENDY-ONE”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2009, pp. 580–586.

[71] Johnson, H., Graham, M., *High-speed digital design: a handbook of black magic*. Ed. by Howard Johnson and Martin Graham. Prentice Hall Englewood Cliffs, NJ, 1993.

[72] Kandel, E. R., Barres, B. A., Hudspeth, A. J., “Principles of Neural Science, Fifth Edition”. In: ed. by Eric R. Kandel, James H. Schwartz, Thomas M. Jassell, Steven A. Siegelbaum, and A. J. Hudspeth. The MacGraw-Hill Companies: New York City, United States, 2013. Chap. 2: Nerve Cell, Neural Circuitry and Behavior, pp. 21–38.

[73] Kandel, E. R., Barres, B. A., Hudspeth, A. J., “Principles of Neural Science, Fifth Edition”. In: ed. by Eric R. Kandel, James H. Schwartz, Thomas M. Jassell, Steven A. Siegelbaum, and A. J. Hudspeth. The MacGraw-Hill Companies: New York City, United States, 2013. Chap. 26: Low-Level Visual Processing: The Retina, pp. 577–601.

[74] Kandel, E. R., Schwartz, J. H., Jassell, T. M., Siegelbaum, S. A., Hudspeth, A. J., *Principles of Neural Science, Fifth Edition*. Ed. by Eric R. Kandel, James H. Schwartz, Thomas M. Jassell, Steven A. Siegelbaum, and A. J. Hudspeth. The MacGraw-Hill Companies, 2013.

[75] Kerrisk, M. “The Linux programming interface: a Linux and UNIX system programming handbook”. In: ed. by Michael Kerrisk. No Starch Press: San Francisco, United States, 2010. Chap. 63: Alternative I/O Models, pp. 1325–1374.

[76] Knuth, D. E. “The Art of Computer Programming: Sorting and Searching”. In: ed. by Donald E. Knuth. Pearson Education, Inc.: London, United Kingdom, 1998. Chap. 6: Searching, pp. 392–583.

[77] Kobayashi, T., Dean-Leon, E., Guadarrama-Olvera, J. R., **Bergner, F.**, Cheng, G., “Multi-Contacts Force-Reactive Walking Control during Physical Human-Humanoid Interaction”. In: *IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*. 2019, pp. 33–39.

[78] Kobayashi, T., Dean-Leon, E., Guadarrama-Olvera, J. R., **Bergner, F.**, Cheng, G., “Whole-Body Multicontact Haptic Human–Humanoid Interaction Based on Leader–Follower Switching: A Robot Dance of the “Box Step””. In: *Advanced Intelligent Systems* (2021), p. 2100038.

[79] Kõiva, R., Zenker, M., Schürmann, C., Haschke, R., Ritter, H. J., “A highly sensitive 3D-shaped tactile sensor”. In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2013, pp. 1084–1089.

[80] Leblebici, D., Leblebici, Y., *Fundamentals of high-frequency CMOS analog integrated circuits*. Ed. by Duran Leblebici and Yusuf Leblebici. Cambridge University Press, 2009.

[81] Leboutet, Q., **Bergner, F.**, Cheng, G., “Online Adaptive Configuration Selection for Redundant Arrays of Inertial Sensors: Application to Robotic Systems Covered with a Multimodal Artificial Skin”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[82] Leboutet, Q., Dean-Leon, E., **Bergner, F.**, Cheng, G., “Tactile-Based Whole-Body Compliance With Force Propagation for Mobile Manipulators”. In: *IEEE Transactions on Robotics* 35.2 (2019), pp. 330–342.

[83] Leboutet, Q., Guadarrama-Olvera, J. R., **Bergner, F.**, Cheng, G., “Second-order Kinematics for Floating-base Robots using the Redundant Acceleration Feedback of an Artificial Sensory Skin”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020.

[84] Lee, W. W., Kukreja, S. L., Thakor, N. V., “A kilohertz kilotaxel tactile sensor array for investigating spatiotemporal features in neuromorphic touch”. In: *Biomedical Circuits and Systems Conference (BioCAS)*. 2015, pp. 1–4.

[85] Lee, W. W., Tan, Y. J., Yao, H., Li, S., See, H. H., Hon, M., Ng, K. A., Xiong, B., Ho, J. S., Tee, B. C. K., "A neuro-inspired artificial peripheral nervous system for scalable electronic skins". In: *Science Robotics* 4.32 (2019).

[86] Lichtsteiner, P., Posch, C., Delbruck, T., "A 128 $\times$ 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor". In: *IEEE Journal of Solid-State Circuits* (2008). DOI: 10.1109/JSSC.2007.914337.

[87] Liu, S.-C., Delbruck, T., "Neuromorphic sensory systems". In: *Current opinion in neurobiology* 20.3 (2010), pp. 288–295.

[88] Liu, S.-C., Delbruck, T., Indiveri, G., Whatley, A., Douglas, R., *Event-based neuromorphic systems*. Ed. by Shih-Chii Liu, Tobi Delbruck, Giacomo Indiveri, Adrian Whatley, and Rodney Douglas. Washington, United States: John Wiley & Sons: New York City, United States, 2015.

[89] Lumelsky, V. J., Shur, M. S., Wagner, S., "Sensitive Skin". In: *IEEE Sensors Journal* 1.1 (2001), pp. 41–51.

[90] Mahowald, M. "VLSI analogs of neuronal visual processing: a synthesis of form and function". PhD thesis. California Institute of Technology, Pasadena, CA, United States, 1992.

[91] Maiolino, P., Maggiali, M., Cannata, G., Metta, G., Natale, L., "A flexible and robust large scale capacitive tactile system for robots". In: *IEEE Sensors Journal* 13.10 (2013), pp. 3910–3917.

[92] Massa, D., Callegari, M., Cristalli, C., "Manual guidance for industrial robot programming". In: *Industrial Robot* 42.5 (2015), pp. 457–465.

[93] Matsui, T., Hirukawa, H., Ishikawa, Y., Yamasaki, N., Kagami, S., Kanehiro, F., Saito, H., Inamura, T., "Distributed real-time processing for humanoid robots". In: *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*. Hong Kong, China, 2005, pp. 205–210.

[94] Maurer, W. D., Lewis, T. G., "Hash table methods". In: *ACM Computing Surveys (CSUR)* 7.1 (1975), pp. 5–19.

[95] Meier, K. "A mixed-signal universal neuromorphic computing system". In: *IEEE International Electron Devices Meeting (IEDM)*. 2015, pp. 4–6.

[96] Miskowicz, M. "Analytical approximation of the uniform magnitude-driven sampling effectiveness". In: *IEEE International Symposium on Industrial Electronics*. Istanbul, Turkey, 2004, pp. 407–410.

[97] Miskowicz, M. "Send-On-Delta Concept: An Event-Based Data Reporting Strategy". In: *Sensors* 6.1 (2006), pp. 49–63.

[98] Miskowicz, M. "Event-based sampling strategies in networked control systems". In: *10th IEEE Workshop on Factory Communication Systems (WFCS)*. 2014, pp. 1–10.

[99] Miskowicz, M. *Event-Based Control and Signal Processing*. Ed. by Marek Miskowicz. CRC Press LLC: Boca Raton, United States, 2016.

[100] Mittendorfer, P. “From a Multi-modal Intelligent Cell to a Self-organizing Robotic Skin – Realizing Self- and Enriching Robot Interaction”. PhD thesis. TUM, 2014.

[101] Mittendorfer, P., Cheng, G., “Humanoid multimodal tactile-sensing modules”. In: *IEEE Transactions on Robotics* 27.3 (2011), pp. 401–410.

[102] Mittendorfer, P., Cheng, G., “3D surface reconstruction for robotic body parts with artificial skins”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. Vilamoura, Portugal, 2012, pp. 4505–4510.

[103] Mittendorfer, P., Cheng, G., “Integrating discrete force cells into multi-modal artificial skin”. In: *12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE. 2012, pp. 847–852.

[104] Mittendorfer, P., Cheng, G., “Open-loop self-calibration of articulated robots with artificial skins”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2012, pp. 4539–4545.

[105] Mittendorfer, P., Cheng, G., “Uniform cellular design of artificial robotic skin”. In: *7th German Conference on Robotics; Proceedings of ROBOTIK 2012*. VDE. 2012, pp. 145–149.

[106] Mittendorfer, P., Cheng, G., “From a Multi-Modal Intelligent Cell to a Self-Organizing Robotic-Skin”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany*. 2013.

[107] Mittendorfer, P., Dean, E., Cheng, G., “3D spatial self-organization of a modular artificial skin”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*. IEEE. 2014, pp. 3969–3974.

[108] Mittendorfer, P., Dean, E., Cheng, G., “Automatic robot kinematic modeling with a modular artificial skin”. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE. 2014, pp. 749–754.

[109] Mittendorfer, P., Yoshida, E., Cheng, G., “Realizing whole-body tactile interactions with a self-organizing, multi-modal artificial skin on a humanoid robot”. In: *Advanced Robotics* 29.1 (2015), pp. 51–67.

[110] Moradi, S., Qiao, N., Stefanini, F., Indiveri, G., “A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)”. In: *IEEE transactions on biomedical circuits and systems* 12.1 (2017), pp. 106–122.

[111] Mosteller, R. D. “Simplified calculation of body-surface area”. In: *The New England Journal of Medicine* 317 (1987), p. 1098.

[112] Mukai, T., Onishi, M., Odashima, T., Hirano, S., Luo, Z., “Development of the tactile sensor system of a human-interactive robot “RI-MAN””. In: *IEEE Transactions on Robotics* 24.2 (2008), pp. 505–512.

[113] Neckar, A., Fok, S., Benjamin, B. V., Stewart, T. C., Oza, N. N., Voelker, A. R., Eliasmith, C., Manohar, R., Boahen, K., “Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model”. In: *Proceedings of the IEEE* 107.1 (2019), pp. 144–164.

[114] Neugebauer, M., Kabitzsch, K., “A New Protocol for a Low Power Sensor Network”. In: *IEEE International Conference on Performance, Computing, and Communications*. Phoenix, United States, 2004, pp. 393–399.

[115] Neumann, J. “First Draft of a Report on the EDVAC”. In: *Moore School of Electrical Engineering* (1945).

[116] Nori, F., Traversaro, S., Eljaik, J., Romano, F., Prete, A. D., Pucci, D., “iCub whole-body control through force regulation on rigid non-coplanar contacts”. In: *Frontiers in Robotics and AI* 2.6 (2015), pp. 1–18.

[117] Ohmura, Y., Kuniyoshi, Y., Nagakubo, A., “Conformable and scalable tactile sensor skin for curved surfaces”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Orlando, United States, 2006, pp. 1348–1353.

[118] Orchard, G., Meyer, C., Etienne-Cummings, R., Posch, C., Thakor, N., Benosman, R., “HFirst: a temporal approach to object recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 37.10 (2015), pp. 2028–2040.

[119] Patestas, M. A., Gartner, L. P., “A Textbook of Neuroanatomy”. In: ed. by Maria A. Patestas and Leslie P. Gartner. Blackwell Publishing: Hoboken, United States, 2006. Chap. 10: Ascending Sensory Pathways, pp. 137–170.

[120] Posch, C., Matolin, D., Wohlgenannt, R., “An Asynchronous Time-based Image Sensor”. In: *ISCAS, IEEE International Symposium on Circuits and Systems*. Seattle, United States, 2008.

[121] Posch, C., Matolin, D., Wohlgenannt, R., “A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS”. In: *IEEE Journal of Solid-State Circuit* 46.1 (2011), pp. 259–275.

[122] Posch, C., Serrano-Gotarredona, T., Linares-Barranco, B., Delbruck, T., “Retinomorphic Event-Based Vision Sensors: Bioinspired Cameras With Spiking Output”. In: *Proceedings of the IEEE* 102.10 (2014), pp. 1470–1484.

[123] Qiao, N., Mostafa, H., Corradi, F., Osswald, M., Stefanini, F., Sumislawska, D., Indiveri, G., “A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses”. In: *Frontiers in neuroscience* 9 (2015), pp. 1–17.

[124] Ramirez-Amaro, K., Dean-Leon, E., **Bergner, F.**, Cheng, G., “A Semantic-Based Method for Teaching Industrial Robots New Tasks”. In: *KI-Künstliche Intelligenz* 33.2 (2019), pp. 117–122.

[125] Ramirez-Amaro, K., Dean-Leon, E., Dianov, I., **Bergner, F.**, Cheng, G., “General recognition models capable of integrating multiple sensors for different domains”. In: *IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. 2016, pp. 306–311.

[126] Ros, P. M., Crepaldi, M., Bartolozzi, C., Demarchi, D., “Asynchronous DC-free serial protocol for event-based AER systems”. In: *IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. Cairo, Egypt, 2015, pp. 248–251.

[127] Roudaut, Y., Lonigro, A., Coste, B., Hao, J., Delmas, P., Crest, M., “Touch sense: Functional organization and molecular determinants of mechanosensitive receptors”. In: *Channels* 6 (2012), pp. 234–245.

[128] Saal, H. P., Bensmaia, S. J., “Touch is a team effort: interplay of submodalities in cutaneous sensibility”. In: *Trends in neurosciences* 37.12 (2014), pp. 689–697.

[129] Saladin, K. S., Gan, C. A., Cushman, H. N., “Anatomy & Physiology: The Unity of Form and Function”. In: ed. by Kenneth S. Saladin. McGraw-Hill Education: New York City, United States, 2018. Chap. 16: Sense Organs, pp. 575–625.

[130] Schaik, A., Liu, S.-C., “AER EAR: A matched silicon cochlea pair with address event representation interface”. In: *IEEE International Symposium on Circuits and Systems (ISCAS)*. Kobe, Japan, 2005, pp. 4213–4216.

[131] Schmitz, A., Maiolino, P., Maggiali, M., Natale, L., Cannata, G., Metta, G., “Methods and technologies for the implementation of large-scale robot tactile sensors”. In: *IEEE Transactions on Robotics* 27.3 (2011), pp. 389–400.

[132] Shannon, C. E. “A mathematical theory of communication”. In: *Bell system technical journal* 27.3 (1948), pp. 379–423.

[133] Siegelbaum, S. A., Koester, J., “Principles of Neural Science, Fifth Edition”. In: ed. by Eric R. Kandel, James H. Schwartz, Thomas M. Jassell, Steven A. Siegelbaum, and A. J. Hudspeth. The MacGraw-Hill Companies, 2013. Chap. 5: Ion Channels, pp. 100–125.

[134] Someya, T., Kato, Y., Sekitani, T., Iba, S., Noguchi, Y., Murase, Y., Kawaguchi, H., Sakurai, T., “Conformable, flexible, large-area networks of pressure and thermal sensors with organic transistor active matrixes”. In: *Proceedings of the National Academy of Sciences* 102 (2005), pp. 12321–12325.

[135] Strawn, G. “Howard Aiken: Mastermind of the Harvard Mark Computers”. In: *IT Professional* 21.6 (2019), pp. 66–68.

[136] Strohmayr, M. “Artificial Skin in Robotics”. PhD thesis. Fakultät für Informatik des Karlsruher Instituts für Technologie (KIT), 2012.

[137] Suh, Y. S. “Send-on-delta sensor data transmission with a linear predictor”. In: *Sensors* 7.4 (2007), pp. 537–547.

[138] Tar, A., Cserey, G., “Development of a low cost 3d optical compliant tactile force sensor”. In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2011, pp. 236–240.

[139] Thorpe, S., Delorme, A., Van Rullen, R., “Spike-based strategies for rapid processing”. In: *Neural networks* 14.7 (2001), pp. 715–725.

[140] Timmermann, D., Hahn, H., Hosticka, B. J., “Low Latency Time CORDIC Algorithms”. In: *IEEE Transactions on Computers* 41.8 (1992), pp. 1010–1015.

[141] Wakatabe, R., Kuniyoshi, Y., Cheng, G., “O (logn) algorithm for forward kinematics under asynchronous sensory input”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 2502–2507.

[142] Wösch, T., Feiten, W., “Reactive Motion Control for Human-Robot Tactile Interaction”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2002.

[143] Yamaguchi, T., Numasato, H., Hirai, H., “A mode-switching control for motion control and its application to disk drives: Design of optimal mode-switching conditions”. In: *IEEE/ASME transactions on mechatronics* 3.3 (1998), pp. 202–209.

[144] Youssefi, S., Denei, S., Mastrogiovanni, F., Cannata, G., “A real-time data acquisition and processing framework for large-scale robot skin”. In: *Robotics and Autonomous Systems* 68 (2015), pp. 86–103.