Event-based signaling for large-scale artificial robotic skin - Realization and performance evaluation

Florian Bergner, Emmanuel Dean-Leon and Gordon Cheng

***Abstract*— In this paper we describe how we realized event-based signaling for large scale artificial robotic skin. We developed a new algorithm for the event generation on multi-modal skin cells. The skin cells have two modes, the conventional data sampling mode and the event mode. A comprehensive performance evaluation and comparison of these two modes is presented. We perform different experiments on our robot TOMM which has two UR5 robot arms, each covered with 260 multi-modal skin cells. Each skin cell samples 9 signals of 4 different modalities. Finally we derive models for extrapolating CPU usage and network traffic for larger numbers of skin cells and higher sample rates. The results show that the event-based system has superior performance and its performance edge increases with larger numbers of skin cells and higher sample rates. Experimental validation on our real robot system shows that in reactive control the event-based system reduces in comparison to the conventional system the packet rate by 48.2% and the CPU usage by 17.79%. We extrapolate the worst case for 5000 cells and show that the event-based system can at least reduce the packet rate by 21.2% and the CPU usage by 17.46%.**

I. INTRODUCTION

A. Motivation

Giving robots skin and thus giving them the sense of touch has already been addressed in the 1980's [1]. The reasons for equipping robots with skin are diverse and are recently becoming more important as robots are evolving to a state where they need to interact and collaborate with humans. For intuitive and easy interactions interactive robots need to become aware of people and tactile guidance and thus perceive and process tactile stimuli. Skin for robots will not only enhance human-robot interaction to a completely new level, skin will make robots also safer for human-robot interaction.

Recent work on robotic skin [2]–[6], [17], [18] demonstrate different approaches to tackle general hardware and software challenges on realizing skin which partially covers robots. Eventually, to fully exploit the capabilities of robotic skin, skin has to cover a robot completely. We estimate the lower bound of skin cells for full robot coverage to somewhere between 3000 and 5000 skin cells. Scaling up robotic skin to cover a whole robot induces many new complex challenges, e.g. handling, conveying and processing vast amounts of tactile information. These challenges become more prominent with increasing amounts of skin cells, up to the point where solutions become infeasible or non-existent. One way to cope with these challenges is to organize skin in a modular, distributed form with separated skin regions connected with multiple interfaces [2], [4]. This results in low latency and high bandwidth communication paths by using parallelism. However it also increases the number of cables and interfaces which induces higher costs and deployment problems in space constrained mobile robots. The work of [7], [8] proposes an organized architecture to handle tactile information more efficiently in higher processing layers. Organizing data helps to reduce latency by reducing searching time. However these contributions do not reduce the amount of data which has to be communicated and processed. But a low latency, high speed flow of information is essential for stable and fast controllers and thus for fast robot reactions to tactile stimuli. A very promising bio-inspired approach for solving these challenges is using event-based signaling and processing [10]–[16], [20]–[23]. The event-based principle can reduce the amount of data which has to be conveyed and processed while not reducing or loosing information. In the standard approach the synchronous system acquires and processes information at a predefined constant sampling rate, in the asynchronous event-based approach however the whole system becomes only active on the appearance of new information. More precisely, the ideal event based sensor becomes only active and forwards information whenever the transduced and monitored information change exceeds a specified threshold. Obviously, such an event-based system has the potential to generate less data which has to be conveyed and processed. This results in lower demands on communication bandwidth and processing power for the same amount of skin cells.

In our previous paper [9] we demonstrated through simulations with off-line event generators and real sensor data that event-based signaling indeed reduces the requirements on the communication bandwidth. This work now focuses on the realization of the previously proposed event-based signaling

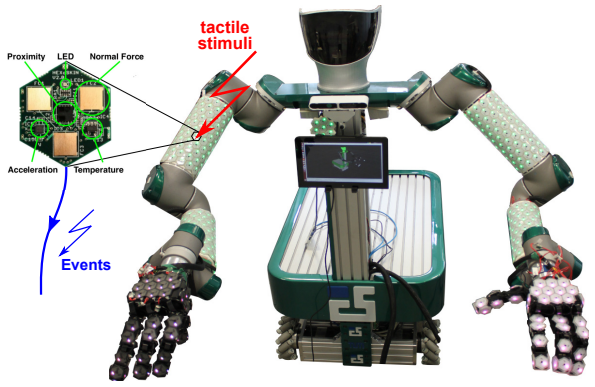


Fig. 1. Robot TOMM with robotic skin using event-based signaling.

All authors are with the Institute for Cognitive Systems, Technische Universität München, Munich, Germany, see http://www.ics.ei.tum.de

principle in our robot skin architecture. The full realization of the new concept allows us to compare it to the conventional system in real time and to fully and comprehensively evaluate its performance. The performance evaluation allows us to extrapolate demands on communication bandwidth and CPU usage for larger robotic skin in the near future. We expect event-based systems to reveal a much better performance than conventional systems for large numbers skin cells and high sample rates.

B. Related Work

In the field of sensing many works already take advantage of event-based approaches. So far the works on event-based vision sensors of [10]–[15] are most advanced and the proposed sensors and principles are already used in various applications. The proposed vision sensor monitors the temporal contrast at pixel level and generates a pixel event whenever its contrast exceeds a defined threshold. The events are time multiplexed within a matrix structure and forwarded using the address event representation (AER) protocol. In the AER protocol the presence of an event itself encodes its occurrence time and the address bus encodes the occurrence location of the event. These vision sensors provide low data rates and high temporal resolution.
The work of [16] investigates the feasibility of developing a neuromorphic event-based force sensor which uses the same principles as the event-based vision sensors. The proposed force sensor transduces forces to voltages and voltages changes above a specified threshold generate force events. The authors propose to arrange these event-based force sensors into arrays and propose the AER protocol to convey force events from the periphery to higher processing layers. The surveys of [17], [18] refer to this neuromorphic event-based force sensor and deduce that neuromorphic force sensors will be advantageous for data reduction in large scale robotic skin. The authors also propose on site embedded signal processing using SVD for data reduction. However they do not provide solutions for the wiring problem when using the parallel AER protocol or solutions for sufficient on site embedded computing power. A solution to the wiring problem of the AER protocol is presented in [19]. The authors introduce a low latency high speed AER serializer/deserializer which is able to transport an AER event by using only 3 wires. This serial interface has potential but requires reliable high speed connections between skin cells. The works of [20]–[23] introduce two principles for event-based systems within the context of signaling in low power distributed sensor networks, namely the *send-on-delta* principle and the principle of *compound architectures*. The send-on-delta principle is quite similar to the event generation proposed in [10]–[15], [19]. But instead of immediately propagating the event using the multiplexed AER protocol, the event triggers the transmission of a packet containing the event. The send-on-delta principle reduces transmission rates and still works with standard communication interfaces. However the principle cannot match the high temporal precision of the AER protocol. The send-on-delta principle allows compound architectures where the signal transduction and acquisition is still synchronous but where the information flow from the periphery to higher layers is event based. These two principles offer the chance to employ event-based principles in existing robotic skin systems with conventional sensors. Neuromorphic sensors which transduce stimuli directly to events would reduce the processing load of the skin cell microcontrollers and the energy consumption of the whole system. However such multi-modal event-based touch sensors do not exist yet.
The works of [4], [7], [8] focus on the overall skin software architecture and on how to convey tactile information from the periphery to higher processing levels in a generalized and organized way. However the proposed system is based on the send-on-request principle which enforces a synchronous information acquisition process. This process increases latencies as new information has to be requested which doubles the impact of the network delay.

C. Contributions

To our best knowledge this work introduces the first realization of event-based signaling in larger scale robotic skin. We integrated event generators to our skin cell firmware and developed a skin driver library and ROS interface. The whole skin system supports now two modes, the conventional data sampling mode and the event mode. This allows us to switch between modes while executing experiments. Furthermore we investigate the behavior of the new system, compare it to the conventional one and provide a thorough and comprehensive performance evaluation. We also propose empirical models for the relationship between CPU usage, packet rate, sample rate and the executed experiment such that we can extrapolate CPU usage and packet rate for large scale robot skin and higher sample rates.

II. SYSTEM DESCRIPTION

A. CellulARSkin

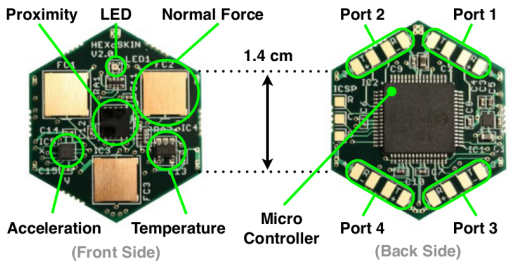


Fig. 2. CellulARSkin.

For our experimentation and evaluation we use our robot skin *CellulARSkin* [2], a modularized, multi-modal skin. The skin consists of hexagonally shaped skin cells (see Fig. 2). The sensors of such a skin cell transduce different tactile modalities like vibration (3D acceleration sensor), pressure (3 capacitive force sensors), pre-touch (proximity sensor) and temperature (2 temperature sensors). The microcontroller on the back of the skin cell collects the sensor data, filters the acquired data and packs it into a data packet. The data

packet which contains 9 sensor values is sent through the skin cell network to higher processing layers. The skin cells communicate with up to four neighbors and embody the skin cell communication network (see Fig. 3). The network is managed by the distributed microcontrollers of the skin cells and is initialized during the start-up process. The skin cells can change their sample rate on demand.

In this work we extended the skin cell firmware such that it supports the conventional data sampling mode and the event mode. Since the skin cell network only supports one fixed packet size the event packet size has to be the same as the data packet size. This constraint prevents us from using the optimal packet size for event packets. The interface boxes at the root of a skin cell network convert skin cell network packets to standard UDP Ethernet packets (see Fig. 3).

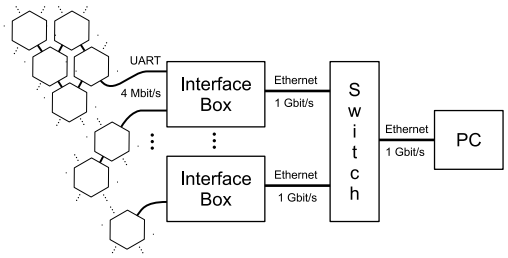


Fig. 3. The skin cell network architecture and interface to the PC.

B. Event generation

1) The general principle: We use the *send-on-delta* principle, introduced by [20], [21] within a *compound architecture* to generate the events. In our previous work [9] we explained how we can use of this principle in the context of robot skins. We demonstrated by off-line simulation with real sensor data how the proposed event generator will behave and how it will reduce network bandwidth demands. The proposed event generator samples, locally on the microcontroller, sensor values as fast as it can. The event generator monitors the change of the sampled signal and generates an event whenever the accumulated change exceeds the given threshold.

$$\left|\frac{\mathrm{d}}{\mathrm{d}t}f(t)\right| \mathrm{d}t \geq \mathit{thresh} \rightarrow \textbf{event} \tag{1}$$

For discrete sampled signals with a sampling period of $T_\mathrm{s} = t_i - t_{i-1}$ this equation results in:

$$\frac{\left| \int_{t_{i-1}}^{t_i} \frac{\mathrm{d}}{\mathrm{d}t} f(t)\ \mathrm{d}t \right|}{t_i - t_{i-1}} \geq \mathit{thresh} \rightarrow \textbf{event} \tag{2}$$

$$\Delta_\mathrm{f} = |f(t_i) - f(t_{i-1})| \geq \mathit{thresh} \rightarrow \textbf{event} \tag{3}$$

In our implementation an event is represented by its sensor type (proximity, force, acceleration and temperature) and the sensor value at its occurrence. In this way we avoid drifts.

2) The realization on skin cells: We integrate the event generator into the existing microcontroller firmware which allows us to switch between the conventional data sampling mode and the event mode. This eases the experimentation where we need to compare the performance of both modes. As a drawback we cannot change the event packet size to its optimum [9]. Algorithm 1 describes how the skin cell program generates events from updated sensor values. First the skin cell acquires and filters sensor data and thus gets the most recent set of sensor values. If the cell is in the data mode then it directly creates a normal data packet and queues it to the output buffer. However if the skin cell is in event mode then the skin cell program calculates the absolute difference between the current sensor values and the previous ones. Whenever the difference of one sensor value is above the given threshold then the current sensor value is saved and an active event flag is set. Next the skin cell creates a packet containing the current values of all active events. Fig. 4 shows how to encode and store events into an event packet. Note that we make use of the whole space of the fixed 20 byte size of a packet and store up to 6 events into one packet. From the simulations in [9] we know that a smaller packet which encodes only 3 events would have been optimal.

Algorithm 1 Event generation

```
prevData ← 0
loop
  update currData
  if mode is DATA_MODE then
    create data packet
    send data packet
    continue
  end if
  diff ← |currData − prevData|
  mask ← 0
  for i = 0 to 8 do
    if diff[i] > thresh[i] then
      set bit i in mask
      prevData[i] ← currData[i]
    end if
  end for
  if any set bit of enabledMask is set in mask then
    create event packet
    send event packet
  end if
end loop
```

byte 0	Event data packet header
byte 1 - 2	Cell Id (14 bits)
byte 3 - 4	Event type mask (9 bits + 1 bit) 9 bits → 9 event types 1 bit → either packet one or two
byte 5 - 18	6 × 16 bit values for 6 events
byte 19	End of packet

Fig. 4. Event data packet definition.

C. System architecture

1) Skin Driver Library: Besides supporting the data mode and event mode in the skin cell firmware, the new host software which acquires and processes skin information also needs to support both modes. For this reason we completely revised the host software and created a new system architecture which generalizes and modularizes the skin driver library for host applications (see Fig. 5). The

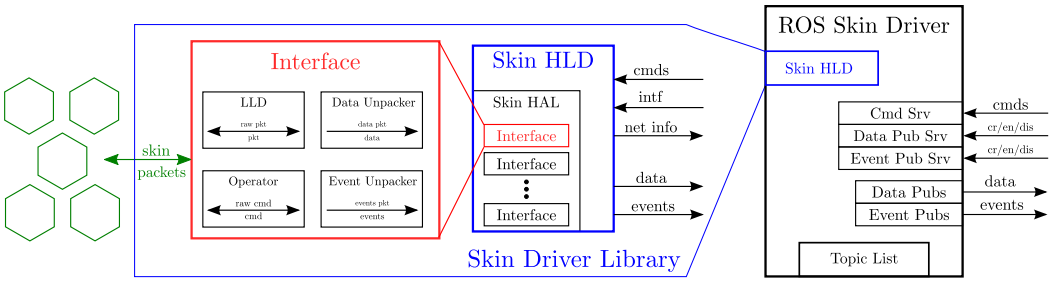


Fig. 5. Skin Driver and ROS skin driver.

skin hardware abstraction layer *Skin HAL* generalizes the low level part of the skin driver. The *Skin HAL* provides a set of *Interfaces* which manages connections to skin cell networks over different hardware interfaces like Ethernet/UDP or USB (see Fig. 5). The low level driver *LLD* of the *Interface* converts skin packets to more general skin driver packets and data/event unpackers convert data/event packets to skin cell data or skin cell events. The interface also handles the initialization of the skin network and the different operation modes. The high level driver *Skin HLD* (see Fig. 5) provides a generalized high level interface and allows the selection of a desired hardware connection. The new skin driver library allows applications directly to select either data or event mode which simplifies our experiments.

2) Event unpacking: Algorithm 2 demonstrates how we unpack event packets to events in the event unpacker module of a skin driver *Interface*. The unpacking algorithm uses the

Algorithm 2 Event unpacking

```
loop
  clear eventList
  fill eventPktQueue
  for each eventPkt in eventPktQueue do
    get eventMask
    for each bit set in eventMask do
      get eventType
      get eventValue
      create event
      append event to eventList
    end for
  end for
  publish eventList
end loop
```

event mask of the event packet to find out which events are active and which event values are encoded in the event value section of the event packet. Each bit in the event mask represents one specific event like a force sensor 1 event or an x-axis acceleration event. The order of the bits also represents the order of the event values in the event value section of the packet. After an event is unpacked it is added to the event list. The event list is finally forwarded to the higher abstraction levels of the skin driver.

3) Event to Skin Cell Data conversion: The skin driver library also provides a *Event to Data Converter* module which converts skin cell events back to normal skin cell data containing all the current sensor values of a skin cell. This module has a local memory of all skin cell sensor values which are updated according to the incoming events. Every update of a skin cell data block also triggers the forwarding of this block to higher processing layers. The converter module is helpful in applications which not yet support events but which want to take advantage of event-based signaling.

4) ROS skin driver: The experiments include our robot TOMM which uses ROS (see Fig. 5). The ROS skin driver provides a ROS interface to the skin and makes extensive use of the new skin library. The ROS interface supports skin cell data and skin cell events. In this way ROS nodes can take full advantage of event-based signaling.

D. Performance evaluation

We developed a performance evaluation library which allows us to monitor the CPU usage of a specified thread and the network traffic of the skin interface and the ROS skin interface. We determine the CPU usage by reading special pseudo files in the Linux */proc* file system and we calculate the network traffic using the *pcap* library. Our performance evaluation tool is embedded in the ROS environment such that we can measure performance indicators along with other values from the skin or the robot.

III. EXPERIMENTS

A. Performance indicators and their dependencies

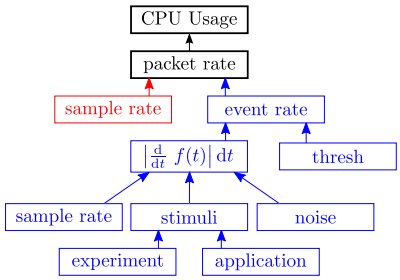


Fig. 6. Performance indicator dependency tree.

For building up a consistent and comprehensive performance analysis and comparing between data mode and event mode, we are particularly interested in two performance indicators: 1) the CPU usage and 2) the packet rate. A bounded and low packet rate is extremely important for a stable skin cell network and a low CPU usage of the skin driver is important for compact, autonomous robotic

skin applications. Both indicators are limiting factors for applications with very large robot skin. Our previous work [9] indicates that the event mode reduces packet rates. With the following experiments we formally prove that this is also true for real systems. Moreover we show that the CPU usage is coupled with the packet rate and we notice that the event packet rate has many dependencies. We execute experiments in order to determine the relationships between packet rate and CPU usage and between event packet rate and its many influence factors, e.g. sample rate and threshold. Based on the found relationships we develop empirical models such that we can extrapolate for very large scale robot skin with huge numbers of skin cells and higher sample rates. This will result in extrapolated performance estimates for future robotic skin systems. To keep track of the various dependencies between CPU usage, packet rate and its influence factors, we introduce a dependency tree (see Fig. 6). We observe that the packet rate influences the CPU usage as processing more packets requires more processing power. The data packet rate is only dependent on the sample rate f_{sample} and number of skin cells n:

$$pktRate = n \cdot f_{\text{sample}} \tag{4}$$

while the event packet rate has several dependencies and influence factors. The event packet rate is dependent on the event rate. However the relationship is not straight forward since multiple events occurring at the same time can be packed into one single event packet. How many events occur at the same time is loosely dependent on the external stimuli but mainly a stochastic process. On average relationship between event packet rate and event rate is linear and becomes better than linear as soon as the packet contains more than one event. Furthermore the event rate depends on the threshold and the accumulated change of the observed sensor value over time $\mathrm{d}t$ (see Eq. 1). The accumulated change is dependent on the sample rate, the exerted stimuli and noise. Of course tactile stimuli depend on the experiment or the application scenario. We note that in the event mode the performance indicators CPU usage and packet rate depend on the experiment. Thus experiments in both modes have to be repeated in the same manner to ensure valid comparisons.

B. Experimental setup

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 itself is inactive.
Stroking arm	The upper and lower arm skin of the robot is stroked by two persons, the robot arm itself remains inactive.
Hammering on end effector	The skin is not touched, one person hammers on the end effector. of the robot arm, the robot arm itself is inactive.
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 an arm action/movement.

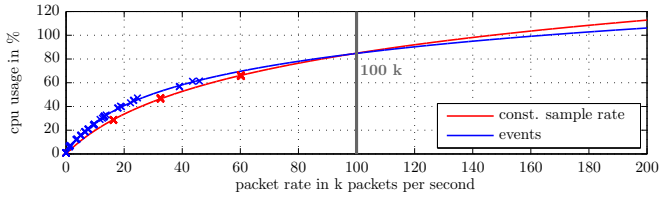
Fig. 7. Overview and description of experiments.

We perform the experiments on our robot TOMM (see Fig. 1). TOMM has two Universal Robot UR5 arms, each covered with 260 multi-modal skin cells. The experiments are only performed on one arm as listed and explained in Fig. 7. The experiments are chosen in such a way that the skin is stimulated in different modalities with different intensities (see Fig. 8) and should span over the most realistic application scenarios and should enclose the best and the worst case. We execute each experiment for 10 seconds with different sample rates in data and event mode. During the execution the stimuli are uniformly and continuously applied such that we can average for the evaluation. For finding good event thresholds we use the standard deviation of the signals as a first hint and then tune the thresholds heuristically to minimize errors. For comparability we use the same event thresholds as we determined in [9]. In the future dynamic thresholds might also contribute to a better event generator performance. We notice that the experiment *stroking while moving* with movements feature rich in acceleration changes will result in high event rates as almost all tactile modalities are stimulated at the same time. Nevertheless we expect for this experiment a packet rate below 100% which we will prove in the future.

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 on end effector	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 %)

Fig. 8. Mean event rate per cell per second and percentage of max. event rate for different modalities at 125 Hz sampling rate on skin cells; 100% event rate in one modality would equal the data rate in the data mode; the packet rate is the mean packet count per cell per second and 100% packet rate would equal the packet rate in the data mode; the colored fields highlight the dominant modalities within a given experiment.

C. CPU load and packet rate



Mode	Index	a	b	R^2
Constant sample rate	d	45.52	$5.453 \cdot 10^{-5}$	0.9993
Events	e	32.65	$1.239 \cdot 10^{-4}$	0.9969

Fig. 10. CPU usage in % vs. packet count; the CPU usage in % is the CPU usage of one of the eight cores of an Intel Core i7 4770 CPU; the data points belong to experiments listed in Fig. 7; the table shows the curve fitting results; the index d denotes the data mode and the index e denotes the event mode; the logarithmic model shows that handling data and event packets becomes more efficient for higher packet rates.

We expect that the CPU load is increasing for higher packet rates and that identical packet rates in data or event mode roughly cause the same CPU usage. Plotting the CPU usage of all the experiments against the packet rate reveals that the relationship is approximately logarithmic:

$$cpuUsage = f(pktRate) \tag{5}$$

$$f_{\{\mathrm{d,e}\}}(x) = a \cdot \log(b \cdot x + 1) \tag{6}$$

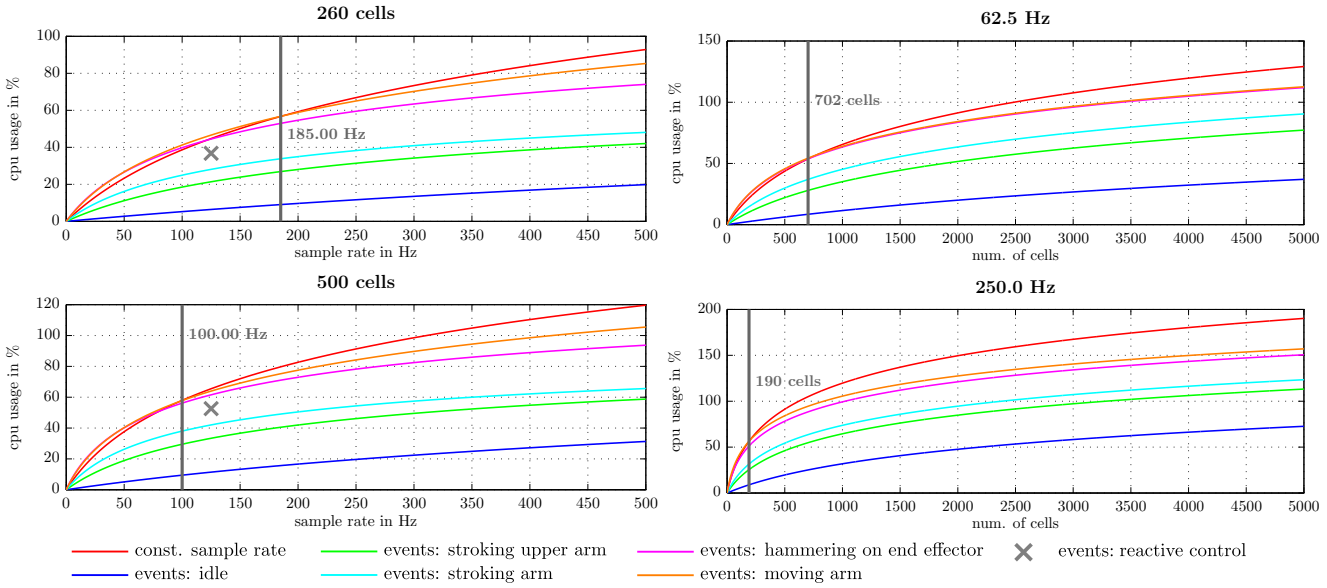
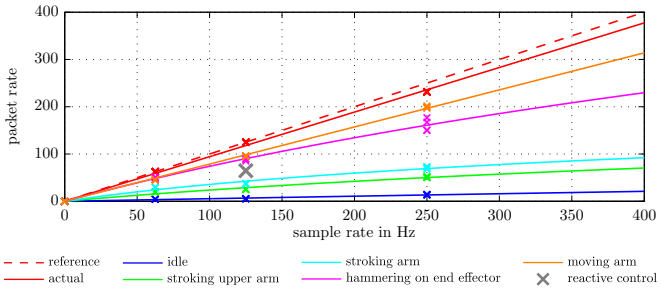


Fig. 9. The two graphs on the left side show extrapolations for the CPU usage with respect to the sampling rate for 260 or 500 cells; the gray bars are the worst case bounds for the sample rate after which the event-based system becomes superior; the two graphs on the right side show explorations for the CPU usage with respect to the skin cell count for a sampling rate of 62.5 Hz and 250 Hz; the gray bars indicate the worst case bounds for the number of cells after which the event-based system becomes superior.

The index d denotes the data mode and the index e denotes the event mode. a and b are fitting parameters which can be found in the table of Fig. 10 and x denotes the packet rate. The logarithmic model verifies that handling data and event packets becomes more efficient for higher packet rates. The relationship also indicates that the overhead of events becomes less significant for higher packet rates.

D. Event packet rate extrapolation



Mode / experiment	Index	a	b	R^2
Const. sample rate: ref.	d_1	1.0	–	–
Const. sample rate: actual	d_2	0.9435	–	0.9971
Events: idle	e_1	243.8846	$2.248 \cdot 10^{-4}$	0.9651
Events: stroking upper arm	e_2	85.0385	$3.215 \cdot 10^{-3}$	0.9764
Events: stroking arm	e_3	74.2308	$6.154 \cdot 10^{-3}$	0.9732
Events: hammering on end effector	e_4	330.2692	$2.512 \cdot 10^{-3}$	0.9888
Events: moving arm	e_5	$1.214 \cdot 10^{7}$	$1.687 \cdot 10^{-5}$	0.9983

Fig. 11. Packet rate per cell in packets per second vs. sample rate on skin cells; the *reference* and *actual* packet rates describe data packets rates of the system in data mode; the other packet rates are event packet rates of the corresponding experiment, e.g. *reactive control*; the data packet rate of the system in data mode is not dependent on the experiment; the table shows the curve fitting results.

The relationship between event rate and event packet rate is not straight forward. We investigate the relationship between the event packet rate, the experiment and the sample rate and predict that the relationship between stimulation intensity, sample rate and event packet rate is better than linear. In the worst case scenario the event rate equals the sample rate and consequently the performance of the system in event mode falls back to the one in data mode (see Fig. 11). The results show that experiments with a higher stimulation intensity, e.g. the moving arm experiment, have a event packet rate close to the linear bound while experiments with lower stimulation intensities are much better than linear. The relationship between the event packet rate and the sample rate can be modeled by logarithmic functions (see Fig. 11):

$$cellPacketRate = g(f_{\text{sample}},\, n = 1) \tag{7}$$

$$g_{\mathrm{d}_i}(x, n) = a \cdot n \cdot x \tag{8}$$

$$g_{\mathrm{e}_i}(x, n) = a \cdot n \cdot \log(b \cdot x + 1) \tag{9}$$

The extrapolations in Fig. 11 indicate that the performance edge of the event based system increases for higher sample rates. The high speed arm movement experiment seems to embody the worst case bound for our sets of experiments.

E. CPU usage extrapolation

We combine the models of CPU usage versus packet rate and packet rate versus sample rate and get extrapolations on the CPU usage for larger numbers of skin cells or for higher sample rates:

$$cpuUsage = h(f_{\text{sample}},\, numOfCells) \tag{10}$$

$$h(x, n) = f \circ g = f(g(x, n)) \tag{11}$$

The results in Fig. 9 show that the event-based system becomes superior for higher sample rates and more skin cells. The gray bars in Fig. 9 display the lower bounds for the

number of skin cells and the sample rate. As long as the number of cells and the sample rate stay above these bounds the event based system has superior performance, e.g. in the worst case scenario with 260 skin cell at 250 Hz the event based system shows a reduction of the packet rate by 21.2% and the CPU usage by 2.72%. However if we increase the number of skin cells to 5000 these values change to 21.2% and 17.46%, respectively (see Fig. 12).

Mode	Number of cells	Sample rate	Rel. packet rate	Rel. CPU usage
data	–	–	100%	100%
events	500	62.5 Hz	79.4%	104%
events	500	100 Hz	78.7%	100%
events	500	500 Hz	78.4%	88.1%
events	100	250 Hz	78.8%	107%
events	190	250 Hz	78.8%	100%
events	260	250 Hz	78.8%	97.2%
events	5000	250 Hz	78.8%	82.54%

Fig. 12. Worst case CPU usage extrapolations for different numbers of skin cells and sample rates; the packet rate and CPU usage is relative with respect to the conventional data sampling mode.

IV. CONCLUSIONS

We discussed the realization of event generators in robotic skin and evaluated in detail how the performance of robotic skin using events compares to a conventional data sampling system. We evaluated the two most important performance indicators, CPU usage and packet rate in event and data mode and extrapolated these indicators for larger number of skin cells and higher sample rates. The results show that the event based system indeed proves to have a higher performance and the extrapolations indicate increasing superiority for more skin cells and higher sample rates. We extrapolate for 5000 cells and show that in the worst case scenario the packet rate reduces to 78.8% of the conventional data mode and the CPU usage to 82.54% of the conventional data mode. In other words, the event-based system at least reduces the packet rate by 21.2% and the CPU usage by 17.46%. For low sample rates and small skin networks the event-based system has a higher CPU usage per packet ratio because of the overhead and a less efficient event packet unpacker. However the performance of the event-based system can be further improved when we use the optimal packet size for event packets [9]. The results also show that the event packet rate is always smaller and in worst case equal to the data packet rate. All together the event-based robot skin system shows good improvements in CPU usage and network load and brings us one step further to large scale robot skin deployment.

ACKNOWLEDGMENT

This work has received funding from the European Community's Seventh Framework Program (FP7/2007-2013) under grant agreement no. 609206.

REFERENCES

[1] L. D. Harmon, "Automated Tactile Sensing", in *The International Journal of Robotics Research*, vol. 1, no. 2, pp. 3-32, 1982.

[2] P. Mittendorfer, E. Yoshida, and G. Cheng, "Realizing whole-body tactile interactions with a self-organizing, multi-modal artificial skin on a humanoid robot", in *Advanced Robotics*, vol. 29, no. 1, pp. 51-67, February 2015.

[3] G. Cannata, R. Dahiya, M. Maggiali, F. Mastrogiovanni, G. Metta, and M. Valle, "Modular Skin for Humanoid Robot Systems", in *4th International Conference on Cognitive Systems (CogSys)*, 2010.

[4] E. Baglini, S. Youssefi, F. Mastrogiovanni and G. Cannata "A Real-Time Distributed Architecture for Large-Scale Tactile Sensing", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1663-1669, 2014.

[5] M. Strohmayr and D. Schneider, "The DLR artificial skin step I: Uniting sensitivity and collision tolerance", in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1012-1018, 2013.

[6] M. Strohmayr and D. Schneider, "The DLR artificial skin step II: Scalability as a prerequisite for whole-body covers", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4721-4728, 2013.

[7] S. Youssefi, S. Denei, F. Mastrogiovanni, and G. Cannata, "A Real-time Data Acquisition and Processing Framework for Large-scale Robotic Skin", in *Robotics and Autonomous Systems*, vol. 68, pp. 86-103, 2015.

[8] S. Youssefi, S. Denei, F. Mastrogiovanni, and G. Cannata, "Skinware 2.0: A real-time middleware for robot skin", in *SoftwareX*, vol. 3, pp. 6-12, 2015.

[9] F. Bergner, P. Mittendorfer, E. Dean-Leon and G. Cheng, "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)*, pp. 2124-2129, 2015.

[10] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 x 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor", in *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566-576, February 2008.

[11] C. Posch, D. Matolin, and R. Wohlgenannt, "An Asynchronous Time-based Image Sensor", in *IEEE International Symposium on Circuits and Systems*, pp. 2130-2133, May 2008.

[12] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, "Retinomorphic Event-Based Vision Sensors: Bioinspired Cameras With Spiking Output", in *Proceedings of the IEEE*, vol. 102, no. 10, pp. 1470-1484, October 2014.

[13] C. Bartolozzi, and G. Indiveri, "Selective Attention in Multi-Chip Address-Event Systems", in *Sensors*, vol. 9, no. 7, pp. 5076-5098, 2009.

[14] C. Bartolozzi, F. Rea, C. Clercq, and M. Hofstatter, "Embedded neuromorphic vision for humanoid robots", in *IEEE Conference on Computer Society*, pp. 129-135, June 2011.

[15] R. Benosman, S.-H. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan, "Asynchronous frameless event-based optical flow", in *Neural Networks*, vol. 27, pp. 32-37, 2012.

[16] S. Caviglia, M. Valle and C. Bartolozzi "Asynchronous, event-driven readout of POSFET devices for tactile sensing", in *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2648-2651, 2014.

[17] L. Seminara, L. Pinna, A. Ibrahim, L. Noli, M. Capurro, S. Caviglia, P. Gastaldo and M. Valle "Electronic Skin: achievements, issues and trends", in *Procedia Technology*, pp. 549-558, 2014.

[18] L. Seminara, L. Pinna, A. Ibrahim, L. Noli, S. Caviglia, P. Gastaldo and M. Valle "Towards integrating intelligence in electronic skin", in *Mechatronics*, 2015. In Press.

[19] G. Rovere, C. Bartolozzi, N. Imam and R. Manohar, "Design of a QDI Asynchronous AER Serializer/Deserializer Link in 180 nm for Event-Based Sensors for Robotic Applications", in *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2712-2715, 2015.

[20] M. Neugebauer, and K. Kabitzsch, "A New Protocol for a Low Power Sensor Network", in *IEEE International Conference on Performance, Computing and Communications*, pp. 393-399, 2004.

[21] M. Miskowicz, "Send-on-delta Concept: An Event-Based Data Reporting Strategy", in *sensors*, vol. 6, no. 1, pp. 49-63, January 2006.

[22] K. Staszek, S. Koryciak and M. Miskowisz, "Performance of send-on-delta Sampling Schemes with Prediction", in *IEEE International Symposium on Industrial Electronics (ISIE)*, pp. 2037-2042, June 2011.

[23] J. Wu, Q. Jia, K. H. Johansson, and L. Shi, "Event-Based Sensor Data Scheduling: Trade-Off Between Communication Rate and Estimation Quality", in *IEEE Transactions on Automatic Control*, vol. 58, no. 4, pp. 1041-1046, April 2013.