Fakultät für Elektro- und Informationstechnik
Lehrstuhl für Medientechnik

Visuo-haptic environment perception for autonomous robotic systems

Nicolas F. Alt

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:		Univ.-Prof. Dr. sc. techn. Andreas Herkersdorf
Prüfer der Dissertation:	1.	Univ.-Prof. Dr.-Ing. Eckehard Steinbach
	2.	Univ.-Prof. Gordon Cheng, Ph.D.

Die Dissertation wurde am 22.04.2015 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 21.12.2015 angenommen.

Visuo-haptic environment perception for autonomous robotic systems

Nicolas Alt, M.Sc. (hons)

November 29, 2016

Acknowledgments

This doctoral thesis represents large parts of my research carried out at the Chair of Media Technology at Technical University of Munich, initially within the DFG excellence initiative research cluster “Cognition for Technical Systems” (CoTeSys).

I would like to express my deep gratitude to my supervisor Prof. Eckehard Steinbach for his enduring support and guidance while pursuing my doctorate. The door to his office was always open for discussions, and he ensured a highly scientific atmosphere at his chair. Also, I would like to thank Prof. Gordon Cheng and Prof. Andreas Herkersdorf for their work in the doctoral committee. Furthermore, I am very much obliged to Patrick Rives for inviting me to his lab at INRIA Sophia Antipolis.

My sincere appreciation goes to my colleagues at the chair for many stimulating discussions, building up robots together and solving every administrative issue. Especially I would like to mention Werner Maier, Clemens Schuwerk, Jingyi Xu, Sebastian Hilsenbeck, Robert Huitl and Florian Schweiger. I would also like to thank my colleagues Martin Lawitzky, Daniel Althoff, Omiros Kourakos, Matthias Rambow and Alexander Mörtl for setting up workshop demos in the CoTeSys robotics lab together, often until late into the night. Also, I would like to acknowledge the work of my students during numerous theses and projects.

Finally, I feel profoundly grateful to my parents for their support and continuous encouragement throughout my years of study and my doctorate, and to my girlfriend Florence Bonnafé for her support, critical thinking and for proofreading this text.

Thank you!

Abstract

Novel autonomous robots – such as domestic robots, service robots or cognitive robots – operate in so-called unstructured environments, like private homes and offices. An essential skill for such systems is perception, i.e. the acquisition of comprehensive knowledge about the scene. Perception systems represent an early stage of intelligence and comprise sensors, signal processing, model building as well as recognition. Visual perception is essential for modeling of complex scenes, but it has limitations – for instance in the presence of transparent or reflective objects. Moreover, the visual modality is insufficient during manipulation, i.e. when a robot interacts with its environment. Haptic perception — i.e. sensing of touch, contact and force – must be integrated as a second modality, using appropriate sensors. Humans serve as a model for this approach, as they also integrate vision and haptics for manipulation tasks.

This work investigates joint visuo-haptic sensing and modeling, and presents appropriate task planners. A visuo-haptic sensor is proposed, which is based on a passive, deformable element mounted on the actuator. Deformations of this element are observed by a camera, and a force/pressure profile is deduced from the image. The same camera is also used for visual scene observation, such that visual and haptic data are acquired coherently. Compared to separate sensor systems, this setup reduces system complexity and costs.

Approaches of joint visuo-haptic modeling are presented for three scenarios, namely navigation in home and office environments, geometry reconstruction as well as classification and manipulation of deformable objects. First, a model for navigation of mobile platforms is proposed, which represents haptic properties of obstacles on a per-object level. Building on data acquired by the visuo-haptic sensor, a navigation graph is extended by so-called "manipulation nodes". These nodes represent obstacles that can be manipulated during navigation, e.g. in order to reach a blocked area in the room. Second, the reconstruction of geometry models from objects is considered. A transparency detector is proposed, which searches for geometric inconsistencies caused by refraction effects. Within transparent regions, geometry cannot be obtained by vision, and is instead explored haptically, i.e. by touching the object with a sensor tip mounted on a robot arm. The exploration plan is generated using an uncertainty estimator based on the biharmonic distance. Third, a deformation model is presented for objects with thin-walled structures, such as plastic bottles. Haptic features for local stiffness are proposed and extracted from a dataset of generated object models during a simulation process. An object classifier, which discriminates object classes based on these stiffness features, is trained with the dataset and tested on real objects. Furthermore, a planner is presented that considers the deformation of objects to find stable grasp configurations.

Contents

Notation viii

1 Introduction 1
- 1.1 Robots in unstructured environments . . . 1
- 1.2 Visuo-haptic perception and manipulation . . . 3
- 1.3 Contributions of this work . . . 6
- 1.4 Structure of this dissertation . . . 9

2 Background and related work 11
- 2.1 Sensors for autonomous robots . . . 11
  - 2.1.1 Haptic and tactile sensors . . . 12
- 2.2 Object models and reconstruction . . . 14
  - 2.2.1 Visual reconstruction . . . 14
  - 2.2.2 Transparency detection . . . 16
  - 2.2.3 Tactile reconstruction . . . 17
  - 2.2.4 Modeling of elastic materials . . . 19
  - 2.2.5 The finite element method . . . 21
- 2.3 Detection and tracking . . . 23
  - 2.3.1 Images features . . . 23
  - 2.3.2 Visual tracking . . . 24
  - 2.3.3 Homography estimation and decomposition . . . 26
  - 2.3.4 Combined detector and tracker . . . 28
  - 2.3.5 Visual classification and recognition . . . 29
  - 2.3.6 Mapping and navigation . . . 30

3 Visuo-haptic sensors 33
- 3.1 Motivation . . . 33
- 3.2 Modeling of the deformable element . . . 35
- 3.3 Foam-based sensor for laminar contacts . . . 39
  - 3.3.1 Tracking of foam deformation . . . 40
  - 3.3.2 Analysis of measurement accuracy . . . 42
  - 3.3.3 Fitting of geometric primitives . . . 45

3.4 Beam-based single point 6D force-torque sensor . . . 46
3.5 Tracking objects and scene . . . 48
3.6 Experiments . . . 51
3.6.1 Sensor accuracy . . . 51
3.6.2 Exploration of objects with a mobile platform . . . 53
3.6.3 Naïve haptic mapping with a mobile platform . . . 54
3.6.4 Grasping with the foam-based sensor . . . 56
3.6.5 Manipulation with the beam-based sensor . . . 57
3.7 Summary . . . 58

4 Joint planning of navigation and manipulation **61**
4.1 Haptic tags . . . 62
4.2 Graph-based planner . . . 64
4.2.1 Integration of haptic information . . . 65
4.3 Local planning . . . 68
4.3.1 Direct push . . . 69
4.3.2 Sideward push . . . 70
4.3.3 Transparent objects . . . 73
4.3.4 Special operations . . . 74
4.4 Experiments . . . 75
4.5 Summary . . . 77

5 Visuo-haptic geometry fusion **79**
5.1 Effects of transparency . . . 80
5.2 Transparency detector . . . 81
5.2.1 Local background model . . . 82
5.2.2 Detection of inconsistent geometry . . . 84
5.3 Geometry estimation in transparent regions . . . 85
5.3.1 Surface extrapolation . . . 87
5.4 Haptic exploration of geometry . . . 88
5.5 Experiments . . . 90
5.5.1 Transparency reconstruction . . . 90
5.5.2 Simulation of haptic exploration . . . 92
5.6 Summary . . . 95

6 Deformation models for thin-walled objects **97**
6.1 Simulation of object deformation . . . 99
6.1.1 Volumetric object models . . . 99
6.1.2 Parametric model generation . . . 100
6.1.3 FEM-based elasticity simulation . . . 101

6.1.4 Grasp patterns . . . 102
6.1.5 Modeling the contents of containers . . . 103
6.1.6 Stiffness maps and features . . . 105
6.2 Grasp planning for deformable objects . . . 107
6.3 Haptic object classification using stiffness features . . . 110
6.3.1 Feature selection and exploration planning . . . 110
6.3.2 Scenarios for haptic exploration . . . 112
6.3.3 A process for visuo-haptic classification . . . 114
6.4 Results and experiments . . . 116
6.4.1 Simulation-based models . . . 116
6.4.2 Exploration of real objects . . . 118
6.5 Summary . . . 124

7 Conclusion and Outlook **127**
7.1 Conclusion . . . 127
7.2 Outlook . . . 129

List of Figures **134**

List of Tables **137**

Publications by the author **139**

Bibliography **140**

Notation

Abbreviations

2D/3D	Two-/Three-dimensional space
CPU	Central Processing Unit
DoF	Degrees of Freedom
dpi	Dots Per Inch
ESM	Efficient Second order Minimization (tracker) [67]
ICP	Iterative Closest Point [84]
IMU	Inertial Measurement Unit
IP	Internet Protocol
ISO	International Organization for Standardization
FEM	Finite Elements Method
GPU	Graphics Processing Unit
HD	High Definition (Full HD: 1920×1080 px)
KLT	Kanade-Lucas-Tomasi tracker
LWR	Lightweight Robot
MSE	Mean Squared Error
NCC	Normalized Cross-Correlation (coefficient)
PCA	Principal Component Analysis
PUR	Polyurethane (foam)
RBF	Radial basis function
ROS	Robot Operating System
SDF	Signed Distance Function
SIFT	Scale-Invariant Feature Transform [63]
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
px	Pixel (picture element)
RANSAC	Random Sample Consensus [28]
USB	Universal Serial Bus
Voxel	Volumetric pixel – value in a 3D grid

Symbols

A	Area
δ	Deformation of a soft material
d	Distance
E	Elastic modulus (Young's modulus)
$\mathcal{F}$	Feature
F	Force
K	Intrinsic camera matrix
$\mathcal{M}$	2D grid map
$\mathbf{n}$	Normal vector
$\mathcal{N}$	Neighborhood (within d around x: $\mathcal{N}_d(x)$)
P	Pressure
σ	Standard deviation (of Gaussian)
$\mathcal{S}$	Score value
$\mathcal{T}$	Transformation in 2D/3D (rotation and translation)
$\tilde{\mathcal{T}}$	Coordinate frame associated to a transformation $\mathcal{T}$
$\mathbf{u}$	Point in an image with components $\mathbf{u}_{x,y}$
$\tilde{\mathbf{u}}$	Point in an image, homogeneous coordinates $\tilde{\mathbf{u}} = c \cdot [x, y, 1]^T$
$\mathbf{X}$, $\mathbf{X}^{(F)}$	3D point in the world with components $\mathbf{X}_{X,Y,Z}$, in frame F

1 Introduction

This chapter starts with an introduction to the context of this work, i.e. autonomous robots operating in unstructured scenes. Next, a definition of visuo-haptic perception and manipulation is given, together with some examples. Finally, a summary of the contributions of this work is presented.

1.1 Robots in unstructured environments

Robots have been successfully used in industry for decades. Within these environments, the primary goals are to perform tasks faster, with greater precision, fewer interruptions and lower costs than human workers. Some applications require to move heavy loads and to operate in hazardous environments. Most commercially available robots have been optimized for these needs, since there is a huge demand of such systems in industrial production. As a consequence, they are equipped with highly precise sensors for self-monitoring, and their connection elements or links are built rigidly to avoid deformations. This design ensures repeatability and high absolute precision of less than 1 mm. Once programmed and calibrated, these systems reproduce trajectories with great accuracy, even without using any external sensors. Preprogramming only makes sense in a so-called structured environment, whose geometry is known up to a low number of well-defined unknowns. For instance, in automation systems, the structure and dimensions of machine parts are known exactly. Strong constraints are imposed to workpiece positions and object motions, e.g. from a linear drive, and sensors are mounted at fixed locations. The type or identity of an object is either known before, or determined by dedicated technologies, such as RFID or 2D bar codes. Tolerances are kept low by design. Sensors as simple as a light barrier often suffice for environment perception. The KUKA Titan is an industrial robot with the highest lifting capabilities on the market – and still offers a repeatability of ± 0.2 mm. Obviously, high requirements on rigidity and precision, both at the sensor and the mechanical level, go along with considerable costs.

Robots that operate in unstructured environments, on the other hand, require a much different skill set compared to their ancestors. They represent a new paradigm of robotics, which has seen tremendous interest in research and has recently also been implemented in some commercial products. In unstructured scenes, many types of objects are present, without prior knowledge about their identities and poses (i.e. position and rotation). This also applies to persons, furniture and the structure of the room. Robots need to observe the scene constantly, interpret their observations and react accordingly. Only weak prior assumptions can be

(a) Samsung Navibot **(b)** Bosch Indego

Figure 1.1: Commercial domestic robots are available for specialized tasks, such as vacuum cleaning *(a)* and lawn mowing *(b)*. Many robotic vacuum cleaners rely already on visuo-haptic perception, since they integrate a camera for mapping and bumpers for collision detection. Currently, these systems use relatively simple environment models. Their capabilities could be improved considerably by richer environment perception. Images: Samsung, Bosch.

made, such as the existence of a floor plane in indoor scenes. Precision and repeatability are less important than perception and adaptability. Depending on the required skills and the application, there are various non-exclusive and non-strict classifications of such robots. Skill sets and complexity of these robots vary greatly, even within one class:

- Domestic robots, which perform various service tasks in private homes
- Cooperative robots, which work together with humans, e.g. in production
- Service robots, which assist humans in various environments
- Autonomous robots, which work autonomously for an extended period of time
- Cognitive robots, which learn and reason about complex scenes
- Humanoid robots, which have a body shape and dexterity similar to humans

Cleaning platforms, such as robotic vacuum cleaners, see Fig. 1.1, can be classified as domestic, service and autonomous robots. Similar systems are available for other cleaning tasks or lawn mowing. The first commercially successful models used very simple sensors and planning methods, while most of the current models perceive their environment with cameras and contact sensors for planning and mapping. Obstacles in the environment are not manipulated, but avoided. Cooperative robots enable new concepts in industrial production, allowing humans and robots to work together in the same working space. A recent report [98] investigates the possibilities of this approach and presents first case studies and visions for future joint human-robotic production environments. There is also substantial research in the area of humanoid service robots, which have very promising applications in households, in medical applications or in the domain of care for the elderly, see Fig. 1.2. The PR2, see

(a) Willow Garage PR2 **(b)** Fraunhofer Care-o-Bot 3 **(c)** Honda ASIMO [85]

Figure 1.2: Many humanoid robots with wheeled platforms or legs have been presented for research applications. Typically, they are equipped with a large number of various sensors. Apart from their high price, the performance of their perception and manipulation systems is not yet sufficient for practical applications outside of lab environments. Images: Willow Garage, Fraunhofer IPA/R. Bez, Honda.

Fig. 1.2a, is a "semi-humanoid" research platform, which helped researchers build a large number of robot demos for different everyday tasks, such as grasping of household objects [45] or folding of clothes [74].

Lightweight robot arms such as the KUKA lightweight arm (LWR) [14] or the Universal Robots UR5/10 [97] offer active compliance based on numerous sensors, enabling safe cooperation with humans. The Baxter robot [38] is equipped with two lightweight arms, many sensors and vision-based planning systems. It takes over dangerous, repetitive or unproductive tasks from humans, such as (un)loading machines and preparation of tools or components.

In the following, the term "autonomous robot" will be used for any robot, which operates autonomously in unstructured environments.

1.2 Visuo-haptic perception and manipulation

One of the common core functions of all autonomous robots is the perception of their environment – i.e. the acquisition of an adequate level of understanding about persons, objects, obstacles, scene structure and accessible areas around the robot. Robot perception comprises sensors (such as cameras, force/touch sensors, microphones, etc.) and subsequent processing of the provided data in order to obtain an abstract representation of the environment – a model. Perception systems provide different levels of abstraction, such as – in case of vision

(a) Soft/transparent obstacles in office scene **(b)** $F = 40\,\mathrm{N}$ – ✗ **(c)** $F = 50\,\mathrm{N}$ – ✗ **(d)** $F = 60\,\mathrm{N}$ – ✓

Figure 1.3: Manipulation in unstructured environments requires visuo-haptic perception. Visuo-haptic models provide a richer representation of "soft" obstacles, such as vases, boxes and paper bins, facilitating navigation in home or office scenes *(a)*. Grasp plans must consider the stiffness of objects to avoid extensive deformation *(b)* or destruction *(c)*. Rigid objects *(d)* may require much larger grasping forces. In both scenarios, haptic data may complement geometry information within transparent regions.

– images, depth maps, geometric models, geometric primitives, object identity, object class or other semantic information. This can be compared to perception in humans or animals, which consists of specialized parts of the brain ("visual intelligence" [43]) dedicated to the visual, auditory and haptic senses.

In manipulation, perception is a multimodal problem, mainly involving vision and haptics. As long as objects are not in contact with the robot, information is acquired via cameras or other vision-based remote sensing methods. Once contact with an object has been established, perception is extended to the haptic modality. Haptics consist of the tactile and kinesthetic sense. The former represents contact events as well as vibrations when sliding over a surface and exhibits a relatively large spectral range. The latter is connected to (usually low-frequency) forces applied to an object as well as the physical locations in space. Other modalities are beyond the scope of this work, even though they may be of high relevance in certain scenarios: Auditory perception is essential for human-robot interaction, outdoor environments require temperature or moisture sensing.

Many autonomous robots are designed to cooperate with humans in households, office spaces or public buildings. One important characteristic of these *human environments* is their design according to the needs of humans. Therefore, robots must adapt to the skills and capabilities of humans to a certain degree in order to operate in these surroundings. Only general assumptions about these environments can be made by the robot – such as the type

of the environment, the existence of structures like walls, doors or tables, the navigability on floors as well as general rules for the location of certain types of objects. Large generic databases of objects that may be encountered in the environment enable object classification, see Sec. 2.3.5. However, there is no predefined specific knowledge about the environment as in industrial robotics, such as locations, poses or identities of specific objects. All this knowledge must be acquired by the robot itself, using its perception system. Adaptability and relative accuracy of the end-effector with respect to objects are of relevance, rather than high absolute accuracy, repeatability and speed. This has important implications for the hardware and mechanical design of autonomous robots: Absolute accuracy of sensors can be lower, and connection elements can be built lighter, since high rigidity is not essential.

Navigation is an important skill for mobile autonomous robots. Their perception system must be able to recognize navigable areas – i.e. unoccupied areas of the floor, as well as relevant building structures, such as doors, elevators and stairs. Also, systems for object detection or recognition are needed to identify obstacles or objects that should be manipulated. The required skill level as well as the required capabilities for environment understanding depend on the specific tasks a robot performs. Office or indoor scenes exhibit "hard obstacles", such as walls or heavy tables, which are easy to observe with a visual sensor and may be modeled with a binary occupancy map. However, there are also "soft" obstacles as well as objects that are difficult to observe with visual sensors – such as boxes, paper bins or transparent vases, see Fig. 1.3a. Multimodal environment models, which also integrate haptic data, represent the properties of such objects and allow for more advanced navigation schemes. In most cases, modeling at a relatively low level of detail is sufficient. Additional perception skills are required for robots navigating in nature, within cities or in other environments. These are more diverse and less structured than indoor environments – even the detection of navigable areas becomes more complex. Reliable perception is a prerequisite for any system providing higher levels of intelligence – such as complex action planning, or anything generally referred to as artificial intelligence. Yet, these aspects are beyond the scope of this thesis.

Another essential skill for autonomous robots is grasping, which allows for controlled manipulation of the environment. It is a complex task that requires detailed object models comprising local geometric, visual and haptic data. Models of visual appearance enable fast classification and pose estimation, geometric models are essential to find possible grasp points, and haptic models are important to verify the stability of grasp configurations, as well as to adjust the grasping force. If an object is detected based on a general model, additional object states and parameters may need to be determined for reliable manipulation. Some states – such as the fill level of a glass – change quickly and must be estimated before each interaction with the object. Haptic parameters are especially important for deformable objects. Fig. 1.3 shows deformable or soft objects that are typical for household environments. If grasping parameters are chosen incorrectly, manipulation may result in unwanted results – such as overflow of the bottle's contents in Fig. 1.3b or even destruction Fig. 1.3c. Different grasping forces may be required for objects that are similar in geometric and semantic aspects, but

differ in the haptic domain. For instance, the stiff glass in Fig. 1.3d requires a much larger grasping force than the deformable plastic cups.

Manipulation tasks are based on information provided by perception and are closely coupled with it. Similar to perception, different levels of abstraction are used for manipulation planning: On a semantic level, a manipulation plan might describe how to bring a certain type of object, such as a mug, in relation to another object, such as a coffee machine. This level is associated with semantic scene knowledge acquired by perception. In more detailed plans, the object identity, approximate locations and trajectories are specified. At the most detailed level, expected forces and poses are calculated at each time step and compared to current sensor readings. This includes also grasp patterns while the robot is in contact with an object. Here, the association between manipulation and perception becomes much closer, connecting the two systems together in a control loop. Manipulation is controlled based on data from perception, which in turn depends on the manipulation operation. Perception algorithms used during manipulation must therefore run in realtime. This coupling between perception and manipulation is a fundamental difference to perception in multimedia applications, where the input data can be processed offline and does not depend on the results of the perception system.

1.3 Contributions of this work

This work investigates selected aspects of joint visuo-haptic perception for robotics in unstructured environments. Joint visuo-haptic processing at the sensor level is considered first. Typically, robots rely on distinct sensor systems for these two modalities – such as cameras on the head, near the end-effectors and on other parts of the body, force-torque sensors in the arm as well as distributed touch and contact sensors on manipulators, see also Sec. 2.1. These sensors provide two fundamentally different representations of the same physical objects and of events, which generally provide complementary or non-redundant information. Even in cases when data from one representation can be transformed into the other, such as for shape information, accuracy and domain are typically different. For many tasks, data from the two modalities must be fused in a coherent way. For instance, a grasping system first uses visual pose estimation to place the gripper accordingly. The event of contact between the gripper and the object is determined using touch sensors, since visual sensors do not provide adequate accuracy. The touch event goes along with a refinement of the object's pose estimate. Next, haptic and visual sensors work together to control the grasping force and determine the reaction of the object. Synchronization between the two sensors is crucial for stability.

Chapter 3 introduces the concept of visuo-haptic sensors, which use a single camera for both visual and haptic sensing. Dedicated haptic sensors are replaced by a passive deformable element, such as a piece of plastic/rubber foam or a rod of spring steel. This element may also be an integral part of the robot, such as a link of the arm. Since its deformation characteris-

tics are known, force and pressure values can be obtained from deformations tracked by the camera. This concept provides a number of advantages compared to separate sensor systems: First of all, visual and haptic data are naturally coherent, since they are acquired from the same sensor. Furthermore, the deformable element makes the actuator naturally compliant, i.e. it retracts as soon as a force is applied. Soft components on robots offer advantages for safety and ease requirements on latency and control rate. Finally, system complexity is reduced considerably, since dedicated active sensor models are replaced by passive components. Sensors are often located at extremal parts of the robot, which goes along with an additional effort for cabling and maintenance. Cameras offer inherent cost advantages, since they can be built very small, be placed at beneficial locations and have become very low-cost, driven by a large consumer market.

Implementations of the visuo-haptic sensor are presented for mobile platforms [3, 5, 6], the fingers of a gripper [7], as well as for tools attached to robot arms. The mobile platform uses the sensor to detect collisions with obstacles and acquires "haptic tags," which are used for tasks related to navigation. Haptic tags provide a per-object representation of haptic properties, such as friction force, deformation, object motion and contact shape. Mounted on a gripper, the visuo-haptic sensor replaces costly laminar tactile sensors and provides measurements of the current gripper position, grasping forces, pressure profile as well as deformation characteristics of the object. Finally, an implementation based on a thin metallic rod, which is mounted between a robot arm and a tool provides up to 6D force-torque values and replaces corresponding dedicated modules.

Apart from sensing, multimodal perception also requires appropriate object and environment models. Such models represent visual and haptic object properties in a consistent way. Visual properties include the geometric shape of the object surface as well as its texture. Haptic properties include weight, (local) stiffness and surface roughness. Yet, it must be considered that capabilities of haptic sensors are very diverse, concerning for instance their spatial resolution as well as the accuracy and range of force measurements. An overly detailed model, which can neither be acquired nor verified with the available haptic sensors, is useless. Of course, coarse models can be derived from more accurate object models. It is beneficial to provide a model adapted to the level of detail required by a given manipulation task.

Three approaches of visuo-haptic object modeling are proposed and discussed in the context of manipulation tasks. First, Chapter 4 investigates joint navigation and manipulation planning for small mobile platforms with limited haptic sensors [6]. The environment is typically represented by a 2D map acquired with a laser scanner. Accordingly, the visual object representation is a 2D shape or footprint. Additional visual features, such as texture, may be included – but generally, a complete geometric model is not available. Haptic properties are represented using the haptic tag mentioned above – associated either with the entire object, or with a low number of contact points on that object. This representation allows for planning of simple manipulation operations, such as moving an obstacle out of the way or opening a door. A topological graph is used to represent both the structure of the map, i.e. its skeleton, as well

as manipulable objects in the environment. Indoor environments, see Fig. 1.3a, exhibit many "soft" obstacles that are manipulable even by small platforms. Using standard path planners, cost-effective joint navigation and manipulation plans are created. As in existing work, navigation plans include abstracted navigation instructions, similar to street-level navigation. In addition, the proposed plans represent the above mentioned manipulation tasks associated to navigation problems. Costs for navigation and manipulation are considered in a consistent way. Using additional constraints, alternative navigation/manipulation solutions can be identified – such as pushing an obstacle aside, or taking a detour to drive around it. The representation is flexible and also allows for the integration of specific navigation/manipulation tasks such as utilizing doors or elevators.

Second, an approach to reconstruct object geometry jointly from vision and haptic exploration is presented in Chapter 5. Existing multi-view approaches obtain the surface geometry of an object based on the premise of Lambertian reflection. This means that brightness and color of each point on the surface are independent of the viewing direction. Approaches such as KinectFusion [76] combine several views of an object based on this consistency assumption. Like that, they create highly accurate geometric models with a noise level below that of the raw sensor data. The Lambertian model is invalid for transparent and reflective materials, such that multi-view reconstruction fails in these cases. Transparent objects do not follow this model and show other complex effects – such as background refraction. This effect is exploited by the multi-view transparency detector proposed in [4]. It provides a 3D estimate of transparent regions after a depth camera has been moved around the object. These regions represent the primary failure cases of visual reconstruction in indoor scenes. Reliable information about the object geometry within these regions can be obtained haptically, i.e. by touching the object. To this end, a haptic exploration planner is presented, which is based on the available visual geometry as well as the detected transparency region. A surface extrapolation method based on radial basis functions [16] is applied to the available geometry. The certainty of this extrapolated surface decreases with increasing distance to support points. Furthermore, the density of support points needs to be larger in complex regions of the surface. In order to plan the next exploration point, a score is calculated based on an efficient mesh distance measure, the biharmonic distance [60]. The surface is refined iteratively by integration of new "haptic" points in the extrapolation process and subsequent recalculation of the score. The resulting complete geometric model is essential for accurate grasp planning, and also allows building and learning of visual object models.

Third, a visuo-haptic object model for grasping and classification of thin-walled deformable objects is presented in Chapter 6. The object geometry is represented on a detailed level, i.e. an accurate surface mesh with texture is available or acquired using multi-view approaches. Such a model allows for visual object recognition using a variety of approaches, see Sec. 2.3.5. Thin-walled objects, such as bottles, cups and other containers used in human environments exhibit a strong relation between local stiffness and geometry – e.g. they are soft on flat planer areas and hard near edges. Additionally, these objects exhibit several challenges for vision-

based approaches, offering a natural motivation to include the haptic modality: Containers are often partially transparent, allowing only for the acquisition of partial visual geometry models. Furthermore, their deformation behavior cannot be predicted solely based on vision, since they are made of different materials such as plastics of varying elasticity and thickness, glass or ceramics.

In the proposed approach, volumetric models are built based on a surface mesh as well as typical parameters for wall thickness and material type. During an offline simulation phase, the deformation behaviour and local stiffness is determined for generic grasp patterns. The patterns are applied to every point on the surface, using simulation based on the Finite Element Method (FEM). This process is repeated for a large dataset of multiple object classes with variations of several shape parameters within each class. The resulting database comprises typical containers used for drinks, food, detergents and cosmetic products class [9]. Stiffness maps showing the local stiffness for each point on the surface as well as corresponding stiffness features are obtained for each object model.

An approach for haptic object classification is proposed, which relies on the results from an initial visual stage. Visual methods cannot reliably determine material properties, nor subtle yet important geometric details. Therefore, results from the visual stage are ambiguous – they may contain multiple different candidate objects as well as objects associated to multiple haptic models. The haptic recognition stage must resolve these ambiguities based on haptic properties. Since haptic exploration – i.e. touching the object on several points – is a time-consuming process, an efficient exploration plan must be generated. The goal is to use as little touch points as possible and to select them such that they provide the largest possible information gain at each step. To this end, a decision tree is built based on the selected candidate objects and their stiffness features from the database. At each node, the value of a specific feature (a touch point) is acquired. Based on the obtained value, the next best exploration point is determined. This classification process allows us to determine, for instance, if a detected cup is made of a solid material or of plastics – and whether the geometry contains details such as reinforcement structures. Furthermore, a grasp planner is presented [8], which integrates stiffness information and object geometry to find feasible grasp points. The location of these grasp points may depend on subtle geometric details, which are unobservable by vision, as well as on the internal state of the object. A closed bottle, for instance, is stiffer than an open bottle, due to the counterpressure of the enclosed liquid or air. Also, liquids in an open bottle spill over if deformations become too large, as depicted in Fig. 1.3b. The presented grasp planner considers these effects using a deformation model.

1.4 Structure of this dissertation

This dissertation is structured as follows: In the following chapter, related work and relevant background are reviewed. The concept of visuo-haptic sensors based on plastic foams or thin

metallic rods ("beams") is introduced in Chapter 3, along with several implementations. In Chapter 4, the planning scheme for joint navigation and manipulation on mobile platforms is presented. Chapter 5 presents the transparency detector as well as the method for completion of geometry by haptic exploration of missing parts. The deformable object models for haptic object classification and grasping are presented in Chapter 6. The dissertation is concluded with Chapter 7.

Parts of this dissertation have been published in [4, 5, 6, 7, 8, 9].

2 Background and related work

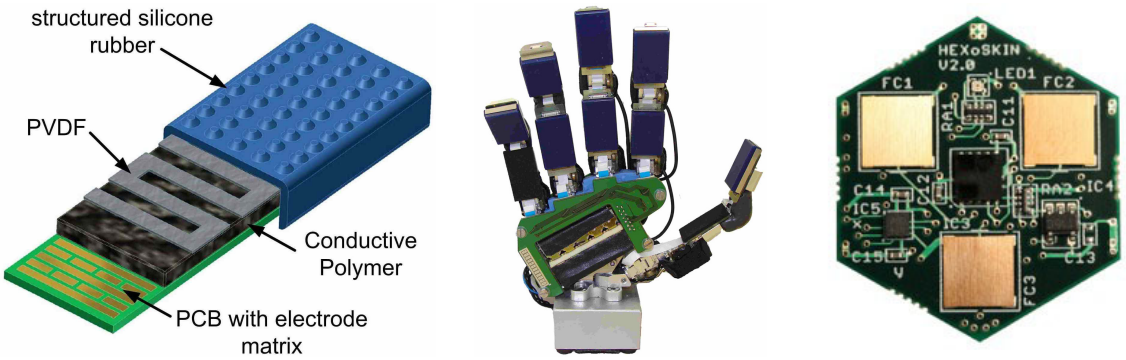
The following sections present an overview of sensor systems on autonomous robots as well as of the most important methods this work builds upon or relates to. This comprises methods and algorithms using haptics or vision for modeling, reconstruction, detection and tracking. For in-depth discussions of specific topics, the reader is referred to the given references.

2.1 Sensors for autonomous robots

Autonomous robots rely on a variety of different sensor systems in order to obtain a rich, multi-modal representation of the environment. For instance, force-torque sensors measure all degrees of freedom (3D force and 3D torque) at one point using at least 6 built-in discrete force sensors. They are a standard component of industrial robotics and offered as integrated modules with standardized mechanical mountings. Force-torque sensors are used in many scenarios and usually installed between a robot arm and a tool (the end-effector) in order to measure the contact forces/torques between the tool and the environment. Their readings are used in manipulation tasks like grasping, mounting, picking and placing of objects. Also, gravitational forces and acceleration forces while moving objects can be measured. For articulated manipulation tasks, such as opening a drawer, constraints imposed by the articulated object are deduced from the readings of force-torque sensors in order to control the force and motion of the robot arm accordingly. Finally, trajectory teaching requires the force-torque imposed to the tool by a human to let the robot arm follow a desired motion.

Touch sensors are haptic sensors, see below, that measure surface forces or contacts and are used on many parts of a robot. Simple variants use switches to detect contact with an object or pressure-sensitive resistive elements to obtain a single force value at a single point of contact. They belong to the kinesthetic or tactile perception system, depending on the extent and frequency of the measured forces. This work does not consider mid- or high-level frequency tactile signals, which appear for instance during vibrations, contact events or while sliding over a surface. Touch sensors are often mounted to extremal parts of the robot, such as the fingers, the feet or the base. It may even be desired to measure touch on the entire surface of a robot to allow for natural interaction with humans, or for perception in environments where objects may collide with the robot at any point.

Sensors are also used extensively within the robot in order to determine its system state. In a robot arm, each joint has its own (absolute) position or velocity encoder. High requirements on accuracy are imposed to these encoders, since they are essential to calculate the arm pose



(a) Sensor module **(b)** Robot hand **(c)** Artificial skin sensor

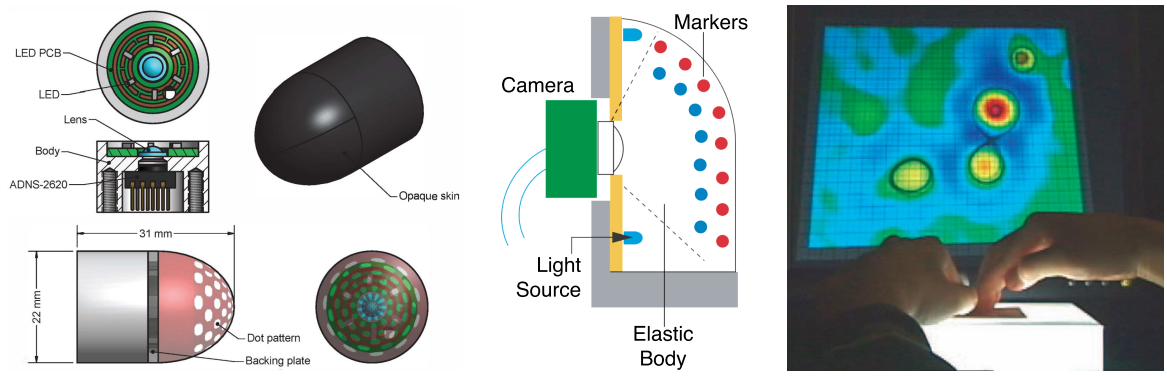
Figure 2.1: Laminar tactile sensors *(a)* provide 2D contact force profiles and are used for instance on advanced robotic hands *(b)*. Multimodal sensor modules *(c)* are mounted on the body of a robot and linked together to form an "artifical skin" with tactile sensing. Image sources: [35, 75].

using forward kinematics. Additionally, joints are often also equipped with torque sensors for teaching applications, improved control and safety reasons. In some cases they can replace a dedicated force sensor at the end-effector, as implemented on the KUKA LWR arm [14]. Similar sensors may be found in other actuators.

Modern robots are also equipped with active and passive visual sensors. Laser scanners provide accurate range data and are extensively used to build environment models, such as 2D maps for navigation [36] or 3D point clouds for motion and grasp planning. Cameras in the visual spectrum are commonly used by computer vision algorithms for tasks like object classification, visual search or pose estimation. Depth data may be acquired with stereo camera setups or active sensors, such as the Kinect or Xtion, see e.g. [53]. The advent of the latter, low-cost sensor has accelerated research on depth data and point cloud processing. Most vision algorithms can also be applied to images captured in the near-infrared range, which offers some interesting possibilities, such as active lighting invisible to humans.

2.1.1 Haptic and tactile sensors

Haptic perception mainly relies on various force sensors, which usually either measure a force (or force-torque) vector at a single point, or a distributed force/pressure over a surface. The latter type, see Fig. 2.1, is a laminar sensor array that measures a (usually scalar) force value over a 2D surface. The "tactile images" provided by such sensors are used for complex grasping tasks or tactile object classification. A review of tactile sensors is given in [101]. The sensor proposed in [35], see Fig. 2.1a, is based on a piezoelectric material and provides a 4×7 tactile image. The authors use this sensor on a humanoid human hand, see Fig. 2.1b, and propose methods for slip detection as well as contact shape classification. Similar devices are now commercially available. A sensor module designed to cover larger areas, which allows



(a) Fingertip sensor **(b)** GelForce sensor **(c)** GelForce input device

Figure 2.2: Tactile sensors with optical readout: An integrated camera-like sensor observes displacements of an elastic structure. Image sources: [48, 57].

humans to interact physically with the "body" of a robot, is presented in [75], see Fig. 2.1c. It provides multimodal sensing of temperature, proximity, normal force and acceleration. The touch sensor becomes a distributed areal sensor system, much like the human skin. Each module is rather complex, since it has its own signal processing system and connects to its neighbors, building a "peer-to-peer" communication network.

Force sensors that measure contact force profiles or vibrations are categorized as tactile sensors. On the other hand, grasping and manipulation forces as well as forces measured in arm joints are kinesthetic data. Even for a single sensor, the distinction is sometimes not clear, which is why we use the generic term "haptic sensor". Force sensors have been proposed based on different principles – such as optical readout, pressure-sensitive resistors or piezoelectric transducers [54, 96, 102].

A commonly used principle for force sensors is optical tracking of displacements in an elastic material. In [57], a sensor is proposed that is intended as a fingertip for a robotic hand, see Fig. 2.2a. It consists of an opaque, dome-shaped elastic material, to which forces are applied from the outside. Dots are drawn on the inside surface of this dome. These dots are used as visual features and observed by a specialized camera, which provides 18×18 pixels at up to 1500 Hz. The camera is located in the inside of the sensor, together with a light source. It is assumed that only frontal forces are applied, but the setup would also be able to measure shear forces. The authors measure the strain-stress relation for the dome-shaped structure and fit a model. An algorithm is presented for low latency detection of a contact event with a microcontroller. It uses a classification tree based on image statistics. A second algorithm is presented to calculate a force estimate based on optical flow, running at a much lower frame rate of about 5 Hz.

The "GelForce" sensor [48] is intended for applications in robotics and human machine interfaces. First, this sensor may be used as a soft fingertip sensor for robots (Fig. 2.2b).

Second, the authors suggest to use it as a natural haptic input device (Fig. 2.2c): A user applies 3D forces over a 2D surface, which may be used for virtual reality and entertainment applications. The ability to provide a traction field – i.e. a multidimensional force reading over a surface – is a special feature of this sensor, compared to most other systems. The sensor consists of a solid block of transparent silicon rubber, with colored markers (small spheres) embedded near the "sensitive" surface, see Fig. 2.2b. From the other side, a camera with VGA resolution looks into this transparent block and tracks the markers using their center of mass in the image. The tracker provides a displacement field of the material as a function over the surface. In order to obtain 3D displacement data, the authors put markers of different colors at two different depth levels, e.g. 3/8 mm. They exploit the fact that the amount of displacement reduces within the material with increasing distance from the surface. Like this, a direct depth measurement – which would require more complex sensors like stereo cameras – is not necessary. The traction field is calculated from the displacements measured in the image based on elasticity theory.

For both sensors, the camera, illumination, processing unit and the elastic material are an integral part of the device. As a system component, these devices are purely haptic sensors. The integrated camera only observes the elastic material itself. In contrast, the visuo-haptic sensor concept presented in this work benefits from a separation of the passive mechanic and the active components of the sensor. Since the camera is located "outside" the elastic material, it provides coherent, multimodal measurements from both the sensitive component and the environment.

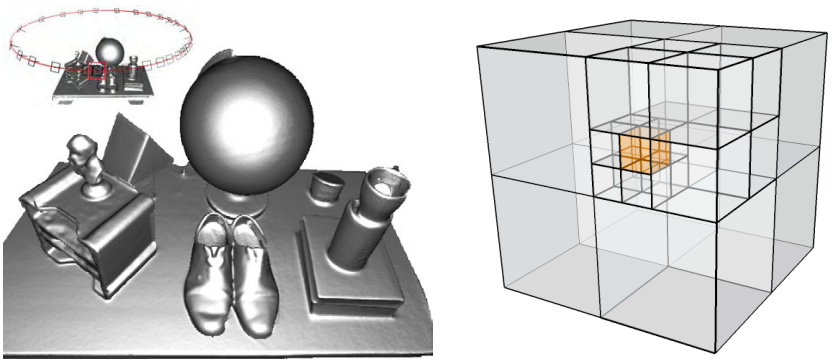
In [49], a sensor dubbed "vision-based active antenna" is proposed. It is based on a camera pointing onto a flexible beam to which a force is applied. This concept is similar to the beam-based sensor presented in Chapter 3. The authors track the deflected beam in the image by searching the brightest points along pixel columns. While this approach provides great accuracy and robustness to occlusions, it is limited to 2D force measurements in structured environments. The authors argue that the sensor can be used both during a detection and an approach phase. Yet, since the beam is only detected in front of a black or single-colored background, the two modes cannot be run simultaneously.

This subsection is partly taken from [6].

2.2 Object models and reconstruction

2.2.1 Visual reconstruction

Single view observations from (depth) cameras only show a part of the environment due to a limited viewing angle and due to occlusion, which is a fundamental property of projection. Much like the scene memory of the human brain, many processes require a scene representation that goes beyond the current observation. In order to build a complete geometric representation of a scene or an object – a 3D model – multiple observation from many differ-



(a) KinectFusion **(b)** Octree representation

Figure 2.3: KinectFusion *(a)* reconstructs small scenes in realtime. The estimated camera trajectory around the scene is shown in the top left. OctoMap allows for the reconstruction of large scenes, since it uses octrees *(b)* for an efficient subdivision of space. *(a)* Image source: [76].

ent viewpoints must be fused. State-of-the-art approaches integrate all available views that can be processed from a single, moving camera, without imposing any constraints on the camera trajectory. This requires simultaneous camera tracking and model building. Ideally, models contain every surface visible from any point in "free space" within a given space. This requires "complete coverage" and a corresponding viewpoint planner. Models should contain real physical dimensions and optionally color information.

In robotics, such environment or object models enable a great variety of applications: Physical motions must be planned such that collisions between the robot and the environment are avoided. Trajectories of robot arms are planned with numerous constraints, which must consider every moving segment [79]. Navigation of mobile platforms requires finding an efficient route around obstacles – a problem which is often simplified to planning in a 2D map, see [86]. Also, grasp planning relies on geometric models to find reliable grasp configurations [73]. Finally, semantic information – such as the identity or class of objects – is extracted from 3D models for higher level planning. Typical model scales range from individual objects or small scenes to entire buildings.

KinectFusion [76] reconstructs 3D models of small scenes based on a hand-held Kinect camera. It uses measurements from the depth sensor and optionally integrates color information. The system runs in realtime on powerful GPUs (graphics processing units) and works with arbitrary camera movements, as long as they are not too fast. All measurements from the Kinect are combined in a global volumetric model, which fuses the views from all perspectives. The global model is a dense, voxel-based representation called signed distance function (SDF), which shows the distance of each voxel from the closest surface. The fusion process significantly reduces sensor noise and incorporates data observed from different viewpoints. Yet, outliers or geometric inconsistencies result in modeling errors, which may lead to track-

ing problems. To speed up processing, the SDF is truncated, i.e. it is only evaluated close to surfaces. The KinectFusion method iterates over four major steps:

Measurement Extract vertices and normals from the depth map provided by the Kinect.

Pose Estimation The camera pose is estimated using a multi-scale ICP algorithm, which aligns the current observation to the surface estimated from the model (see below). Tracking against a global model avoids the accumulation of drift between frames.

Reconstruction update A local SDF is generated based on the current observation in order to update the global model. Integration of this SDF to the global SDF is performed by weighted averaging. The global model accumulates weights from all integrated measurements in a separate storage.

Surface prediction A dense surface representation is extracted from the global model using ray-casting for the current viewpoint. This surface is used for the next pose estimation step, resulting in a feedback loop between model estimation and pose estimation.

As long as there are no missing viewpoints, the volumetric model provides a complete 3D model of the scene, see Fig. 2.3a. Surface meshes can be extracted from the SDF using for instance the Marching Cubes algorithm [62]. The authors in [76] also present virtual reality applications, which allow for realistic physics simulations. For robotics, geometrically accurate object models can be built with KinectFusion.

Dense voxel grids require a lot of memory and are thus limited to small spaces. A volume of 1000^3 voxels requires already 1 GB of memory (using only one byte per voxel) – which must be available on the graphics card for GPU-based implementations. In [44], an approach called OctoMap is presented, which allows storing voxel-based scene representations of any scale. The authors present models of individual buildings and of an entire campus. There are many large regions in such spaces that are free/unoccupied and can be represented in a compact fashion. (Since depth sensors do not see inside objects, it is not required to model large regions of occupied space.) The approach is based on a hierarchical partitioning of space using an octree, which subdivides space into four cubes at each level, see Fig. 2.3b. Subdivision stops if a cube is completely unoccupied, which means that large unoccupied areas are represented by "supervoxels". Otherwise, subdivision is continued up to a certain maximal resolution. The corresponding tree can be stored in a very compact bitstream. As a side-effect, the octree-based representation also allows for efficient collision detection.

2.2.2 Transparency detection

The above-mentioned reconstruction methods assume so-called Lambertian surfaces, for which diffuse reflection dominates. Transparent materials exhibit almost no diffuse reflection, and require more complex image formation models. Specifically, the background behind a transparent object needs to be considered. A detector for transparent objects with curved surfaces is proposed in Chapter 5.

In [65], a detection method for transparent objects with the Kinect sensors is presented. It takes advantage of the fact that many transparent objects appear as holes in the depth map (no/invalid data) with this sensor. These holes are considered as candidates for transparent objects and serve as an initialization for a segmentation algorithm, based on contour extraction in the RGB image. Objects and their poses are detected in the image by comparing the region with pre-learned models. Transparent objects appear at least partly as holes in the depth image because they do not reflect enough light from the Kinect's IR projector or disturb the projection too much for successful pattern matching. However, there are also other causes for holes, such as occlusion boundaries, specularities, surfaces with a very low albedo, active light sources or depth values beyond the measurement range. These causes must be detected and rejected.

Other recent approaches for the detection of transparent objects work with transparency features that model the appearance as an additive combination of patches [30] or rely on the partial absorption in transparent material, measured from two viewpoints with an active sensor [55]. In [47], an overview of methods for the reconstruction of specular and transparent objects using various sensors and setups is given. The presented approaches are classified by the underlying image formation principle (such as diffuse reflection, refraction, scattering) and corresponding object types.

The presented approaches all address very specific problems concerning object type, sensor and setup. A general framework for the perception of objects with non-Lambertian reflection does not exist.

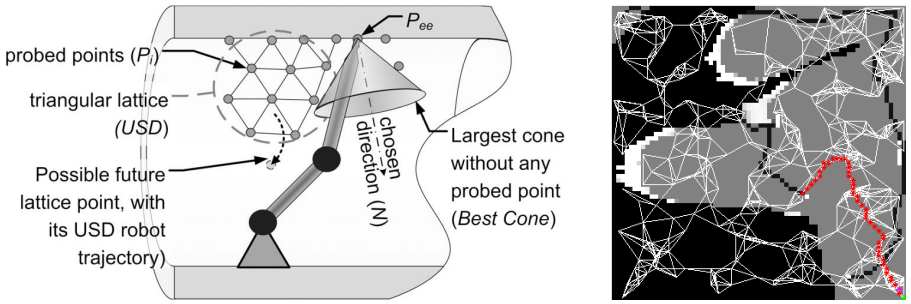
This subsection is partly taken from [4].

2.2.3 Tactile reconstruction

In tactile reconstruction, the geometry of an object or a scene is acquired by active physical contact, i.e. by touching objects on their surface with a robotic manipulator. This is a time-consuming process, since it may take a second or more to acquire a single point. Purely tactile exploration is therefore either performed with strong spatial constraints (such as on an object in the hand of a robot), or applied in harsh environments where other sensors fail. Since the capabilities of manipulators and tactile sensors vary greatly, reconstruction methods are optimized for specific setups – there are no "all-purpose" approaches.

Generally, tactile reconstruction may be separated into two processes: First, exploration planning determines the path of exploration, usually based on the available data. This step is essential, since haptic sensors only measure a single point or a small region of the object. A dense sampling grid – as provided by visual sensors – cannot be obtained in an acceptable time. Second, models are fitted to the measured data points. Often, geometric primitives are used for this process, but it is also possible to use more complex, generic surface models.

In [70], a method for tactile exploration of unknown surfaces is presented, and applied to the problem of mapping oil wells. A point probe sensor mounted on a robot arm provides



(a) Tactile mapping **(b)** Multi-modal exploration

Figure 2.4: *(a)* Tactile exploration and mapping of unknown geometries with a single contact probe. *(b)* Joint visuo-haptic exploration based on an occupancy grid map. White lines show trajectory candidates (probabilistic road map), and one selected trajectory for tactile exploration is drawn in red. Image sources: [15, 70].

the 3D location of a single contact point. It is argued that this approach is more practical and allows for more accurate fitting of geometric models in the presence of noise, compared to probing the surface by sliding along lines. The proposed exploration plan is a two-stage process: Multiple points are acquired within a local neighborhood, until a model can be fitted with sufficient certainty, see points P_i in Fig. 2.4a. Then, an exploration step is performed towards the direction of greatest geometric uncertainty. To this end, the largest cone whose tip is located at the current end-effector position and does not have any data points on its inside is determined. The axis of this cone is the next exploration direction, see Fig. 2.4a. The authors argue that this approach is not optimal, but fast and effective. Geometric primitives, such as spheres, cylinders and planes, are fitted to the explored points. The environment is modelled using multiple such primitives.

An approach for joint visuo-haptic reconstruction in 2D is presented in [15]. Unknown regions in scene observations from a visual sensor are haptically explored by a robotic hand using a probabilistic approach. The authors start with several views of the scene obtained by a stereo camera mounted on a robot's head. The depths maps are projected as 3D points into a common reference frame, segmented into objects and background and subsequently projected down to a 2D occupancy grid map. This grid map exhibits free, occupied and unknown cells in areas that were occluded or not seen by any camera view. The state of these unknown cells is predicted using a Gaussian process, which takes the spatial correlations into account. A likelihood map is obtained, which shows the occupancy probability and variance, and serves as the basis for exploration planning.

Tactile exploration planning amounts to planning a trajectory through the scene, which is expected to provide most information about scene geometry. Exhaustive exploration is infeasible, since it is too time-consuming. The authors use active learning to select trajectory points that maximize the information gain. The set of possible trajectories is reduced to

a manageable size by only considering paths in a so-called probabilistic road map, which is generated by sampling cells in the likelihood map and connecting them. Multiple utility functions are proposed, which consider the variance in the likelihood and the distance either for individual cells or for an entire path. Fig. 2.4b shows the occupancy grid map, the probabilistic road map, as well as one planned trajectory. Contact with an object is detected by a sensor matrix on the robot hand, taking into account a noise model of the sensor. For each measurement along the trajectory, the map is updated using a probabilistic model. Results are given for simulated and real scenes and compared to spanning tree coverage as a baseline method. The authors note that in the real world, objects are moved by tactile exploration and suggest to use visual tracking to determine pose changes.

Knowledge of the internal state (full, empty, open, closed) of a container object is important for various manipulation tasks. A method to acquire the internal state of deformable bottles is presented in [20]. The authors extract a 6-dimensional tactile feature vector, which comprises contact events, grasping duration, compression velocity and ratio. Features are extracted while grasping objects with a robotic hand with tactile sensor arrays. Additionally, the authors extract high-frequency features, which were found to be useful in an experiment with humans. Internal states are distinguished with a trained classifier – there is no explicit object model. The exploration of internal state is also considered in this work, see Sec. 6.4.2. The features proposed in [20] are similar to the "haptic tag" presented in Sec. 4.1.

2.2.4 Modeling of elastic materials

All materials deform to some extent when external pressure or forces are applied. Elastic deformations are non-permanent – the material returns to its initial shape as soon as external forces are removed. With increasing deformation, shape changes become plastic (i.e. irreversible), and at some point, most materials break (fracture). For robotic manipulation tasks considered in this work, it is undesired to change the shapes of objects permanently. Therefore, it must be ensured that deformations remain in the elastic range.

Homogeneous isotropic elastic materials are described by two fundamental material properties: First, Young's modulus E is a measure of stiffness and is defined as the ratio of stress to strain along the same, single axis. Stress is the force per unit area (pressure), and strain is the dimensionless ratio of deformation over the initial length. Typical values of E are given in Table 2.1. Second, the dimensionless Poisson's ratio ν describes the expansion of a material perpendicular to the compression axis. The two parameters are illustrated and measured with a simple experiment: A round rod of length l, diameter d and area of cross section A is loaded with a force f along its longitudinal axis, as in Fig. 2.6. This results in a change of length Δl as well as an expansion of the rod along its diameter Δd given by:

$$\Delta l = \frac{l}{EA} f, \quad \Delta d = -d\nu \frac{\Delta l}{l} \tag{2.1}$$

Material	E [GPa]
Rubber	0.01 – 0.1
Low density polyethylene (LDPE)	0.11 – 0.45
High density polyethylene (HDPE)	0.8
Polypropylene (PP)	1.5 – 2.0
Polyethylene terephthalate (PET)	2.0 – 2.7
Aluminium	69
Copper	117
Spring Steel (EN 10270-1 DH)	206

Table 2.1: Value of the Young's modulus E for various materials ($1\,\mathrm{GPa} = 10^9 \frac{\mathrm{N}}{\mathrm{m}^2}$) [99]

There are equivalent material parameters, such as the Lamé parameters λ, μ, which are used in models of 3D elasticity:

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)} \tag{2.2}$$

Elasticity in 3D may be modeled by various constitutive material models, such as the Saint Venant–Kirchhoff model for hyperelastic materials, or the Cauchy-elastic model, see [92]. First of all, a deformation map $\phi(\mathbf{X})$ is defined, which maps from each point in the reference (undeformed) state to the corresponding point in the deformed state, see Fig. 2.5. In the following, the *deformation gradient tensor* $\tilde{F}$ is used, which is the Jacobian of ϕ, i.e. $\tilde{F} = {}^{\partial\phi}/_{\partial\mathbf{x}}$. For instance, if an object is stretched linearly around the center, $\phi(\mathbf{X}) = \gamma\mathbf{X}$, and $\tilde{F} = \gamma I$. In order to express the "severity" of a given deformation, a strain measure is derived from $\tilde{F}$. A commonly used measure is the Green strain tensor $\tilde{E} \in R^{3\times3}$:

$$\tilde{E} = \frac{1}{2}\left(\tilde{F}^T\tilde{F} - I\right) \tag{2.3}$$

The Green strain is zero in the reference configuration ($\tilde{F} = I$), for rotations ($\tilde{F} = R$) and for translations of the entire object. In order to discard the rotational component of the deformation map, $\tilde{F}$ can be decomposed into a rotational and a non-rigid component. In that case, the Green strain is derived only from the second component.

Deformation results in the accumulation of potential energy in the deformed object. For hyperelastic materials, the potential energy $e(\phi)$ is a function of the deformation map only, which means that the path of compression is irrelevant, and that there is no loss of energy. Since the amount of deformation varies over the object, a local energy density Φ is introduced. The potential energy e is obtained by integration of Φ over the entire object. In the St. Venant-Kirchhoff material model Φ is expressed as follows:

$$\Phi = \mu \operatorname{tr} \tilde{E}^2 + \frac{\lambda}{2}\operatorname{tr}^2 \tilde{E} \tag{2.4}$$

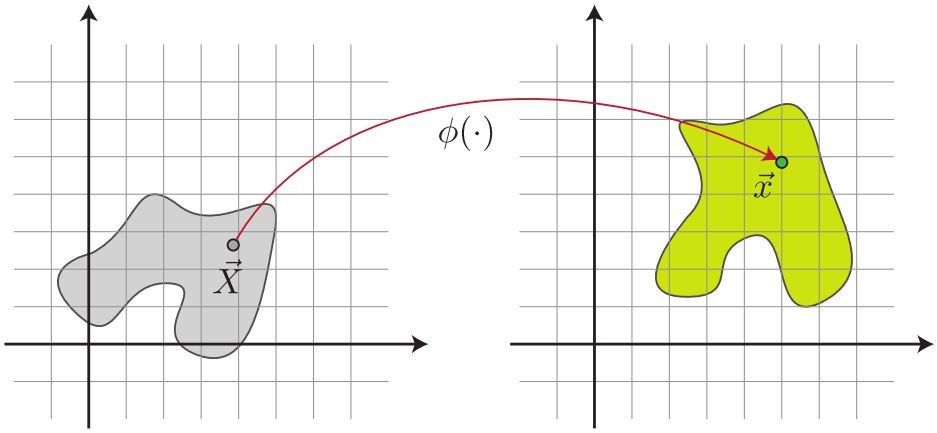


Figure 2.5: Deformation map ϕ from the reference configuration *(left)* to the deformed shape *(right)*. Image source: [92].

For numerical simulations, all measures are discretized, yielding a discrete potential energy $e(\mathbf{X})$ per node. Elastic forces at point $\mathbf{X}$ are obtained from the derivative of the energy:

$$\mathbf{f}(X) = -\frac{\partial e(\mathbf{X})}{\partial \mathbf{X}} \tag{2.5}$$

2.2.5 The finite element method

The Finite Element Method (FEM) is widely used in engineering and physics to calculate physical properties of complex objects. In this work, it is applied to numerically obtain the deformation of objects made of elastic materials. The principal idea of FEM is to partition complex objects into simple, standardized elements, for which the physical relation of interest can be expressed or approximated. A strain-stress relation is calculated for each element, the so-called *element stiffness matrix*. Volumetric objects are represented by a volumetric mesh, which, in the simplest case, consists of tetrahedron elements (the 3-simplex). Adequate elements and meshing are crucial to obtain physically correct solutions. Elements are connected and interact with each other only at a finite number of points, the nodes. The complete object is described by the *system stiffness matrix*, which is created by combining all element matrices, considering continuity and equilibrium conditions at the nodes. This matrix relates forces and displacements at all nodes. A solution of the system provides forces (stress) and deformations (displacements or strain) for all nodes, given external constraints, the boundary conditions. These are given for an arbitrary number of nodes and either specify the displacement or the external force.

A complete introduction to FEM goes beyond the scope of this text, so the interested reader is referred to the extensive literature in this field [92]. In order to illustrate the principal ideas, a simplified scenario of a 1D FEM is presented here, according to [24, Ch. 15]. Consider a

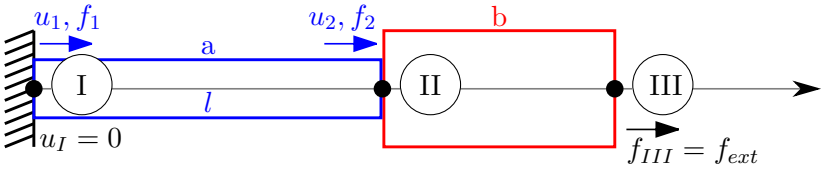


Figure 2.6: Simplified example for a 1D FEM assembled from rod elements. Node I is fixed, and an external force f_{ext} is applied to node III.

structure consisting of round rod elements of constant cross section. The object is built from several of these elements. The problem is considered in 1D only, i.e. forces/deformations are calculated and applied only along the longitudinal direction. Elements are aligned along this single axis, as depicted in Fig. 2.6.

First, consider a single element e, colored blue in the figure. The element stiffness matrix K_e relates displacements $u_{1,2}$ and forces $f_{1,2}$ at the two nodes of a single element. Dynamic effects as well as the self-weight are ignored here. For the rod element, the force-displacement relation can be given directly from Eqn. (2.1):

$$\mathbf{f_e} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} \frac{EA}{l} & -\frac{EA}{l} \\ -\frac{EA}{l} & \frac{EA}{l} \end{bmatrix}_e \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = K_e \mathbf{u_e} \tag{2.6}$$

Entries in this matrix are equivalent to spring stiffness constants. The geometry of the element e is represented by the length l and area of the cross section A. The only relevant material property (see Sec. 2.2.4) is E, since ν represents lateral deformations perpendicular to the considered axis. In the 3D case, the calculation of element stiffness matrices is more complex. It is based on the 3D elasticity models discussed in Sec. 2.2.4, in particular a discretized version of the deformation energy, Eqn. (2.4).

The system stiffness matrix is assembled from all element matrices, considering the connection of elements at the nodes. Element nodes $(1, 2)$ are mapped to system nodes (I, II, III). At nodes connected to multiple elements, element displacements are identical, e.g. $u_{2,a} = u_{1,b} = u_{II}$, and forces are superimposed. The system matrix K is assembled based on this principle and relates all node displacements and forces. The element matrices from the blue and red elements are indicated:

$$\mathbf{f} = \begin{bmatrix} f_I \\ f_{II} \\ f_{III} \end{bmatrix} = \begin{bmatrix} \left(\frac{EA}{l}\right)_a & -\left(\frac{EA}{l}\right)_a & 0 \\ -\left(\frac{EA}{l}\right)_a & \left(\frac{EA}{l}\right)_a + \left(\frac{EA}{l}\right)_b & -\left(\frac{EA}{l}\right)_b \\ 0 & -\left(\frac{EA}{l}\right)_b & \left(\frac{EA}{l}\right)_b \end{bmatrix} \begin{bmatrix} u_I \\ u_{II} \\ u_{III} \end{bmatrix} = K\mathbf{u} \tag{2.7}$$

For large systems, K is sparse. In order to solve the system, boundary conditions must be incorporated. To this end, the rows and columns that correspond to fixed displacements

(here, $u_I = 0$) are removed from K, and the reduced system is solved. Finally, the removed forces (here, f_I) are calculated.

2.3 Detection and tracking

2.3.1 Images features

Local image features are salient locations in an image, which optionally provide a descriptor or "fingerprint" of image patches. Good features are descriptive, e.g. specific to a given texture, but still tolerant to changes of perspective and lighting conditions. One of the most popular methods for feature detection and description is the Scale-invariant Feature Transform (SIFT) [63], but numerous other feature types exist. The search for equivalent descriptors between two images is called feature matching. Matching is usually performed between a current image, such as a live camera view, and either a model image, a (potentially large) feature database, an older image or an image taken from another perspective. Successfully matched features enable pose estimation and object recognition, see Sec. 2.3.5.

Feature detection In a first stage, salient locations robust to changes in perspective and lighting are identified in the image. This means that a feature should be detected at the same physical location in two pictures that are taken under different conditions. The survey in [72] investigates various detectors with respect to this property. One important measure is the *repeatability*, which quantifies how many feature locations are recovered in two views of the same scene. Most feature detectors search for corners (e.g. [40]), blobs or regions.

The SIFT method searches for blobs (dark or bright regions) by convolution of the image with a difference of Gaussians (DoG). The DoG is an approximation of the Laplacian operator, which exhibits a high repeatability [72]. The process is robust to image noise and invariant to global illumination changes. In order to obtain scale-invariance, detection is performed at different scales by filtering the image with Gaussians of different standard deviations:

$$\sigma_{n+1} = k\sigma_n = k^{n+1}\sigma_0 \tag{2.8}$$

The parameter k determines the resolution of the scale space. Scale-invariance is essential to cope with varying image resolution, camera zoom and distance to the physical structure. In practice, a *scale space* representation is created from the input image by repetitive convolution with a Gaussian. At each scale level, the image is increasingly blurred according to Eqn. (2.8). Once the image resolution is reduced by half, a new octave is generated by subsampling the image by a factor of two. This avoids increasing complexity of the convolution, since the maximal filter size (which depends on σ) remains fixed. The DoG operator

$$\mathrm{DoG} = G_{\sigma_{n+1}} - G_{\sigma_n}, \quad \text{with } G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{2.9}$$

calculates the difference between two adjacent levels in scale space, which allows for efficient computation. Features are local extrema of the 3D filter output and are represented as a triplet of image location and scale (x, y, σ). The number of detected features obviously depends on the image contents. For typical photos, several hundred features are extracted.

Feature description In a second stage, a feature descriptor, which is a characteristic vector of the texture in a local patch, is calculated. Most approaches calculate descriptors only at detected feature locations, which results in a sparse representation of the image. This means that the invariance and repeatability of the detector directly influences the quality of descriptors. Note, however, that there are also descriptors that are extracted in a dense fashion from the image or the scale space, such as histogram of oriented gradients (HoG) [22]. This approach is very time-consuming, but it ensures that no image region is "missed".

The SIFT descriptor is computed within a square patch around the feature location with a scale-dependant side length of 12σ. The patch is divided into 4×4 subblocks and rotated, such that it is aligned with the dominant gradient orientation at the feature. For each subblock, an 8-bin histogram of weighted gradients is generated, where each bin corresponds to a range of gradient orientations. The bins are concatenated to a $4 \times 4 \times 8 = 128$ dimensional vector, which is finally normalized to account for intensity changes. This fingerprint describes the texture around the feature location and is invariant to intensity changes, scale changes and in-plane rotations. Small changes in the texture are tolerated, since the gradients in the subblocks provide some degree of abstraction.

Descriptors are matched by nearest neighbor search using the Euclidean distance in the 128-dimensional feature space. The ratio of the distance from the closest neighbor to the second closest one is taken as a measure of match quality. Only ratios greater than 0.8 are accepted as valid matches. Still, incorrect matches are possible and must be tolerated by subsequent processing steps. Nearest neighbor search is an exhaustive search, which is too inefficient if a descriptor needs to be compared to many candidates. Instead, the search space can be discretized using a hierarchic, tree-based approach [77]. Discretized features are referred to as "visual words" and can be processed by algorithms similar to those used for search in text documents.

2.3.2 Visual tracking

Template tracking Image registration or template tracking is used to align an image I to a reference template I_R, whereby the image is a warped version of the template. Usually, the image is a view of the planar template from another perspective, such that the warp is a homography. The homography represents a 6D transformation up to a scale factor. Tracking is a local method, which means that I and I_R must already be close to each other. This is usually ensured if a template is tracked in a video sequence, or if an estimate of its pose is

available from some other method. Popular tracking methods are the family of Lucas-Kanade algorithms [11] and efficient second-order minimization (ESM) [67].

The goal of these template trackers is to minimize the sum of squared differences between the current image I and the reference image I_R:

$$\mathbf{q} = \text{argmin}_{\mathbf{q}} \sum_{\mathbf{u}} \left[I(w_{\mathbf{q}}(\mathbf{u})) - I_R(\mathbf{u}) \right]^2 \tag{2.10}$$

This sum is calculated over all pixels $\mathbf{u}$ in I_R. The current image I is warped by w, which is a homography (or a simpler image transformation) modeled by the parameters $\mathbf{q}$. Minimization is performed with respect to these parameters. They are updated incrementally based on a current estimate in an iterative fashion: $\mathbf{q} \leftarrow \mathbf{q}' + \Delta\mathbf{q}$. For the first iteration, $\mathbf{q}'$ is an estimate from a previous frame or from another method. The expression in Eqn. (2.10) is approximated by a Taylor series of second order for ESM, or first order for the Lucas-Kanade algorithm:

$$\mathbf{\Delta q} = \text{argmin}_{\mathbf{\Delta q}} \sum_{\mathbf{u}} \left[I(w_{\mathbf{q}'}(\mathbf{u})) + \nabla I \frac{\partial w}{\partial \mathbf{q}'} \Delta\mathbf{q} - I_R(\mathbf{u}) \right]^2 \tag{2.11}$$

The image gradient ∇I is given at $\mathbf{u}$ in horizontal and vertical direction, and $\frac{\partial w}{\partial \mathbf{q}'}$ is the Jacobian of the warp at $\mathbf{u}$. Minimization of Eqn. (2.11) is a least squares problem, which is solved by setting its partial derivative to zero:

$$\mathbf{\Delta q} = \hat{H}^{-1} \sum_{\mathbf{u}} \left[\nabla I \frac{\partial w}{\partial \mathbf{q}'} \right]^T \left[I_R(\mathbf{u}) - I(w_{\mathbf{q}'}(\mathbf{u})) \right], \quad \hat{H} = \sum_{\mathbf{u}} \left[\nabla I \frac{\partial w}{\partial \mathbf{q}'} \right]^T \left[\nabla I \frac{\partial w}{\partial \mathbf{q}'} \right] \tag{2.12}$$

Where $\hat{H}$ is the Gauss-Newton approximation of the Hessian matrix. Images $\nabla I \frac{\partial w}{\partial \mathbf{q}'}$ are referred to as steepest descent images. The expression in Eqn. (2.12) is solved iteratively, until the parameter update $\mathbf{\Delta q}$ drops below a certain threshold. The resulting $\mathbf{q}$ parametrizes the homography between I and I_R, which can be decomposed into a 6D pose change, see Sec. 2.3.3. Finally, a convergence test can be performed based on the normalized cross-correlation coefficient (NCC) between the template and the warped image. For an illustration of the process including intermediate images, see [11, Fig. 2].

Template tracking is a direct method, i.e. it uses the pixels of an image directly, without an intermediate feature extraction step. The accuracy of pose estimation with tracking is higher than with features, since each pixel is considered for registration, allowing even for sub-pixel accuracy. The search range, however, is limited to a local region (which, however, can be extended by a scale space). Even though tracking works with dense data, it is faster than feature based approaches in many scenarios. Moreover, there are approaches to select a subset of templates or pixels therein that are "good" by an a posteriori measure [89, 12]. This is in contrast to feature descriptors, which use an a priori assumption about "good" data.

The limitation of tracking to planar templates is impractical in many cases. Therefore,

extensions of the presented concept for arbitrary geometries have been proposed. The tracker in [10] tracks/detects the camera pose based on a model, which comprises a set of reference views and is learned online or offline. In addition to the grayscale image, a depth map is integrated, which is sometimes referred to as a "2.5D model". Reference views are selected automatically from the scene. The initial pose is obtained by an exhaustive, yet fast search within the model. The method is robust to lighting variations and small scene changes. It operates in realtime, due to the selection of a subset of informative pixels.

Contour tracking Edges in an image are important features, since they often correspond to contours of objects, i.e. boundaries between objects. In that case, the contour and everything inside move consistently, as long as the object is not occluded. Contours are very characteristic for many objects and are therefore used for tracking and classification. For instance, human faces are only weakly textured and facial features are thus best described by internal and external contours. Other objects – such as printed packages – exhibit a strong texture, which is however much less descriptive than the objects' contours. In this work, contour tracking is used to determine the shape of deformable elements, see Sec. 3.2. Locally, edges are much less informative than texture and only provide a 1D constraint. Therefore, edge trackers usually rely on a global or semi-global edge description of objects.

Snakes [51] are one popular method of this type, which uses a model of points connected to a polyline. Points may move in the image to minimize an energy term, which consists of multiple components: First, there is an image part, which attracts snake points to edge features. A large spectrum of methods for edge detection exists – from simple high pass filters like the Sobel operator [23] to complex learning-based approaches, which try to detect natural boundaries [69]. Second, an internal energy term ensures smoothness of the snake, and third, a term for external constraints limits the motion of the points and allows for initialization. The energy term is minimized by local search, enabling tracking of edges as they move. If some edge points are very weak, the smoothness term ensures that the corresponding points are "dragged" by their neighbors. Initialization is done manually or by another system, which has some a priori knowledge of the estimated object position. Many subsequent methods build on the snakes method, such as the active shape models [21], which learn global shape deformations and include them in the energy term for more reliable tracking.

This paragraph is partly taken from [6].

2.3.3 Homography estimation and decomposition

In computer vision, a homography is an invertible mapping between two images of the same real-world plane. Images are projections according to the pinhole camera model with the 3×3 intrinsic camera matrix K. Equivalent points in two images are related by the expression

$$\tilde{\mathbf{u}}' = c \cdot [u, v, 1]^T \propto H\tilde{\mathbf{u}}, \tag{2.13}$$

with the 3×3 homography matrix H. Here, image coordinates $\tilde{\mathbf{u}}$ are expressed in homogeneous coordinates, i.e. $\tilde{\mathbf{u}} = c \cdot [x, y, 1]^T$, with pixel coordinates $[x, y]$ and an arbitrary scale factor $c \neq 0$. For an introduction about camera models projections and homogeneous coordinates, the reader is referred to [42]. Since scale is irrelevant for the homography matrix, it exhibits eight degrees of freedom (DoF).

The homography may be composed from a given rigid 3D body transformation $\mathcal{T}_{(R,\mathbf{T})}$:

$$\tilde{H} = R - \frac{1}{d}\mathbf{T}\mathbf{n}^T \tag{2.14}$$

Where R is a 3×3 rotation matrix, $\mathbf{T}$ is a 3×1 translation vector, $\mathbf{n}$ is the normal vector of the plane, and d is its distance to the camera. The obtained homography $\tilde{H}$ is based on a projection to the unit plane – it is, so to say, expressed in world units. It may be converted to a homography in the image space expressed in pixel units, as in Eqn. (2.13), using the camera matrix according to $H = K\tilde{H}K^{-1}$.

A homography can be calculated from 4 or more non-collinear 2D point equivalences $\tilde{\mathbf{u}}'_i \leftrightarrow \tilde{\mathbf{u}}_i$ using the direct linear transform (DLT). This process is referred to as *homography estimation*. For an overview of other methods for pose estimation from point correspondences, see [39]. Equivalences are for instance provided by feature matching (see Sec. 2.3.1) or a point/template tracker (if it does not directly provide a homography). For each point, two equations are obtained from Eqn. (2.13) by division of both the first and second row by the third row. Rearranging and stacking the equations for each point equivalence yields the following matrix equation:

$$A\mathbf{h} = \begin{bmatrix} -x_0 & -y_0 & -1 & 0 & 0 & 0 & u_0x_0 & u_0y_0 & u_0 \\ 0 & 0 & 0 & -x_0 & -y_0 & -1 & v_0x_0 & v_0y_0 & v_0 \\ -x_1 & -y_1 & -1 & 0 & 0 & 0 & u_1x_1 & u_1y_1 & u_1 \\ & & & & \cdots & & & & \end{bmatrix} \cdot H^T[:] = 0 \tag{2.15}$$

The unknown homography matrix is rearranged into a vector $\mathbf{h}$ row by row. Matrix A is known from the coordinates of the n point equivalences and has $2n$ rows. The null space of A is the solution of $\mathbf{h}$ for $n = 4$. For overdetermined cases ($n > 4$), $\mathbf{h}$ is obtained by singular value decomposition and corresponds to the right-singular vector of the smallest singular value. To obtain better results in the presence of noise, [42] proposes a normalization scheme.

Point correspondences generally contain outliers from incorrect feature matches. These errors do not just cancel out – the estimated homography will be significantly wrong. In order to identify and exclude outliers, the random sample consensus (RANSAC) algorithm is used [28]. This approach creates subsets of point correspondences by random sampling. Each subset has a fixed size of e.g. four correspondences. For each subset, a homography (the model) is fitted, i.e. Eqn. (2.15) is solved with the corresponding subset of row pairs. Next, it is verified how many of the remaining point correspondences agree to this estimated homography. If the homography has been estimated with an outlier, only very few correspondences will agree.

Otherwise, most of the correspondences will agree to the estimated homography within a certain threshold. A refinement may be performed based on all agreeing correspondences (the "consensus set"). The optimal parameters of RANSAC (size of subset, number of subsets, tolerances) depend on estimates of noise and ratio of outliers.

The inverse process of Eqn. (2.14), i.e. the extraction of a rigid body transformation (pose) from a given homography, is referred to as *homography decomposition*. This step is commonly performed after a homography has been estimated from point correspondences in order to obtain camera or object motion. As discussed, the homography exhibits eight DoF, but there are nine variables on the right-hand side of Eqn. (2.14). The scale d of the translation in Eqn. (2.14) cannot be determined, which corresponds to the loss of depth information during projection. However, for image to model matching, d is usually known from the model, and a full 6D pose can be given. This result is remarkable for robotics – it means that full transformations can be obtained from a 2D camera as long as the scale of the model is known. For a detailed discussion of homography decomposition, the reader is referred to [27, 66]. Based again on a singular value decomposition, four solutions are found for $(R, \mathbf{T})$. The physically correct one is determined based on a positive depth constraint.

2.3.4 Combined detector and tracker

A combined tracker/detector based on SIFT (Sec. 2.3.1) and the ESM tracker (Sec. 2.3.2) is used throughout this work in order to detect objects and obtain their exact 6D pose in realtime. The system detects/tracks multiple arbitrary planar templates, such as printed designs or natural photos, simultaneously. A method to select "good" templates for tracking based on a fast quality estimator has been proposed in [1]. Each template model is initialized on startup, i.e. SIFT features are extracted, and the tracker performs pre-calculations. Many artificial objects can be tracked directly using their printed texture as a model. Also, it is straight-forward to create real objects from template models at the exact scale with a standard printer. These templates are easily attached anywhere on a robot or an object. They offer more flexibility, ease of use and a higher tracking quality than 3D models or the predefined markers from ARtoolkit [52]. Since the real-world size of the template is known from the resolution (dpi value) stored in the image file, a full 6D pose can be obtained. The tracker is implemented as a Robot Operating System (ROS) node, allowing for communication with any other module and with various cameras. In most cases, a high quality USB webcam with Full HD resolution is used.

The system relies on a GPU implementation of SIFT to search for the template models in the camera image. This step is too slow for processing at a frame rate of 30 Hz on mid-range GPUs. Tracking is performed with ESM, which is either initialized from SIFT or from the respective pose in the previous frame. The latter case is preferred, since it allows skipping the slow SIFT detection step. Poses provided by ESM are more accurate and less noisy than from SIFT matches. Tracking is always done with the "original" model, such that there is

no drift, even if performed over long periods of time. Processing is split in multiple threads, running at different rates. Thread A performs SIFT detection and matching, but only if there is currently any template whose pose is unknown. Matching provides an estimate of the homography. The ESM runs in thread B at frame rate and tracks all templates, for which initial poses are available. If required, each tracker may run in a separate thread to reduce latency on multi-core CPUs. The real-world pose of the template with respect to the camera is obtained by homography decomposition, given the camera parameters.

2.3.5 Visual classification and recognition

There are countless approaches for visual recognition, optimized for different kinds of objects and relying on various features. Despite the extensive research in this field, an integrated method, which works for all kinds of objects does not exist. Rather, a suitable detector for the relevant types of objects must be chosen. One class of widely used approaches is based on bag of words or bag of features [77] with features such as SIFT. Objects are represented by a set of local features or discretized features, which enables accurate recognition of textured objects in very large databases. Discretized features (words) make it possible to apply efficient search trees, which enables search in databases with millions of objects within less than a second. Such a database can contain representations of all mass-produced objects the robot might encounter. Images of such objects can be retrieved from online shops or other public databases [88]. Good recognition results are achieved for artificial objects with locally planar textured surfaces – such as many household objects. However, this representation does not take shape and contour into account. Local texture features are unstable and unreliable at or near edges. Also, geometric relations between features are typically not considered.

Contours are bad locations for local texture features, but they do provide descriptive information for many objects, since they may correspond to their geometric shape. Histograms of oriented gradients (HOG) [22] mainly encode edge information and have been used as a feature for recognition and classification. The approach is time-consuming, since the HOG descriptor is extracted in a dense fashion from the image in scale space. It is however possible to reduce the search space a posteriori. One such approach based on a novelty detector is presented in [2]. In a 2D image, outer and inner contours cannot be distinguished from other edges in the texture, such that the relevant edges/counters must be learned. Combined with SVM-based learning, HOG has shown to be very effective for class-level detection of objects. Even objects with large intraclass shape variations can be detected reliably.

As geometry or shape is highly relevant for manipulation, classification based on geometry or shape features is an obvious approach. Depth information must be acquired with an active depth sensor or a stereo camera system. The latter solution provides correct depth values only in textured areas, which is a limitation for many objects. Typically, a shape database of complete 3D models is built offline. Models are matched to a partial 3D observation (or a depth map) from a single or a low number of viewpoints. Again, collections of 3D models on

the web may be utilized to create this database automatically. Matching may be performed by local 3D features extracted from the models and the depth observation. In [18], local surface features are calculated at descriptive locations and mapped to a discrete space using a hash table. Corresponding candidate objects are found using a voting scheme. Finally, a verification is performed based on iterative closest points (ICP) [84], which aligns point clouds from the database with the observation. Additionally, ICP provides an accurate pose estimate of the object. As for the above method, approaches based on local features allow for scaling when larger databases are used. For simple shapes, however, local features are not descriptive enough to identify an object. Instead, a global shape feature may be used [95], or the point clouds can be matched directly using ICP, which is a time-consuming process.

2.3.6 Mapping and navigation

Visual SLAM systems based on laser scanners [86] are commonly used on robots for self-localization within the scene and for simultaneously building/updating a 2D map of the environment. Most approaches use a probabilistic occupancy model for the grid cells, which is continuously updated when new measurements are available. In this way, an entire floor of a building can be mapped by driving a robot platform around it, even though the coverage of a single sensor measurement is much smaller. The Karto SLAM system [50] is one such approach. A method for loop closure [59] should be used in larger scenes, as maps would become inconsistent otherwise, due to the accumulation of small pose errors. Loops are detected based on local scan matching and subsequent refinement of an internal pose graph. To reduce sensor costs, the laser scanner can be replaced with a scan from a single line of the Kinect depth sensor [78].

Maps can be directly used to perform navigation tasks, i.e. to find a collision-free path between two points. The state of each cell is set to occupied/free/unknown according to the highest probability. Costs for free cells are often assigned as discussed in [58], based on a distance map $\mathcal{M}_d$, which shows the proximity of the closest obstacle. A platform with a circular footprint[1] of radius r_{rob} may not come closer to an occupied cell than r_{rob} – any cell that is closer to an obstacle is thus assigned infinite costs. In proximity to an obstacle – e.g. for $\mathcal{M}_d \in [r_{rob}, 3r_{rob}]$ – the platform must drive slower and may require correction movements. Therefore, in these regions costs are (e.g. linearly) decreasing with increasing $\mathcal{M}_d$. Any point $\mathcal{M}_d > 3r_{rob}$ has constant costs, based on the maximum platform speed. For navigation, each grid cell is represented as a node in a graph and connected to its neighbors with the above-calculated weights. Standard path planning algorithms such as $A^\star$ [41] may then be applied to this graph. Alternatively, fast-marching approaches [32, 58] may be used – they create a potential field with a single minimum at a predefined goal position. While the generation of this field is time-intensive in larger maps, this representation enables to quickly find the shortest path from any point in the map to the predefined goal.

[1] For non-circular footprints, see [68]

Direct navigation on maps is used for local planning, but unfeasible for large scenes. Global planning requires more abstract representations. For instance, [25, 94] use a skeleton for navigation. Skeletons represent the thinned structure of objects and can be created from a binary map $m := (\mathcal{M}_d < r_{rob})$ using the morphological "thinning" operation [90]. The skeleton exhibits a single path in the middle of each corridor or through each doorway, as well as intersections, which are placed into central areas. Furthermore, paths span into corners (stubs) and circle free-standing obstacles, showing that there are alternative ways to drive around them. The conversion of a skeleton to a topological graph is straight-forward, by placing a node at each pixel and connecting it with its direct neighbors. Edge weights of this graph are taken from the cost map discussed above.

Navigation is much faster on such a simplified graph, yet it is not possible to reach the exact goal position. Modern path planning systems are thus often split into two parts [68], using a global planner, which works on an abstracted representation of the scene and a local planner, which utilizes a detailed local map and controls the platform in realtime. Abstract representations of scenes have also been created using other approaches – for instance, [71] proposes a method based on map segmentation inspired from image processing. The approach presented in [103] combines depth data and laser scans to recognize scenes, thus representing the environment at a semantic level.

This subsection is partly taken from [6].

3 Visuo-haptic sensors

Robotic manipulation is based on both visual and haptic/tactile perception. Typically, there are two separate sensor systems for the two modalities, which results in increased costs and complexity. In this chapter, the concept of a visuo-haptic sensor is presented, which provides haptic data from an external standard camera. To this end, a deformable element with known characteristics – such as a metallic rod or piece of plastic/rubber foam – is mounted onto a robotic manipulator and observed by the camera. Force, pressure and impression are calculated from the visual observation, using known stiffness parameters of the deformable element. The mechanical part is completely passive, low-cost and offers passive compliance. The camera can be shared with other systems and for instance perform visual tracking or detection tasks near or at the manipulator. Thereby, multimodal and coherent measurements are obtained using only a single sensor.

This chapter is partly taken from [5, 6, 7].

3.1 Motivation

As outlined in Sec. 2.1, current intelligent robots rely on many sensor modules for different modalities located all around the robot. There are a number of problems with such a highly complex system of diverse and distributed sensors. First of all, distributed components go along with a high effort during development, production, programming and maintenance of the robot. Each sensor unit has its own CPU, firmware, hardware revision and interface, which must be kept compatible during software updates or the exchange of hardware components – even across different revisions. Interfaces can be simplified by using standardized protocols based on UDP or TCP/IP – yet the implementation of a network stack drives up power consumption and software complexity of the sensor modules. Additionally, complex and costly wiring is required, especially for areal skin-like sensors and for components in the arm. The installation of a wiring harness for distributed components is a costly process even in existing production processes, despite the extensive use of network or bus systems. Robots pose additional challenges for wiring: Space for cables is especially scarce, placing them through tight holes is time-consuming, and there are high requirements on mechanical stability of wires in moving parts – such as joints that rotate up to 360°. All these factors drive up the infrastructure costs for a sensor module.

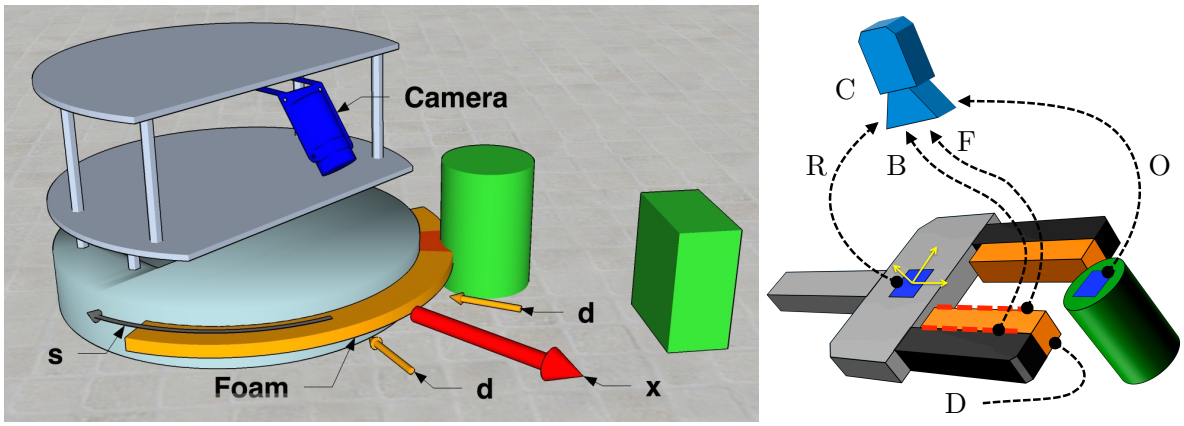
From the perspective of signal processing, coherence and synchronization in spatial and time domain is required between the sensors. This is a precondition both for multimodal data

fusion as well as for a smooth handover between sensors. To ensure accurate synchronization, mutual calibration between sensors is required – which becomes increasingly complex as more and more sensors are involved. Special hardware designs, such as synchronized system clocks, may be necessary. In case of grasping or manipulation of objects, positioning relies on a first stage based on (remote) visual tracking, followed by a refinement stage based on tactile sensors. A handover point between the two sensor systems must be determined, and there should be no jumps in the pose estimation of the object. Otherwise, the grasp may be unstable, or the object might get hit and fall.

Cameras play a distinguished role in the design of future sensor systems for intelligent robots – both concerning their capabilities and system complexity. They are the most versatile sensor type, provide high resolutions and dense sampling, exhibit high quality measurements at very low costs, and they are already involved in many robotics tasks. Therefore, efficient and versatile sensor systems should rely on cameras were possible. Versatility comes from the fact that computer vision algorithms can extract diverse data from images – potentially even using only a single sensor. There are algorithms providing the poses, geometries, identities, types or classes of objects and obstacles. Furthermore, tracking algorithms are important to identify the motion of objects or humans. Specialized methods exist for detection and tracking of humans or faces as well as for mapping and navigation, see Sec. 2.3.6. Based on such algorithms, cameras have the potential to replace many other sensors – with important implications on the complexity of sensor systems. Commercial systems based on this concept are now used in the automotive industry [87], replacing radar and other specialized sensors with a simple camera. Driven by consumer applications in digital cameras and smartphones, cameras are available at very low costs and in small sizes. Their technical specifications are constantly improving, providing excellent image quality (i.e. high resolution, frame rate, dynamic range and low noise level). Current consumer models even provide considerably higher frame rates (up to 1000 Hz, 240 Hz on many smartphone), which goes along with reduced processing delays. Even though lighting may limit the use of such extreme frame rates, such rates are comparable to those of many dedicated tactile or force sensors.

Both the visual and haptic modalities have their specific shortcomings: Visual methods, for instance, fail for transparent or specular objects. Also, vision does not provide information about weight or deformability of an object directly. Haptic sensors do not allow for remote sensing, provide only sparse information about an object and require time-intensive exploration steps. Therefore, many robotic tasks rely on dedicated tactile sensors and on cameras. For example, grasping or manipulation tasks rely on both visual and haptic sensors, based on the proximity of an object. Cameras determine the identity and rough pose of an object, while contact events, forces and potentially a refined pose are obtained from a tactile sensor.

It is therefore feasible to combine these two types of sensors into a single visuo-haptic system. Cameras allow for remote sensing and are essential for environment mapping, even on simple platforms. With the proposed sensor design, tactile data and haptic events are acquired using the same cameras. To this end, a deformable element with known material



(a) Mobile platform **(b)** Robotic gripper

Figure 3.1: *(a)* Foam-based visuo-haptic sensor on a mobile platform for haptic exploration. The passive deformable element, a plastic or rubber foam rod (orange), is observed by a camera (blue). It is pushed against the green object, resulting in a deformation of the foam. The exploration direction is $\mathbf{x}$, the foam deforms along $\mathbf{d}(s)$. *(b)* The same concept is applied to the two fingers of a robot gripper. *(a)* Figure adapted from [6].

characteristics is attached to the robotic actuator, see Fig. 3.1. Its deformation is determined visually with high precision using an edge-based tracker, and acting forces or pressure can be derived since the material characteristics are known. For low-end robots this approach provides visual and haptic information from one integrated system, which reduces costs by removing additional haptic sensors. More complex systems, on the other hand, profit from more accurate models of the environment obtained by fusion of visual and haptic data.

3.2 Modeling of the deformable element

Plastic foams The proposed foam-based visuo-haptic sensor uses a piece of deformable plastic or rubber foam. It is shaped as a rod or bar and represents the deformable element. Deformation models and properties of such foams have been studied intensively, see e.g. [33]. The vision system relies on these models to calculate pressure profiles from the observed deformation. The strain-stress relation for these materials is non-linear, typically showing three regions: A region of approximately *linear elasticity* for very low strains, a *plateau* region showing high sensitivity to stress, and a region of *densification* when very high stress is applied, see also Fig. 3.2. Elastomeric materials – such as the widely-used polyurethane (PUR) foams – exhibit a monotonically increasing strain-stress relation [33, Ch. 5]. This allows the stress (or force) to be uniquely determined from the observed strain (normalized deformation). Manufacturers typically guarantee certain production tolerances and measure the deformation of their materials at several points, according to the ISO 2439 standard.

To obtain a complete strain-stress curve for a given deformable element, a calibration procedure is performed: A robot arm pushes a large metal plate slowly onto the foam rod, increasingly compressing it. The position (i.e. strain) is known with high accuracy from the arm controller, while the applied force (or pressure, stress) is measured with a JR3 force sensor mounted on the arm. The process is repeated multiple times to verify repeatability, yielding the data points shown in Fig. 3.2. Different cross sections of the foam rod have been tested, but for now, a regular rectangular cross-section is considered. There are some time-dependent relaxation effects in the material – yet, they are negligible for small manipulation velocities as used by our system. As expected from literature [33], the data points form a curve with three different regions. Here, we mainly rely on the *plateau* region for normalized strains in $[0.1, 0.5]$, which corresponds to a reasonable range of forces for the application at hand. Additionally, this region allows for the most accurate measurements, since the sensitivity of the material to force changes is largest. Note also that the curve exhibits a strong hysteresis effect, depending on whether forces are increased (red curve) or decreased (blue points). Therefore, we only measure the displacement while forces are *increased*.

There are several boundary effects to be considered: Local material stiffness decreases towards the edges of the foam, since support towards one direction is missing. Yet, note that only the effective stiffness along the entire side h is relevant, since the foam deforms equally along w. Dimensions s, w, h are used as in Fig. 3.1a and 3.2e. Calibration is performed on the foam rod with the final cross-section and therefore accounts for the effective stiffness. However, stress discontinuities along s (major axis) require special consideration: During measurement, if the stress applied to the foam is a step function along s (e.g. caused by the edge of an object), the front contour deforms smoothly beyond the contact area, see also shape C in Fig. 3.7b. The smoothing is attributed to internal tension in the foam – so to say, the deformed region pulls in other parts of the foam by lateral forces. As a result, the effective stiffness of the foam is increased near the borders of the contact region. This effect is approximated by assuming that the smoothed contour is actually caused by an enlarged contact area. During calibration there are no such discontinuities, since the metal plate is larger than the plastic foam rod.

A third-order polynomial m_P is fitted to the points obtained from calibration, yielding a phenomenological model for the strain-stress relationship, which is depicted as a red curve in Fig. 3.2. The curve fits the datapoints well, except for the *densification* region, which corresponds to large forces and eventually the saturation of the sensor. The total contact force is obtained by integration over the stress (or pressure), using the calibrated model m_P, sensor width w and height h as well as the deformation $\delta(s)$ over s:

$$F = h \int_{s=s_1}^{s_2} m_P\left(\frac{\delta(s)}{w}\right) ds \approx h \sum_{s=s_1}^{s_2} m_P\left(\frac{\delta(s)}{w}\right) s_d \tag{3.1}$$

$$\text{with } m_P(x) = [c_0, c_1, c_2, c_3] \cdot [1, x, x^2, x^3]^T \tag{3.2}$$

Each object causes one contact region, which is represented by the interval $[s_1, s_2]$, where $\delta(s) > 0$. If multiple objects are in contact with the foam, Eqn. (3.2) is evaluated separately for each contact region. In practice, s is sampled at discrete points spaced s_d apart, see points in Fig. 3.6 (detail, left).

Beams The deformation of beams or rods under load is described by the Euler-Bernoulli beam theory [37] for small deflections. Here, a force and/or moment perpendicular to the beam is applied at its frontal end (F), see Fig. 3.5. The other end of the beam (S) is considered to be fixed ("clamped"). In principle, the beam can be made of any elastic material such as metal or plastics. Yet, no plastic (i.e. permanent) deformation should occur within the relevant deformation range. We use a beam made of spring steel with a diameter of about 2 mm and a length of $10 - 20$ cm.

The deflection curve $\omega_Y(x)$ (usually w in literature) describes the deformation of the beam perpendicular to the X-axis, whereby the X-axis is aligned to the beam in resting position:

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2}\left(EI\frac{\mathrm{d}^2\omega_Y}{\mathrm{d}x^2}\right)\Bigg|_{EI=const} = EI\frac{\mathrm{d}^4\omega_Y}{\mathrm{d}x^4} = q(x) \tag{3.3}$$

This is the commonly-used approximation for small deflection, which ignores the shortening of the deformed beam along the X-axis. The values for the elastic modulus E and the second moment of area I are constants for uniform and homogeneous beams. For a circular cross section of the beam with radius r, $I = \frac{\pi}{4}r^4$. There is no distributed load, such that $q(x) = 0$. Quadruple integration of Eqn. (3.3) yields four integration constants, which are used to fulfill the boundary conditions. Due to the clamping at $x = 0$, $\omega_Y(0) = \omega_Y'(0) = 0$. The derivatives of ω_Y have a distinct physical meaning – specifically the moment is $M = -EI\omega_Y''$, and the shear force is $Q = -EI\omega_Y'''$. Therefore, when a force F is applied perpendicularly to the beam at its end, boundary conditions are:

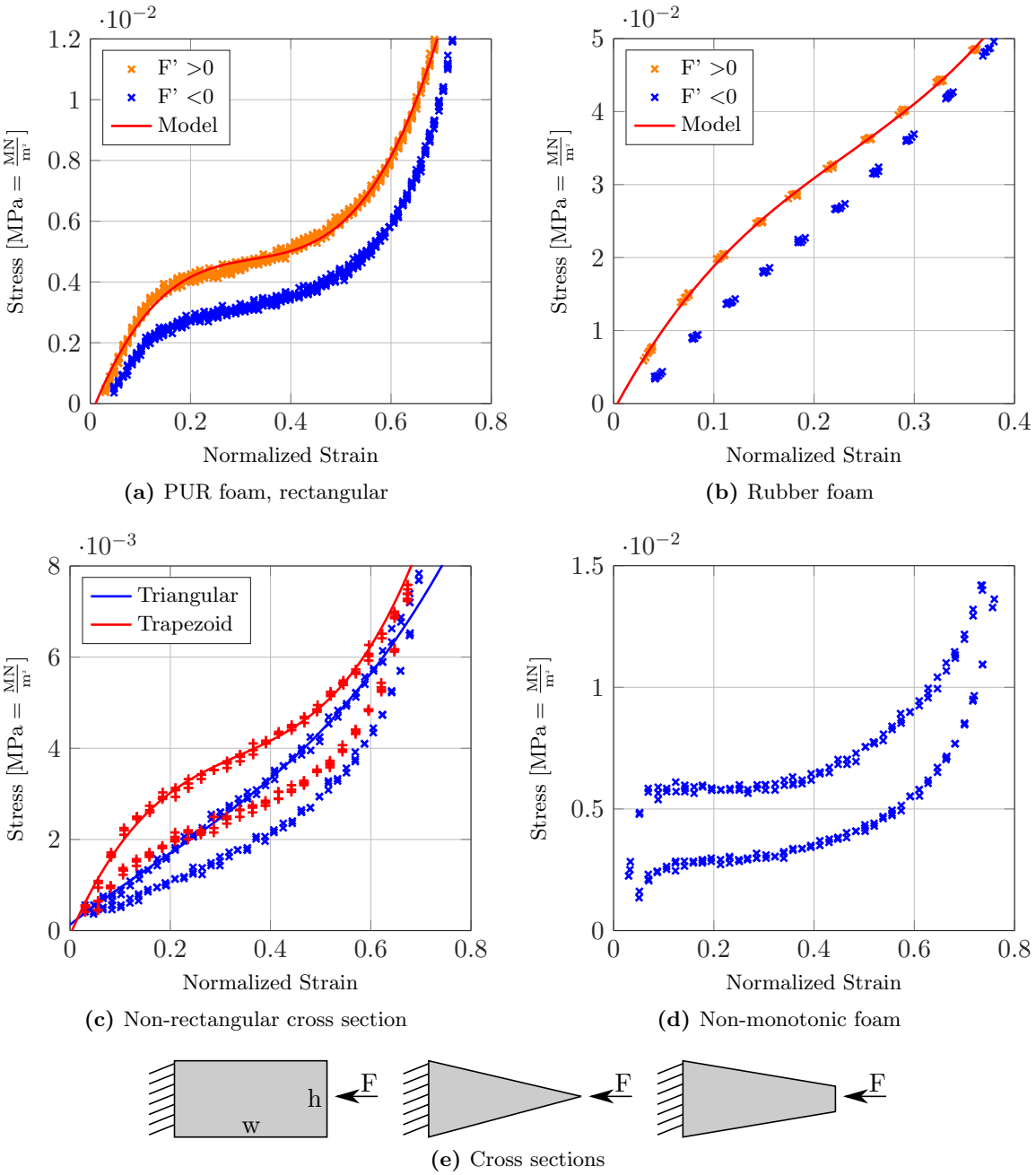
$$\omega_Y'''|_{x=L} = -\frac{F}{EI} \text{ and } \omega_Y''|_{x=L} = 0$$

The force is applied at $x = L$, where L is the length of the beam. With four boundary conditions, a unique solution $\omega_{Y,F}$ can be given for Eqn. (3.3). Similarly, for a moment or torque M applied at $x = L$, a solution $\omega_{Y,M}$ is determined using the boundary conditions:

$$\omega_Y'''|_{x=L} = 0 \text{ and } \omega_Y''|_{x=L} = -\frac{M}{EI}$$

Since the differential equation is linear, the two solutions can be superimposed. For a force and moment applied at $\xi = 1$, with $\xi = \frac{x}{L}$ and clamping at $\xi = 0$, we obtain:

$$\omega_Y(\xi) = \omega_{Y,F} + \omega_{Y,M} = \frac{FL^3\left(3\xi^2 - \xi^3\right)}{6EI} + \frac{ML^2\xi^2}{2EI} = -\frac{FL^3}{6EI}\xi^3 + \left(\frac{FL^3}{2EI} + \frac{ML^2}{2EI}\right)\xi^2 \tag{3.4}$$



(a) PUR foam, rectangular

(b) Rubber foam

(c) Non-rectangular cross section

(d) Non-monotonic foam

(e) Cross sections

Figure 3.2: Experimentally determined strain-stress relation for the rectangular PUR foam *(a)* and the rubber foam *(b)*. Data points are obtained while increasing (orange) or decreasing (blue) stress. The red curve shows the polynomial model m_P fitted to the orange points. The stress obtained with different cross sections *(c,e)* is related to the same area as in *(a)* for comparability. Foams with a non-monotonic strain-stress behaviour *(d)* are unsuited for the proposed sensor.

The deflection of the real beam is observed by a camera at one or multiple points of the curve ω_Y. Extension of the discussed 2D case to 3D is straight-forward by separation along y and z, yielding deformation curves ω_Y, ω_Z. Forces along X or moments around the X-axis cause small deformations that are almost unobservable. They are therefore handled differently, as outlined in Sec. 3.4.

A model according to Eqn. (3.4) is fitted to the observations and solved for F, M to obtain the causing force and torque. In the following, it is assumed that ω_Y and ω'_Y are known only at a single point $\xi_1 \in]0, 1]$. By rearranging Eqn. (3.4) and the derivative thereof, a linear expression $\hat{\omega} = W \cdot \mathbf{f}$ is obtained in matrix form, with $\hat{\omega} = [\omega_Y(\xi_1)\ \omega'_Y(\xi_1)]^T$ and $\mathbf{f} = [F\ M]^T$. Given an observation $\hat{\omega}$, the acting force/moment $\mathbf{f}$ is obtained by inversion of W:

$$\mathbf{f} = W^{-1} \cdot \hat{\omega}, \quad W^{-1} = \frac{2EI}{L^2\xi^2} \begin{bmatrix} \frac{6}{L\xi} & -\frac{3}{L} \\ \frac{3(\xi-2)}{\xi} & -(\xi-3) \end{bmatrix} \Big|_{\xi=1} = \frac{2EI}{L^2} \begin{bmatrix} \frac{6}{L} & -\frac{3}{L} \\ -3 & 2 \end{bmatrix} \tag{3.5}$$

If multiple observations are available, we obtain an overdetermined system $[\hat{\omega}|_{\xi_1}; \hat{\omega}|_{\xi_2}; \ldots] = [W|_{\xi_1}; W|_{\xi_2}; \ldots] \cdot \mathbf{f}$, which is solved for $\mathbf{f}$ by least-square minimization. From Eqn. (3.5), note that F is significantly more sensitive to changes in $\hat{\omega}$ than M for the typical case $L < 1\,\mathrm{m}$. Also, there is a strong coupling of the two components.

3.3 Foam-based sensor for laminar contacts

Force or touch sensors for a laminar or linear contact with an object provide pressure and shape measurements along the contact area. For grippers and mobile platforms, see Fig. 3.1, elongate contact areas are feasible, which can be considered a linear contact. In the proposed sensor, plastic or rubber foams are used for the deformable element, whose deformation is measured visually along the line or curve of one of the visible edges. Using this contour-based approach, deformation and forces perpendicular to the edge can be measured in a dense fashion. Force (or pressure) is derived from the deformation using the known deformation characteristics of the material, see Sec. 3.2. A camera is mounted about 20 cm above the deformable element and points towards it. It shows the top surface of the foam, the floor and scene in its direct vicinity as well as a part of the manipulator or platform itself (see Fig. 3.1 and 3.6). Here, a consumer Full HD USB camera with a diagonal field of view of $\pm 40°$ is used. Such devices exhibit a good image quality and are low-cost due to the large proliferation of similar devices in smartphones. The focus is fixed by software and set according to the distance of the foam. This results in a slight blur of the nearby scene.

On our platform the deformable element is a passive foam rod (orange part in Fig. 3.1a), which is roughly 25 cm long (major axis) and exhibits a cross section of $w \times h = 2 \times 1\,\mathrm{cm}$. A standard PUR (polyurethane) foam is used, which costs only a few cents and can be easily replaced in case of attrition. It is bent to fit to the round platform. The cross section is chosen based on the deformability of the material and the desired force range. One of the

long sides is attached to a rigid part of the robot case, which may be straight or curved, and the opposing side comes into contact with objects. The direction of exploration, denoted $\mathbf{x}$ in Fig. 3.1a, corresponds to the dominant (forward) motion of the platform on the floor. Forces act along vectors $\mathbf{d}(s)$ and deform the foam rod along its width w. Deformation is expressed as a scalar function $\delta(s)$ of displacement [in m], which is densely sampled along s. The normalized strain of the material is $\delta(s)w^{-1}$. Sheer forces parallel to the mounting curve are negligible in this setup and are not measured.

Similarly, the sensor can be used for a grasping system, as depicted in Fig. 3.1b. A two-finger gripper with a linear drive is equipped with rectangular rods of rubber foam on the inside of the fingers. When the gripper closes, the foam comes in contact with the object and deforms depending on the local pressure. Cameras (one for each finger) are mounted above the gripper, such that they observe the top of the fingers and the top surface of the foam material. To prevent occlusion of the foam by the object, cameras are placed slightly outside of the grasping range.

3.3.1 Tracking of foam deformation

The front contour of the foam rod (F) deforms when it comes into contact with an object, see Fig. 3.1. An edge-based tracker determines the amount of deformation (in meters) along the contour. Additionally, the back contour (B) between the foam rod and the mounting plate (e.g. the rigid surface of the platform or the metallic finger of a gripper) is tracked to obtain a reference contour (see below). The use of image edges is most feasible for the application at hand, since the foam has no stable inner visual features. Contour detection based on image edges is stable regardless of lighting conditions, except in complete darkness. The edge strength varies considerably depending on the visual appearance of objects touching the sensor, which must be accounted for by the algorithm. In the rare case where brightness, color and shading values of the foam rod equal those of an object, the edge at the foam's contour would disappear. To prevent this case, it is possible to work with a foam material that changes its color along the major axis.

Edges are tracked using the well-known concept of snakes, see Sec. 2.3.2, which consist of connected points $\mathbf{u}_k$ "snapping" to image edges. We track points along the contour of the foam spaced about $s_d = 3\,\text{mm}$ apart, which allows for an accurate representation of possible deformations. After initialization, snake points move within a certain local neighborhood to iteratively minimize an energy term, which consists of several parts, see Eqn. (3.6): First of all, the negative amount of edge strength e_e tries to keep the points on image edges. In order to avoid snake points snapping to strong internal edges within an object, e_e is limited to the average edge strength/gradient $\overline{\nabla}$ of all snake points. A smoothness term e_s accounts for the fact that the foam contour cannot jump, i.e. the slope of $\delta(s)$ is limited. Furthermore, there are no edges within the foam rod, i.e. in between the inner and outer contour. Therefore, an energy term e_j penalizes points jumping over edges along the path from the point on the

reference contour $\mathbf{u}_k^r$ to $\mathbf{u}_k$. Finally, e_c constrains point motion to the vector of deformation $\mathbf{d}$, which is perpendicular to the reference contour of the foam rod. As this is the major direction of deformation, snake points stay on the same physical point on the foam, and the sampling density remains constant. The total energy is evaluated and minimized in a local neighborhood $\xi_\mathbf{u} = \mathbf{u}_k + [\delta x, \delta y]^T$:

$$e(\xi_\mathbf{u}) = \mathbf{w} \cdot [e_e, e_s, e_j, e_c]\,, \quad \text{with} \tag{3.6}$$

$$e_e = -\min(|\nabla G * i|, \overline{\nabla}),$$

$$e_s = (\mathbf{u}_{k-1} - 2\xi_\mathbf{u} + \mathbf{u}_{k+1})^T(\mathbf{u}_{k-1} - 2\xi_\mathbf{u} + \mathbf{u}_{k+1}),$$

$$e_j = \frac{1}{\overline{e}} \int_{\mathbf{x}=\mathbf{u}_k^r}^{\xi_\mathbf{u}} |\nabla G * i| d\mathbf{x},$$

$$e_c = \begin{cases} 0, & \text{if } (\xi_\mathbf{u} - \mathbf{u}_k^r)||\mathcal{P}^{-1}\mathbf{d}(s) \\ \infty, & \text{otherwise} \end{cases}$$

Where i – image, ∇ – gradient operator, G – Gaussian blur operator for noise reduction, $\mathcal{P}$ – projection operator, see Eqn. (3.7). Weights $\mathbf{w}$ are set such that energy terms are in $[-1, 1]$ within the search space. A one-dimensional constraint is imposed by e_c, keeping each point on a line through the reference point and along the direction of deformation $\mathbf{d}$. Thus, the search for the optimum is fast even for large neighborhoods. Processing at frame rate (30 Hz) poses no problem to a mid-range Intel i5 platform. Note that it is not feasible to integrate shape priors in the energy term, as in more recent work about snakes. The contour of the foam is solely determined by the shape of the obstacles, and the correlation of close-by values of $\delta(s)$ is accounted for by e_s.

In some cases, points on the front contour $\mathbf{u}_k$ may be pulled onto the object by strong edges within the object texture. This effect can be avoided by adding an internal contour to the top surface of the foam. For that purpose, a narrow color stripe is added to the rubber foam on the gripper, see Fig. 3.9b. The snake points $\mathbf{u}_k$ now snap to the edge between the black foam and the green stripe. Since this edge is constant, the localization accuracy of points $\mathbf{u}_k$ is improved. The effective width of the foam strip becomes smaller, and w must be adapted accordingly to account for the changed deformation behavior. Additionally, the color stripe is used to detect occlusion of the foam by the object. Pixels located directly in front of $\mathbf{u}_k$ exhibit the expected (green) color, unless the foam is occluded. The corresponding points $\mathbf{u}_k$ are invalid in this case. Pixel colors are classified using a Gaussian mixture model, which is trained to the observed color stripe during initialization.

The approximate position of the foam rod in the camera image is typically known from a geometric robot model. Otherwise, it may be located using markers – such as the template image mounted on the gripper in Fig. 3.9b. First, the inner snake is initialized by adding points iteratively at a constant distance and having them snap to edge pixels. Points $\mathbf{u}_k^r$ on

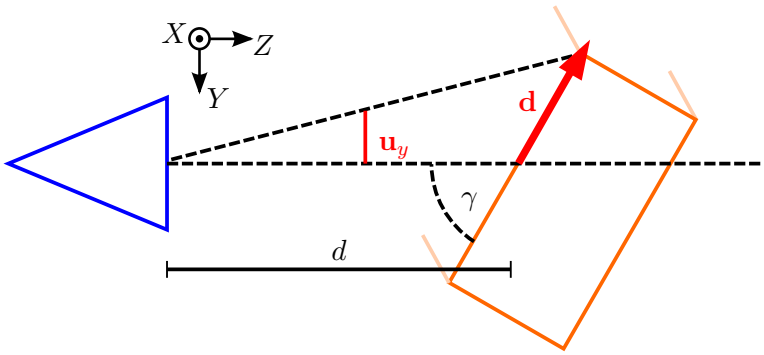


Figure 3.3: Simplified arrangement of camera and foam, side view: The camera (left) looks onto the top surface of the foam (right) along the dashed axis. Deformations along $\mathbf{d}$ are projected onto the y-axis of the image, parallel to Y.

the inner snake serve as a reference position of the sensor base, denoted (B) in Fig. 3.1b. On a gripper, the linear joint moves this reference contour with respect to the reference frame (R). Yet, even on a static setup, (B) may change slightly due to vibrations and movement of the mounting plate or the camera. Next, points of the outer snake are initialized slightly outside of the inner snake. To allow for varying rod widths, these points are pushed away from the inner snake by an additional energy term until they reach the stable outer edge. The idle state of the outer snake is used as the zero reference of displacement δ.

Pixel positions on the snake are converted to real-world coordinates using the intrinsic matrix of the camera K and a coordinate frame $\tilde{\mathcal{T}}^F = (\mathbf{X_0}, R)$ at the center of the foam rod spanned by the exploration vector $\mathbf{x}$ and $\mathbf{y}$ parallel to the floor. The (bent) major axis along s and deformation vectors $\mathbf{d}$ lie on the $x-y$ plane of this coordinate frame. In a robotic system, $\mathcal{T}^F$ is determined from the extrinsic camera parameters and the geometric robot model. Otherwise, the pose can be determined, as mentioned, by markers. In this manner, the real amount of deformation can be obtained from 2D information provided by the camera. The casting operator $\mathcal{P}$ projects a point in the image $\tilde{\mathbf{u}} = [x, y, 1]^T$ [in pixel] via the camera-centered coordinate frame $\tilde{\mathcal{T}}^C$ to a point $\mathbf{X}^{(F)}$ [in meters] on the $x-y$ plane of the reference frame $\tilde{\mathcal{T}}^F$ (the sensor plane) with normal $\mathbf{n} = R_{:,3}$:

$$\mathbf{X}^{(C)} = \frac{\mathbf{X}_0 \cdot \mathbf{n}}{\tilde{\mathbf{u}}' \cdot \mathbf{n}} \tilde{\mathbf{u}}', \quad \tilde{\mathbf{u}}' = K \cdot \tilde{\mathbf{u}} \tag{3.7}$$

$$\mathbf{X}^{(F)} = R(\mathbf{X}^{(C)} - \mathbf{X}_0), \qquad \mathcal{P} : \tilde{\mathbf{u}} \to \mathbf{X}^{(F)}$$

3.3.2 Analysis of measurement accuracy

Visual measurements of the deformable element are error-prone due to several systematic and random error sources. The following discusses how accuracy is affected by tracking noise, sensitivity of the system model, foam cross sections, pose errors and rolling shutter effects.

In a crisp image, without using approaches based on sub-pixel processing, the certainty of edge location is limited to $\pm 0.5\,\mathrm{px}$ along the x, y-axes of the image. Additionally, edge locations may be biased by irregular or changing illumination. In some cases, strong intensity discontinuities on objects close to the sensor may influence edge tracking. Bias effects depend on the surrounding scene and change with a much lower frequency than image noise. In practice, the value is influenced by a number of effects, such that a Gaussian noise model is more appropriate, see Sec. 3.6.1.

In the following, the dependence of the sensor output – i.e. pressure or force – on noise in the image space is analyzed based on a simplified system model. Assume a simplified arrangement of camera and foam rod as depicted in Fig. 3.3: The x-axis ("right") of the image is parallel to X and to the sensor plane (spanned by $\mathbf{s}, \mathbf{d}$), and deformations $\mathbf{d}$ are projected to the y-axis ("down") of the image, parallel to Y. The camera is parametrized by its focal length in pixels f, the distance to the edge point d and the angle γ between the camera axis and the sensor plane. In that case, the projection of a point $\mathbf{X}^{(F)}$ on the foam contour to a point $\mathbf{u}$ in the image simplifies to:

$$\mathbf{u}_y = \frac{1}{d} f \sin(\gamma) \mathbf{X}_Y^{(F)} + C \tag{3.8}$$

The foam deformation is obtained from the difference of two snake points according to $\delta = \mathbf{X}_Y^{(F)} - \mathbf{X}_Y^{(F),R}$, corresponding to $\mathbf{u}_k$ and $\mathbf{u}_k^r$ in the image. Thus, the constant offset C cancels out. It accounts for the point of origin of $\mathbf{X}^{(F)}$ in the camera frame and for the principal point. General camera arrangements can be brought to the form of Eqn. (3.8) by coordinate transformations. The pressure is obtained as a function over the deformation δ by plugging the inverse of projection Eqn. (3.8) into the polynomial model m_P, see Eqn. (3.1). Its sensitivity to errors in the image space is expressed by the derivative of $m_P(\delta(\mathbf{u}_y))$:

$$\frac{\partial m_P}{\partial \mathbf{u}_y} = \frac{\partial m_P}{\partial \delta} \cdot \frac{\partial \delta}{\partial \mathbf{u}_y} = \frac{\partial m_P}{\partial \delta} \cdot d \frac{1}{f} \sin^{-1}(\gamma) \tag{3.9}$$

This function is plotted over the normalized strain (foam compression) in Fig. 3.4 for a setup as used on the mobile platform, with camera distance $d = 0.3\,\mathrm{m}$, viewing angle $\gamma = 70°$ and focal length $f = 1800\,\mathrm{px}$ (equal to the diagonal of the image). Foams with different cross sections are used, see also Sec. 3.2. In case of the standard rectangular cross section, the derivative of pressure with respect to the deformation value is high for small and large strains. This corresponds to a low accuracy of pressure – small changes in the image location have a large influence on the pressure value, and image noise as well as systematic errors get amplified. The high value of the derivative is problematic for low values of the normalized strain – especially so for zero strains: Contact can only be detected if the pressure value is above the noise threshold. Therefore, it is most important to reduce the pressure derivative for low strain values. At high strain values, the high value of the derivative effectively increases the measurement range at the expense of accuracy – an acceptable trade-off.

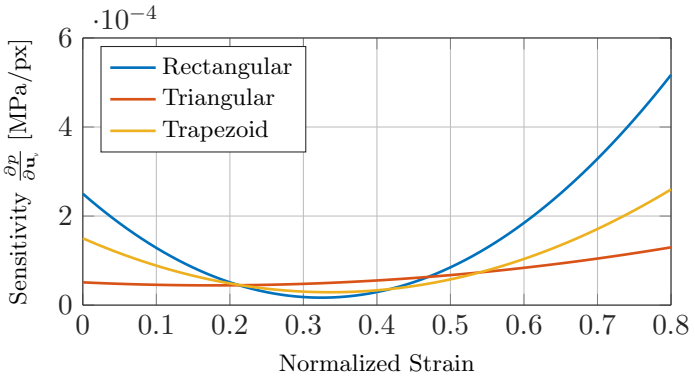


Figure 3.4: Sensitivity of stress measurements to the image (pixel) position for the tested foam rods, see Fig. 3.2, with a typical camera setup.

Foams with triangular and trapezoid cross sections, see also Fig. 3.2, show a more desirable sensitivity curve: The former shows an almost constant low pressure derivative for low and mid-range strains, and for the latter, the "barrier" at zero strain is significantly reduced. The thinner parts of these rods are less stiff and deform first when a low pressure is applied. While increasing the pressure, deformation affects more and more of the thicker parts. Consequently, much lower pressure values are observable, and contact of lighter objects can be detected. At the same time, the depth of the "valley" at mid-range strains and thus the accuracy in this range is reduced. In practice, however, this is barely noticeable.

Despite the high accuracy over the entire range of strains, there are some significant drawbacks of the triangular cross section: First, there is no strong increase of the pressure derivative for high strains, reducing the usable range of the sensor. Second, the front of this foam rod tends to bend up or down when it comes into contact with an object. Like that, reliable measurements are not possible in practice. The trapezoid cross section, on the other hand, does not show these drawbacks and is thus considered the first choice for a versatile sensor. Note that the overall level of the derivative and thus the overall range of measurements is lower for both foam rods, compared to the rectangular rod. This is easily compensated by a stiffer foam material or by an enlarged cross section.

One pressure sample relies on two snake points, i.e. the two edge locations of the reference and front snake. The standard deviation of one pressure sample is therefore $\sigma_{m_P} = \sqrt{2}\frac{\partial m_P}{\partial \mathbf{u}_y}\sigma_{\mathbf{u}}$, with Gaussian noise of standard deviation $\sigma_{\mathbf{u}}$ in the image space. Since snake points are spaced several pixels apart, noise is independent. The total force is obtained by summation over multiple snake points, see Eqn. (3.1). Assuming equal values of σ_{m_P}, the standard deviation of force becomes:

$$\sigma_F = \sqrt{N} h s_d \sigma_{m_P} = \sqrt{2N} h s_d \frac{\partial m_P}{\partial \mathbf{u}_y} \sigma_{\mathbf{u}}, \tag{3.10}$$

with N – number of contact points, hs_d – frontal area of one partial foam element.

Errors of camera pose – i.e. the extrinsic parameters – may also be investigated using Eqn. (3.8). Since deformation is measured with respect to the reference contour, any camera motion parallel to the sensor plane is irrelevant. However, image distortion, if not compensated, results in a slight change of the effective focal length across the image. Deviations of camera distance d or the inclination angle γ, on the other hand, directly result in an incorrect scaling of the measured distance. Zeroing, which is performed during initialization, compensates for this scaling: The maximum width of the foam, without any external forces applied, is saved as a reference for zero normalized strain. This step also accounts for varying widths of the foam rod. A normalized strain of 1 is always given by the reference contour. Consequently, small pose errors can always be compensated, if a zeroing process is feasible.

Most cameras without a mechanical shutter show the so-called "rolling shutter" effect, i.e. the image rows are exposed and read out sequentially at different points in time. For typical video cameras, the time delay between two consecutive rows is $(rn)^{-1}$ for frame rate r and number of rows n. Therefore, camera motion or shaking – in addition to the above-discussed effects – result in a time-dependent scaling of the measured deformation δ along the y-axis of the image. For compensation, the camera motion is determined by tracking of the reference contour. The relevant component of camera motion is along the direction of deformation, which is perpendicular to the reference contour. Given the camera motion and the parameters of exposure, compensation of scaling caused by the rolling shutter is straight-forward.

Finally, inaccuracies in the deformation models result in systematic measurement errors. For instance, our model does not consider lateral deformation directly, see also Sec. 3.6.1. As discussed, there is also a strong hysteresis effect when the direction of deformation is changed. Furthermore, foam stiffness may not be constant, but rather depend on deformation speed, wear of material, temperature or even humidity. These effects, however, go beyond the scope of this work. If required, more accurate deformation models may be built based on a more detailed analysis of foam deformation.

3.3.3 Fitting of geometric primitives

In household or office environments, a robot often grasps or interacts with artificial objects of relatively simple shapes. The mobile platform encounters walls, boxes, table legs, cylindrical trash bins or vases, and for grasping, bottles, boxes, glasses or other containers are highly relevant. When an object is in contact with the sensor, the impression in the foam corresponds to a partial 2D contour of the object. As these artificial objects are aligned perpendicular to the floor and exhibit symmetry, a 2D "footprint" derived from this contour is often sufficient for localization, mapping, navigation or evaluation of the grasp quality. The footprint may refine visual measurements or complement missing data, such as transparent objects. Additionally, it shows how an object responds to manipulation, i.e. how it deforms when being pushed.

Two-dimensional primitives are fitted to the impression of the foam within the contact region (points $\mathbf{X}_k^{(F)}$). They provide a shape estimate of an object beyond the contact re-

gion. The following primitives, which represent typical objects, are used: Lines (for walls, large boxes), line segments (small boxes), corners (boxes, table legs) and circles (trash bins, bottles, glasses, mugs). Line-based primitives are fitted into the point set using least-squares minimization. A line segment is used if the center points of the deformation lie on a line, but the end points do not. Corners are described by two lines that intersect at the point of maximal deformation. Circles are fitted using the algebraic approach described in [81].

Each candidate primitive is tested on how well it fits to the observed points $\mathbf{X}_k^{(F)}$. A score is calculated, based on the mean squared distance between observed points and closest model points as well as the number of points n confirmed by the model:

$$\mathcal{S}_p = \exp\left(-\frac{1}{N}\sum_{k=1}^{N}\frac{1}{\sigma_2^2}d_M^2(\mathbf{X}_k^{(F)})\right)\cdot\frac{n}{N} \tag{3.11}$$

Where d_M – distance of a point from model M, N – number of contact points (deformation set). The primitive with the highest score and $\mathcal{S}_p > 0.5$ is chosen. It is used for the refinement of a map, see Sec. 4.3.3, or for purely haptic mapping, see Sec. 3.6.3. Furthermore, with prior information from vision, the primitive can help to distinguish objects with different shapes (see Sec. 6.3.3). When an object is grasped, the grasp configuration can be verified and corrected if needed based on a comparison of the fitted primitive with a reference.

3.4 Beam-based single point 6D force-torque sensor

While the foam-based sensor provides dense force samples along the foam contour, it is limited to forces acting perpendicularly to that contour. Many applications only require a force reading at a single point, but in multiple dimensions – ideally as a full 6D force-torque reading. Typically, such sensors are mounted between the end-effector of a robot arm or another actuator and a tool – such as a gripper, screwdriver or specialized manipulator. Upon contact, forces and torques imposed by the tool onto an object (or vice versa) must be measured in up to six dimensions. Also, a human could guide the arm by imposing forces onto the tool.

Therefore, another design of a visuo-haptic sensor is proposed, based on a deformable element made of a metal beam and a spring, which together deform in 6D when a force-torque is applied. Again, the passive system is measured by an external camera, which also observes the surrounding scene. It allows measuring forces/torque in multiple dimensions at a single point – much like commercial force-torque sensor modules, see Sec. 2.1. Note that the beam theory can be applied to beams of any shape by adaptation of the shape constant I. Therefore, a soft connection element within the arm – such as the link between the last two joints – could be used as the deformable element.

The proposed sensor is built according to Fig. 3.5 and consists of a round rod of spring steel (the beam) between points (S) and (F) as well as a guided spring between (B) and (S).

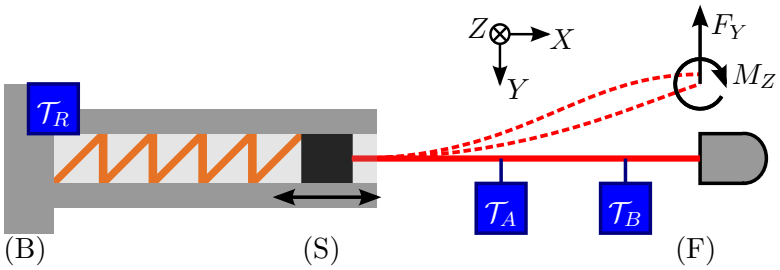


Figure 3.5: Proposed beam-based sensor, based on a round beam made of spring steel (red). Deflections $\omega_Y(x)$ caused by two different forces/torques are shown by dashed lines. The spring (orange) is deformed by a force along or a moment around the X-axis. A camera observes the entire structure and tracks templates $\mathcal{T}_{R,A,B}$.

The sensor base (B) is mounted to the actuator, which is typically a robot arm. The tool is mounted on the other end (F). A camera observes the entire sensor from (B) to (F), as well as the scene around it. Forces or moments/torques applied at (F) along or around axes Y, Z lead to a deflection of the beam as described in Sec. 3.2. The deflections are separable into ω_Y and ω_Z and then processed as in the 2D case. Note, however, that ω_Y represents forces along Y and moments around Z. Due to the guidance along X in point (S), the spring is not affected by these forces/moments.

Forces or moments along or around X hardly deform the beam, since it is very stiff along this direction. Instead, a force along X results in a compression of the spring between (B) and (S) according to Hooke's law. Similarly, a moment around X results in a torsion of the spring. The corresponding deformation models are straight-forward, and the stiffness constants are adjusted independently of the beam. Using this design, in principle, a full 6D force/moment can be read from the beam/spring deformation.

Observations of the deflection curve of the beam ω_Y may be obtained in different ways: An edge tracker, similar to Sec. 3.3.1, may be used to track the two contours along the beam and provide a dense sampling of ω_Y. As noted in a similar method [49], this approach provides a high robustness to occlusion, due to the large number of tracked points. However, point tracking does not provide any reliable depth information, basically limiting force measurements to 1D. Simple features along the beam – such as colored spheres – would allow for easier detection, yet still with very limited depth resolution. A template tracker (see Sec. 2.3.2), on the other hand, provides a full 6D pose, including accurate depth values and the full rotation, which corresponds to the derivatives $\omega'_{Y,Z}$. Planar textured templates are attached along the beam at a single fixture point, respectively. They may be integrated into the design of the robot case. The pose of the beam at the fixture point and the template are related by a constant rigid transformation determined by the mounting structure. Only a low number of templates can be attached along the rod, resulting in a limited number of observations of ω_Y and ω'_Y. The acting force/moment is calculated using Eqn. (3.5) as outlined there.

As indicated in Fig. 3.5, three templates are attached to the sensor structure. The reference template $\mathcal{T}_R$ on the sensor base provides a local reference frame $\tilde{\mathcal{T}}_R$, in which all sensor-related transformations are expressed. Like that, the pose of the sensor is not required a priori, and the camera can move relative to the sensor. In principle, the reference frame may be fixed to the camera frame, or it may be calculated from a robot model. Yet, errors can be reduced significantly, if a reference frame is obtained within the image, close to where measurements are performed. Two templates with corresponding poses $\mathcal{T}_{A,B}$ are attached along the beam at fixed relative positions $\xi_{1,2}$. Since the templates are not aligned perfectly, and the beam might exhibit some plastic deformation, a zeroing procedure is performed at startup, with zero force/moment applied to the sensor. The corresponding resting poses $\mathcal{T}_{A0,B0}^{(R)}$ are expressed relative to the frame $\tilde{\mathcal{T}}_R$.

Measurements of the template poses are performed for each frame and expressed relative to their (constant) resting poses, which in turn are based on the current (variable) reference frame $\tilde{\mathcal{T}}_R$. Thus, the poses $\mathcal{T}_A^{(A0)}, \mathcal{T}_B^{(B0)}$ are zero as long as the beam is not deformed – regardless of the current sensor orientation. Since the Y and Z components are processed separately, the respective translational components of $\mathcal{T}_\star^{(\star 0)}$ correspond directly to samples of $\omega_{Y,Z}$. The slope is obtained from the 3D rotation matrix R, projected to the respective 2D plane, such that $\omega'_Y = {}^{R_{1,0}}/_{R_{0,0}}$ and $\omega'_Z = {}^{R_{2,0}}/_{R_{0,0}}$. Finally, the compression and torsion of the spring is obtained from the X-component of the translation and the respective rotation angle. Force and translation – respectively moment and torsion – are related by the spring stiffness constant.

3.5 Tracking objects and scene

Besides tracking the deformable element, the camera observes the vicinity of the robot platform or actuator – specifically the manipulator itself, the floor or table plane, approaching obstacles and objects in contact with the sensor. Tracking of the tool is important to obtain a reference frame close to the manipulated object. In case of an articulated tool, such as a gripper, the state of the manipulator is determined in addition. Tracking of the environment allows for accurate estimation of the platform motion. Finally, the object is tracked in order to determine its reaction during the manipulation process, see also Sec. 4.1. Specifically, rotations or unexpected sudden motion during contact, such as falling over, are of relevance to detect a failed grasp or push. Besides simple tracking, an object model can be fitted to the manipulated object. Like this, the exact object pose is determined – albeit at considerably higher computational costs.

In case of the platform sensor, approaching obstacles are detected in order to allow the robot to slow down before touching them. This requires object and background to be separable. Here, we consider objects standing on a plane, such as the floor or the tabletop. Detection is again performed in the same camera image to obtain coherent data in the direct proximity

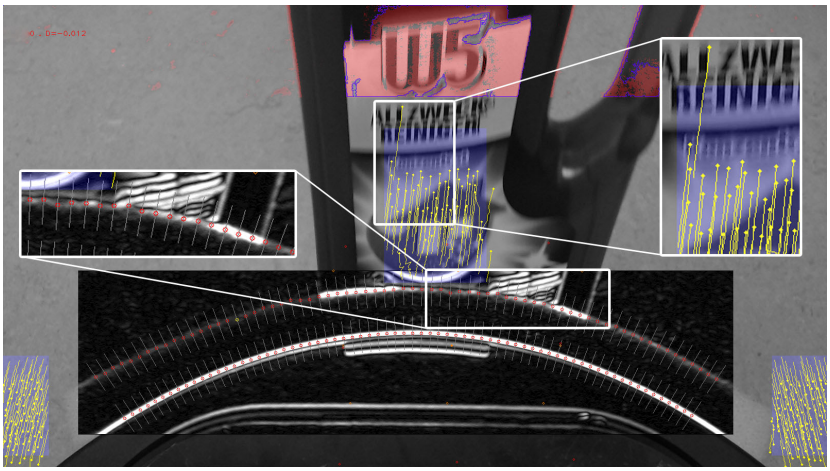


Figure 3.6: Visual point features are tracked on the object surface (yellow), as well as on the floor to measure motion in the respective areas. Here, the object is just falling, and the track is lost due to motion blur. Foreground areas detected by the Gaussian mixture model are depicted in red (top). Figure adapted from [6].

of the sensor. Obstacles can be detected based on visual appearance differences or based on their height – since they are closer to the camera than the floor, they move faster. As feature tracking of the floor is problematic at higher speeds due to motion blur, we chose the former approach. An efficient way to model visual appearance is by using Gaussian mixture models, which are trained based on a set of expected images and detect contradiction to the trained model. Here, the approach from [104] is used, which handles the training process automatically and gradually adapts the model to the scene. The detector is trained automatically and quickly adapts to a change in the environment by learning the new floor model within a few frames. Obstacles are searched for only in the upper part of the camera image where they appear first, see Fig. 3.6.

While an object is being pushed, a Kanade-Lucas-Tomasi (KLT) tracker [64] tracks the floor and the object itself. Floor tracking provides an estimate of ego-motion, complementing the inertial sensors. The KLT feature extractor searches for corner-like structures in the image, which are well-localized and thus suitable for tracking. In order to ensure reliable motion estimates, only stable features are selected for tracking within the appropriate regions of the image using the scores from [89]. Features for floor tracking are selected within small patches left and right of the foam. For the object, a mask is created based on the fitted primitive. Tracks span several frames by matching of features between frames, yielding a sparse motion field of the floor and the object. These feature tracks are depicted in Fig. 3.6. Since the pushing distances are rather short, the drift of the tracker remains negligible. Tracking works well on textured surfaces, but the quality depends on the appearance of the scene. Therefore, visual features are only used to complement scene knowledge. If tracking fails, the force measurements from the sensor are still available. Typical failure cases are transparent or

entirely textureless objects. Floor tracking is more reliable, since household floors are usually textured, and even plain concrete surfaces, as shown in Fig. 3.6, exhibit features from material wear or imperfections.

The motion model of the floor is a 2D translation and rotation of the platform on the (known) floor plane. Its parameters (speed, rotation) are estimated from the feature tracks to obtain a visual motion estimate, which complements the inertial unit (IMU) of the platform. Some feature tracks will be incorrect (outliers) and must be removed. Motion parameters are estimated using RANSAC [28], which finds the majority vote of features and is thus robust to outliers and local failures. Projection of the motion onto the floor plane yields the real-world motion of the platform, coherent to the sensor readings.

The object surface is located in the image area just above the foam (see Fig. 3.6). After foam deformation has reached its maximum, ideally the object, and thus the feature points on it, no longer move relative to the camera. Otherwise the object movement and especially its rotation are estimated using the motion of the features. Feature points are projected onto an estimate of the object surface. It is modeled as a plane that is perpendicular to the floor and is spanned by a line that approximates the deformation region. Since the exact contact points are known and due to the small relative object motion, errors from model deviations remain low. Object motion is determined from feature tracks by homography estimation and decomposition, see Sec. 2.3.3. Rotation on the floor plane and fall events are most relevant. Fall of an object usually occurs into the direction of the push and could be detected by fast rotation around the respective axis. However, tracking will not be able to follow such fast motion, and instead a sudden loss of track is taken to trigger a *fall* event.

Alternatively, a template-based tracker/detector, see Sec. 2.3.4, has been used in [3]. The object pose can be detected with great accuracy in 6D using this approach, and a lost object can be recovered. However, it is required to obtain a texture-based object model. This process is performed when the object comes into contact with the sensor. An object mask is generated using depth information from a Kinect camera. While the tracking and detection performance of this approach is superior to KLT, the need for model acquisition is infeasible for some setups. Obviously, a failure during model acquisition renders tracking and detection impossible. When the object is retrieved from a visual database, however, an accurate texture model is already available, and model-based trackers should be preferred over KLT.

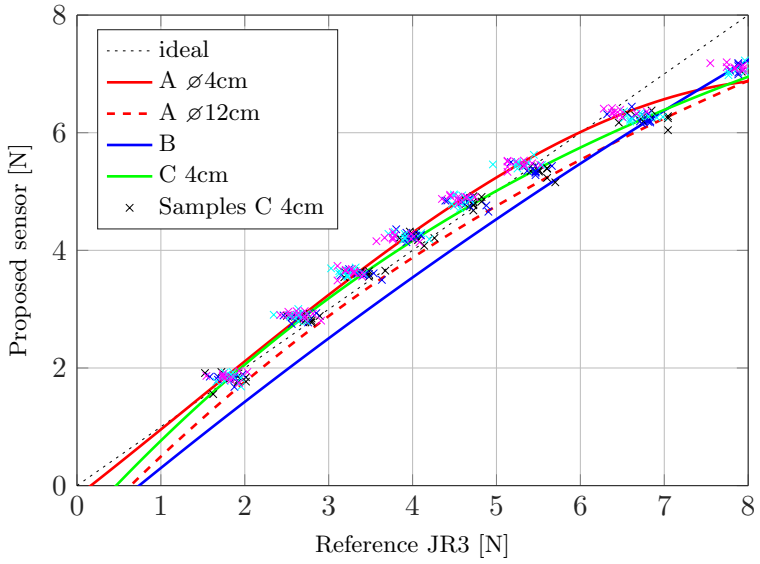
Obviously a number of additional techniques could be applied to acquire visual scene knowledge. A depth camera would allow for stable extraction of the floor surface using plane fitting, and the object geometry could be accurately acquired. However, coherence needs to be ensured, and sensors like the Kinect exhibit a minimum range of 0.5 m. Furthermore, tracking could be performed with multiple templates to increase stability and to reduce noise. A method for the selection of templates based on their estimated tracking performance has been presented in [1]. The proposed system renounces additional techniques and relies only on simple feature tracking, which showed to be effective. This also results in a lower computational load compared to processing of point clouds or depth maps.

3.6 Experiments

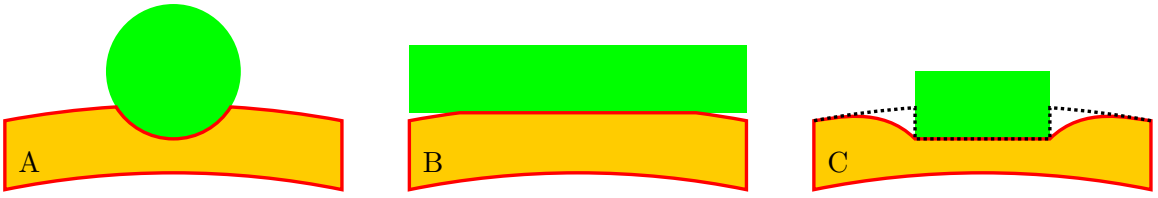
This section presents several experiments conducted with the proposed sensor mounted on a mobile platform, a robotic gripper and a robot arm. Results showing the measurement accuracy of both sensor types are presented in the following subsection.

3.6.1 Sensor accuracy

Foam-based sensor In order to determine the accuracy of the proposed visuo-haptic sensor, objects are pushed into the foam with increasing forces using a KUKA lightweight robot arm. The applied force is measured using a commercial, factory-calibrated JR3 force sensor, which serves as the reference. At the same time forces obtained from the proposed sensor with

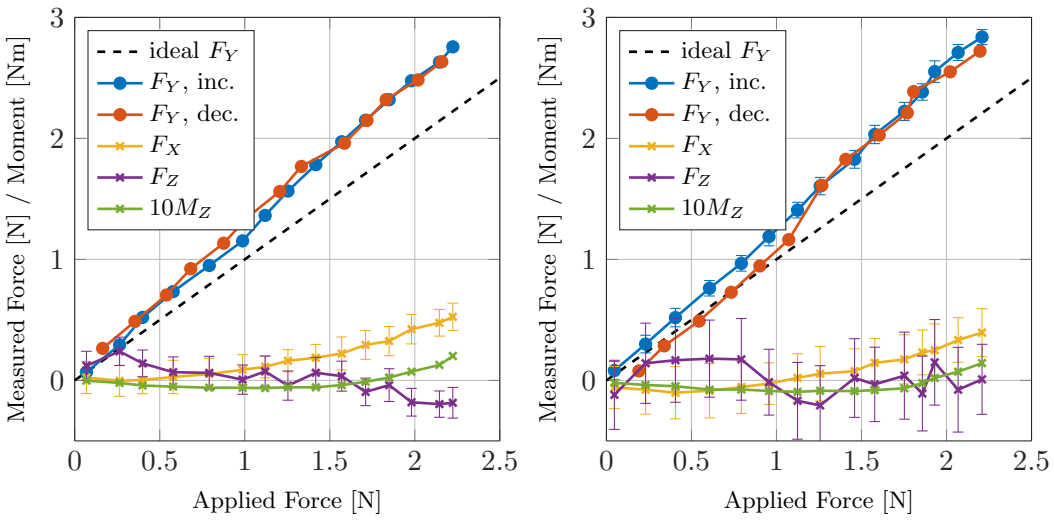


(a) Sensor accuracy



(b) Test shapes

Figure 3.7: *(a)* A comparison of the force measured by the proposed foam-based sensor against a reference obtained with a JR3 force sensor shows good accuracy. Sample points are shown for test shape C, using different colors for different runs. *(b)* Different shapes (green) are pressed into the foam (orange/red). The sensor contour does not follow the edge of object C (black dashed line) and instead shows a smoothed out impression (red). Figure adapted from [6].



(a) Camera distance: 25 cm **(b)** Camera distance: 50 cm

Figure 3.8: The accuracy of the beam-based sensor is tested by applying a force with a KUKA LWR arm along F_Y. Error bars indicate the standard deviation and are scaled by a factor of 5, respectively 2 for F_Z, for better visibility.

Eqn. (3.1) are recorded and plotted against the reference force in Fig. 3.7a. The experiment is repeated with different object shapes as depicted in Fig. 3.7b – two cylinders A with diameters 4 cm and 12 cm, a large plate B as well as one with 4 cm width (C). The pressure applied to the foam by the small plate has discontinuities at the edges and resembles a step function. The impression in the foam is smoothed out due to internal tension, see also Sec. 3.2.

A polynomial of order three is fitted through the obtained data points, yielding the characteristic curves in Fig. 3.7a. Ideally, the curves should follow the black dashed line – yet a small systematic error is observed, which depends on the object shape: Small forces are underestimated, especially for large objects, which may be attributed to the high slope of m_P for low strains. Large forces are underestimated for all object shapes. This error could be further reduced by a more accurate foam deformation model. Note that the curve for the small plate C is close to the ideal line, despite the fact that the foam impression extends significantly beyond the edges of this object. Thus, the modeling of such boundary effects, as discussed in Sec. 3.2, proves to be valid.

Individual data points for object C are depicted with a ×-symbol in Fig. 3.7a. The different colors represent four different experiments and show that the repeatability is within the value of the noise. The observed standard-deviation for the proposed sensor is $\sigma = 0.075\,\mathrm{N}$ along the entire range. This is lower than the value observed for the JR3 reference, $\sigma_{ref} = 0.15\,\mathrm{N}$. However, it must be noted that the JR3 sensor has a sampling rate of up to several kHz and might have been distorted by vibrations of the robot arm. Since the force in Eqn. (3.1) is calculated from multiple points in the image, there is an inherent smoothing effect.

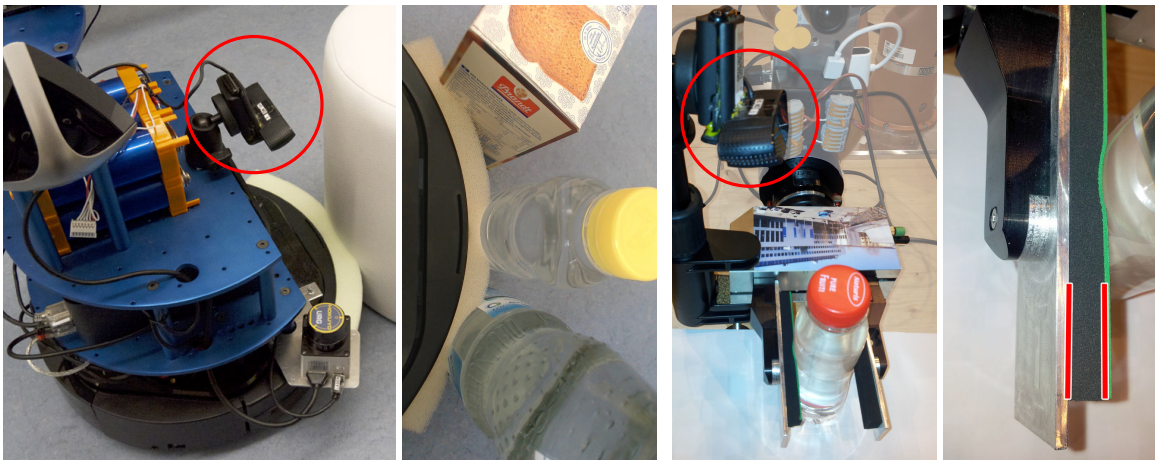
Beam-based sensor In a similar experiment, which was conducted together with [29], the accuracy of the beam-based sensor is analyzed. The mechanics of the sensor are built according to Fig. 3.5, whereby the spring steel rod between (S) and (F) exhibits a length of 12 cm and a diameter of 1.5 mm. Its material parameters for Eqn. (3.4) are $E = 206\,\mathrm{GPa}$, $I = {}^1\!/\!_4\pi r^4 = 2.48 \cdot 10^{-13}\,\mathrm{m}^4$. One template of 4 cm side length is attached to the frame as a reference, and two more are glued to the rod, each at a single point. The frame is fixed at point (B), and the LWR arm pushes against the sensor tip (F) along F_Y. The applied force is increased stepwise from zero to a maximum value, and then again decreased in the same fashion. Force and position are recorded using arm sensors as a reference. A webcam with $1600 \times 896\,\mathrm{px}$ observes the templates from above (along $-Z$) and determines 6D force-torque values according to Eqn. (3.5).

The obtained results are shown in Fig. 3.8. Since the force is applied only along F_Y, it is expected that all other force and torque components remain zero. Indeed, crosstalk to other components is small. The moment M_Z, which is mathematically strongly coupled with F_Y, deviates from zero only for large forces. The force F_X exhibits a low level of coupling, since deformations along Y also shorten the rod along the X-direction. This effect is currently not modeled. Also, a small systematic error, i.e. a deviation from the ideal line, is observed. This may be attributed to incorrect material parameters and to the approximations used in beam theory. A compensation of this error is straight-forward using a calibration process. Contrary to foam-based sensors, there is no hysteresis effect when increasing/decreasing the force. As expected, sensor noise increases with increasing camera distance. However, the noise level is very low, except for the F_Z component. In order to increase accuracy for F_Z – which should be done for larger distances – either a depth/stereo camera and tracker could be used, or observations from two cameras at different locations could be combined.

3.6.2 Exploration of objects with a mobile platform

A mobile platform equipped with the proposed visuo-haptic sensor, see Fig. 3.9a, is driven towards several different obstacles in a room, such as boxes, bottles, tables, doors and walls. Contact with an obstacle is detected when the foam rod starts to deform. The speed of the platform is reduced based on the visual proximity detector to avoid damage to the object. The movement is stopped completely if one of the following conditions becomes true: *(a)* The amount of deformation goes beyond an upper limit, i.e. the strain goes to the *densification* region, *(b)* the total force, see Eqn. (3.1), reaches the pushing capabilities of the robot, *(c)* the robot moves for a distance larger than the width of the sensor. In the latter case, a movable object is observed, and the measured force corresponds to the friction force of the object. In the other two cases, the explored object is fixed – considering the capabilities of the platform.

For both kinds of objects, fitting of geometric primitives (Sec. 3.3.3) is triggered during the halt of the platform. Measurements are most stable in the static case and allow for the best possible fit. Examples of some explored objects are shown in Fig. 3.10, together with the



(a) Platform sensor **(b)** Gripper sensor

Figure 3.9: Foam-based visuo-haptic sensors mounted on a mobile platform *(a)* and a two-finger gripper *(b)*. A webcam is pointed onto the passive plastic or rubber foam mounted on the actuator or platform. The respective camera views (depicted) show the foam and its surrounding. In *(b)*, red lines indicate the locations of the reference (left) and front (right) contours. Figure adapted from [6, 7].

fitted geometric primitive. The correct primitives are fitted with high accuracy in Fig. 3.10 (a)-(d) and (f). The bottles in (d) and (e) are (partially) transparent and could thus not be detected by purely visual methods. The fitted circle primitives have the correct size for (d), but are slightly under- or overestimated for the two bottles in (e). This may be attributed to an imperfect fit of the primitive at the boundaries of the deformation region.

3.6.3 Naïve haptic mapping with a mobile platform

In this experiment, a map of an office environment is created by haptic exploration with a mobile platform. Location information is obtained from the robot odometry and an IMU in the platform. In larger workspaces, cues from a visual localization system should be added to reduce drift. A naïve exploration strategy is used, having the platform drive along the circumference of the room: Whenever an obstacle is hit, the robot is stopped, moves back, rotates counterclockwise to drive in parallel to the obstacle for a while, rotates clockwise and drives straight again. As described above, it is determined whether the obstacle is movable or static by pushing it, and its geometric primitive is stored.

Haptic maps showing static and movable objects are generated during exploration and updated during each contact event. The maps show the confidence score of occupancy, where -1 corresponds to free space and 1 corresponds to occupied space. At each contact event, the stored geometric primitive is added to the map as follows: The binary occupancy map $m_{\mathcal{O}}$ is determined for the primitive – e.g. the area inside a circle is marked as occupied (1), and

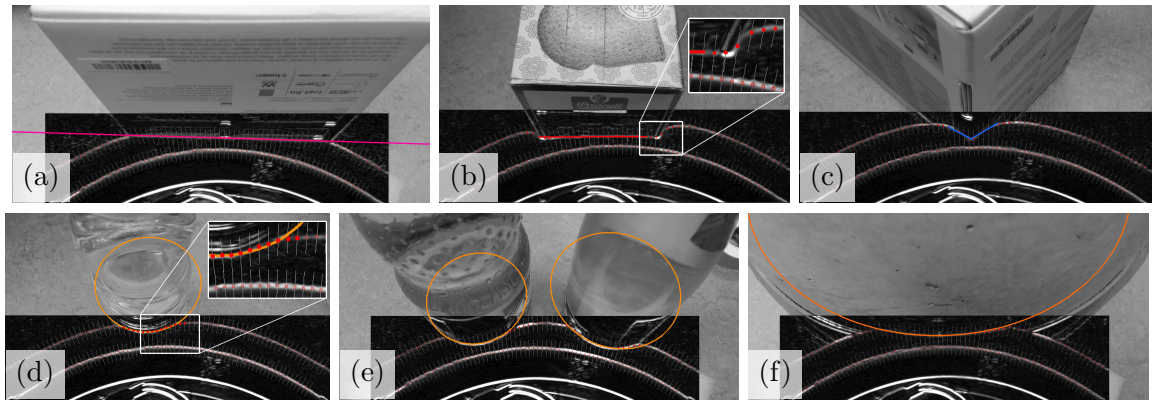


Figure 3.10: View from the platform camera while touching different objects. Pictures are split, showing image gradients in the lower half, and the original image in the upper half. Additionally, the fitted geometric primitives are depicted: *(a)* line, *(b)* line segment, *(c)* corner, *(d-f)* circles. Figure adapted from [6].

the outer parts are marked unoccupied (-1). Beyond the contact points, primitives represent just predictions of the environment, which become more inaccurate with increasing distance from the closest contact point. This is modeled as a normalized distance map $m_d \in [0, 1]$, which shows the confidence of the geometric primitive. Normalization is performed based on the extent of the contact area. Like that, a primitive – such as a line – can predict a larger geometry, yet the prediction is quickly updated once a more confident measurement is available. The current observation is integrated into the global map $\mathcal{M}$ as follows:

$$\mathcal{M} \leftarrow (1 - c\mathcal{T}(m_d)) \cdot \mathcal{M} + c\mathcal{T}(m_{\mathcal{O}} m_d), \tag{3.12}$$

$$m_d(x, y) = \exp\left(-\frac{1}{D} \min_k ||(x, y, 0) - \mathbf{X}_k^{(F)}||\right)$$

The factor c determines how quickly old measurements are replaced and is set to 0.5 here. Transformation $\mathcal{T}$ aligns the new observation with the map frame, using the current pose of the platform. The distance map m_d is calculated from the distance to the closet contact point $\mathbf{X}_k^{(F)}$, normalized by the diameter D of the smallest-circle around the contact area.

Results are shown for a small office space in Fig. 3.11, which was explored using 22 contact events. The structure of the room with the walls, column and door is accurately represented, even though some shapes extend beyond the room. One of the reachable chair legs appears as a small blob left of the paper bin. The discovered movable objects (paper bin near the right wall and two bottles in the left part) are depicted in blue. The bottle in the upper left area was explored twice and moved in between by the platform. Erroneously, a corner primitive rather than a circle is fitted, as the object is small. Note that visual mapping systems would easily overlook this bottle, since it is completely transparent, apart from the cap.

Figure 3.11: An office scene is mapped by naïve haptic exploration and fitting of geometric primitives. Occupied areas are depicted in red, free areas in green and movable objects in blue. Color intensity corresponds to mapping confidence. Figure adapted from [5].

3.6.4 Grasping with the foam-based sensor

In this experiment, the foam-based sensor is evaluated in a grasping task: A two-finger gripper equipped with the sensor, see Fig. 3.9b, is mounted to a KUKA LWR arm and grasps a deformable transparent object. Two rubber foam rods with a cross section of $1.5 \times 1\,\mathrm{cm}$ are attached to the fingers, and a single camera is mounted above the gripper to track one of them. Rubber foam is harder than most PUR foams, providing a force range of up to 100 N, see also Fig. 3.2. The system relies solely on visual data, therefore the dedicated position and force sensors in the gripper and in the arm are not used. Initialization is performed using a picture as a reference template on the gripper.

First, the stiffness and shape of a typical household object, a plastic bottle, is explored, see Fig. 3.9b. While the gripper is closing around the bottle at half height h_2, the deformation and applied pressure are recorded. Fig. 3.12 shows the deformed shape of the bottle determined from the foam impression at five different grasping forces. With increasing pressure (left to right), the entire surface retreats, the curvature decreases slightly and the contact region increases. The experiment is repeated at different height levels of the bottle by moving the gripper with the robot arm (haptic exploration). Stiffness of a thin-walled object exhibits a high dependency on the local geometry, see also Chapter 6. Fig. 3.13a shows that the object is very stiff near the bottom and the top. These are both convex regions, which provide most support. However, it can also be seen that there is a knee in the curves for $h_{3,4}$: Here, the surface bends to the inside and the support from the convex geometry is suddenly lost. Stiffness is low and almost linear near the center.

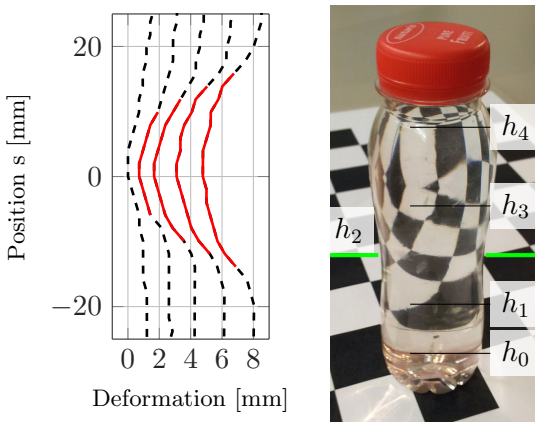


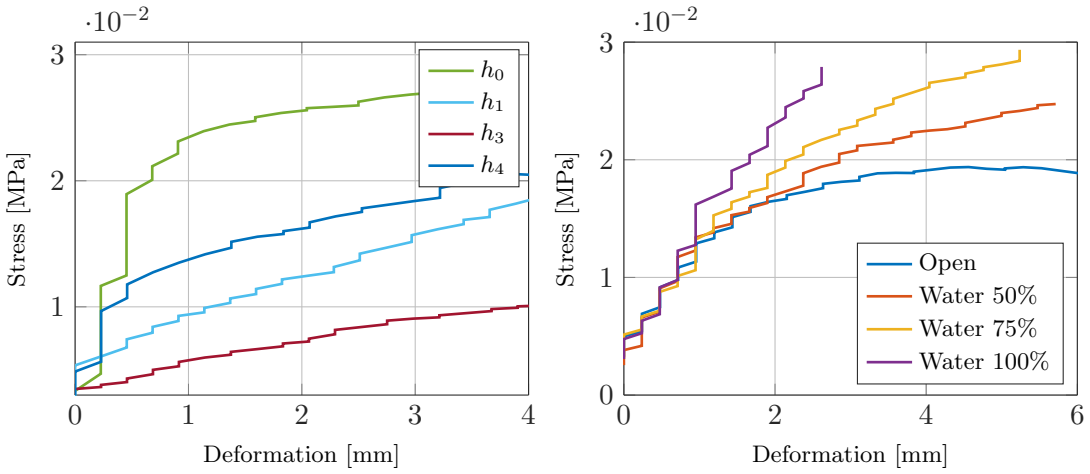
Figure 3.12: A transparent bottle *(right)* with a slight narrowing in the center (at height h_2) is explored by a gripper equipped with the foam-based sensor. The deformation of the front contour is shown for five instants of time in a fixed coordinate frame *(left)*. Force increases from left to right. The detected contact area is shown in red. Figure adapted from [7].

Second, stiffness is measured with different levels of water inside the (closed) bottle. A closed bottle exhibits a tendency to preserve its volume – on compression, enclosed air applies an increasing counterpressure onto the inside surface (see also Sec. 6.1.5). Liquids like water prevent any volume change, such that compression must be compensated by an expansion in other areas. Fig. 3.13b shows the deformation-stress relations in the center of the contact region. They are equal for small deformations, since volume preservation is not dominant here. For large deformations, the bottle with 100% water is the stiffest, since volume changes are prevented by the liquid.

3.6.5 Manipulation with the beam-based sensor

Opening a crown cap, as used on many beverage bottles, is a good example for a manipulation operation that relies on joint visuo-haptic perception. For this experiment, a standard bottle opener is attached to a beam-based sensor according to Fig. 3.5. The sensor is mounted on the end-effector of a KUKA LWR arm, together with a webcam, see Fig. 3.14. Beam deformations caused by forces or torques applied to the crown cap are observed using the template pictures. Furthermore, the same camera observes the scene near the tool in order to search for the bottle. The visuo-haptic sensor is the only sensor used in this system – the dedicates force-torque sensors of the robot arm are not used.

Bottles are recognized and tracked using the texture on their crown caps with the detector/tracker from Sec. 2.3.4. After the user has selected the desired bottle, the robot starts searching for it within its working space. Once the object has been detected and localized, the arm moves the tool above the object and rotates it as required for the uncapping operation, see Fig. 3.14b. The accuracy of this rotation movement is limited by two factors: First,



(a) Varying height **(b)** Varying contents (@h_2)

Figure 3.13: The stiffness of the transparent bottle (Fig. 3.12) is measured with the proposed sensor. Stiffness varies *(a)* along the height of the object, as well as *(b)* with changing counterpressure from varying contents. Figure adapted from [7].

since the wrist joint of the arm is relatively far from the tool, a large motion is required in the joint space. This goes along with a larger positioning error (which could, however, be compensated by visual servoing). Second, the tracker has only a limited depth accuracy, such that the height of the crown cap cannot be determined exactly. Yet, the alignment of the tool on the crown cap must be very accurate in all three dimensions in order to perform the uncapping successfully. This level of precision is unattainable using pure vision, especially in the depth direction. The exact position of the opener is determined haptically, by measuring the forces or moments with the visuo-haptic sensor. Refinement is performed separately along the Z-axis (height), the X-axis and the Y-axis. Once the tool is aligned and locked under the cap, the arm rotates the tool around its center point for uncapping. If the crown cap has been removed successfully, the tracker/detector no longer detects its texture on top of the bottle. The control sequence for this experiment is summarized in Table 3.1.

3.7 Summary

In this chapter, a novel concept for visuo-haptic sensors has been presented. It is based on a passive deformable element with known deformation characteristics that is mounted onto the actuator or even integrated in the actuator itself. A camera is placed at an arbitrary location with a good view of the deformable element and measures the deformation of this element from the image. The camera can be used simultaneously for many additional computer vision tasks, such as tracking and object detection. For many robot setups, an already existing camera can be used, such as an in-hand camera or a camera in the robot's head. Depending

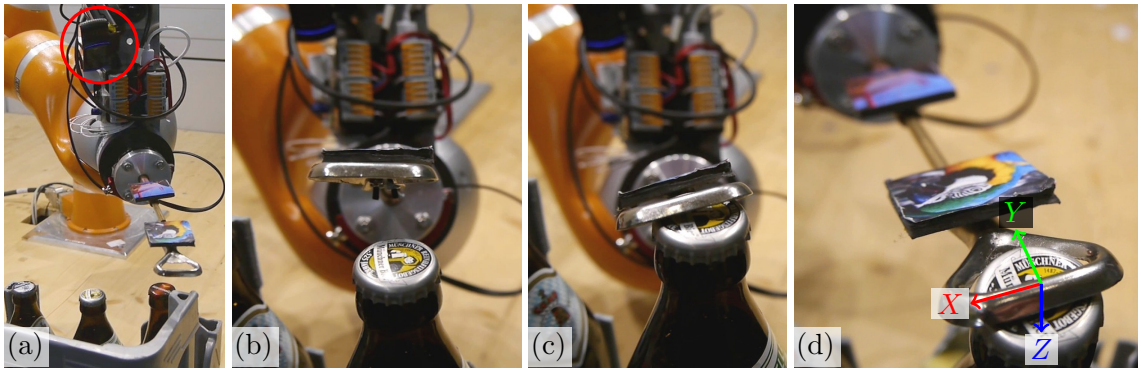


Figure 3.14: Opening a crown cap with a bottle opener mounted on a beam-based visuo-haptic sensor, see Table 3.1. The camera is indicated by the red circle in *(a)* and observes the tool as well as the bottles. The coordinate system of the tool is depicted in *(d)*. In *(c)*, the tool is aligned incorrectly along the X-axis, necessitating a refinement step.

Input	**Task / *Goal***	**Location certainty** X	Y	Z	**Dist.**
—	Initialization	?	?	?	$< 1m$
SIFT features (V)	(a) Scanning motion, SIFT detection *Find object*	1 cm	1 cm	> 1 cm	ca. 20 cm
Template (V)	Template tracking *Track object*	1 mm	1 mm	< 1 cm	
—	Reorient tool, approach object *Tool is near object*	> 1 mm	> 1 mm	1 cm	> 1 cm
F_z (H)	(b) Push down tool (along Z) $F_z > const.$			1 mm	> 1 mm
M_y (H)	(c) Sidewards correction (along X) $M_y \approx 0$	1 mm			1 mm
F_y (H)	Forward refinement (along Y) $F_y > const.$		< 1 mm < 1 mm		0 mm
M_x (H)	(d) Rotate tool, open crown cap $M_x > const.$				
M_x (H) Template (V)	Check for crown cap removal $M_X \approx 0$, *template not on bottle*	?	?	?	1 cm

Table 3.1: Sequence of tasks for opening a crown cap, executed from top to bottom. The location certainty between the tool and the object (the crown cap) is successively improved while decreasing the distance. Tasks rely either on visual (V) input or haptic (H) input, i.e. the force F and moment M obtained from the camera-based sensor. Letters (a) to (d) refer to Fig. 3.14.

on the shape and characteristics of the deformable element, different force readings can be calculated from the image. An implementation based on a plastic or rubber foam is presented for mobile and grippers. It measures a force and contact shape profile along the contour of the foam material. Furthermore, an implementation based on spring steel is presented, which measures a 6D force-torque at a single point – such as between a robot arm and a tool. In experiments, the accuracy of this sensing principle is analyzed.

Compared to existing tactile sensors, this concept offers a number of advantages: The sensitive element itself is passive, which reduces system complexity and requirements for cabling on the robot. The passive element is very low-cost, and replacement in case of wear is straight-forward. A standard consumer webcam is used in the presented implementations. Due to the large proliferation of such devices, they are low-cost and offer a good image quality. As visual and haptic measurements come from the same sensor, they are naturally coherent, i.e. they are synchronous in time and space. Moreover, a failure in one modality can be compensated by the other one – e.g. a transparent object is not directly seen by cameras, but can be detected in the haptic modality based on its contact shape. On the other hand, very light objects do not exert enough force to be detected haptically, but cameras observe their presence. Multiple sensitive elements, as well as very large areas can be monitored – which would be very costly with existing active sensors.

Optical readout and camera-like sensors are used on many existing tactile or haptic sensors, see Sec. 2.1.1. Yet, existing setups integrate the camera into the sensor module, together with the sensitive element. Like that, those designs do not offer any of the advantages given here.

4 Joint planning of navigation and manipulation

Navigation is an essential skill for mobile robots, which is closely coupled with environment perception. On an abstract level, robot navigation is modeled similar as in automotive navigation systems: The structure of buildings is represented as a navigation graph, with nodes corresponding to decision points between different routes, see Sec. 2.3.6. This graph is used for path finding, even in complex environments, estimation of path costs and topological decisions, such as whether to drive left or right around an obstacle. A path in this "global" topological graph serves as the goal trajectory for low-level planning, which considers current sensor observations to avoid collisions. Platform motion commands are generated in realtime, considering acceleration and speed limits. The local trajectory is matched against the global plan, allowing for a certain degree of deviation. On both levels, occupancy grid maps are used to model free and occupied space – yet, for global planning, observations from different viewpoints are fused.

There are a number of manipulation tasks, which are directly connected to navigation, such as opening doors, using an elevator or pushing an obstacle aside. These tasks require haptic models of obstacles as well as haptic perception. Therefore, a method for joint navigation and manipulation planning is proposed here. It combines visually acquired maps from a depth camera or a laser scanner with haptic data acquired from the sensor proposed in Chapter 3. The common representation uses a high-level model based on a graph, which extends navigation graphs provided by vision-based methods. Obstacles that may be removed by manipulation operations, such as pushing an object aside, are represented as additional manipulation nodes. Paths via these nodes come with additional costs and require manipulation planning. In a vision-only model, there are blockades at the locations of these nodes. For instance, a paper bin in front of a doorway is detected as a blockade. With haptic data about this obstacle, it can be removed if access to the doorway is required. Using standard approaches for path finding, an integrated navigation and manipulation plan is generated. Compared to purely visual path planning, shorter paths involving manipulation operations are possible, and task plans can be generated to access parts of a room that seems inaccessible using vision-only perception.

This chapter is partly taken from [6].

4.1 Haptic tags

Modeling detailed haptic properties of objects – such as friction, deformability or roughness – can be very complex, especially since properties generally change along the surface. Haptic exploration of an object with a robot arm is a time-intense process (see Chapter 5 and Chapter 6), which is infeasible for mobile robot platforms as considered here. Instead of an accurate object model, a robot platform requires a prediction of object behavior for planning of simple and standardized manipulation tasks. For these tasks, a detailed haptic object model is not required – it is sufficient to know some global or semi-global object properties. In case of a simple standardized grasp, these parameters could be weight as well as the local friction, deformability at the contact region and the contact shape. Planning of more complex grasp patterns, taking into consideration the local stiffness of objects, is discussed in Sec. 6.2. For pushing manipulations, which represent the primary use of haptic tags, friction forces, object stability and potentially inertia are of greatest relevance. To some extent, these values depend on the actuator, the contact point and the manipulation task at hand. We refer to these properties as haptic tags.

Haptic tags are acquired using the visuo-haptic sensor presented in Chapter 3 while the platform slowly moves towards the object, touches and then pushes it a bit, if possible. In addition to the force profile derived from foam deformation, the camera acquires visual cues from the scene to determine the ego-motion and the reaction of the object on the pushing force, see Sec. 3.5. A simplified haptic tag, which does not include motion parameters, is acquired during grasping processes. The haptic tag consists of the following properties:

- Static friction force $H_F[N]$
- Dynamic friction $H_{F'}[N]$
- Deceleration time $H_I[s]$
- Sidewards drift $H_S[\frac{m}{m}]$
- Deformability $H_D[\frac{m}{m}]$
- Rotation on the floor plane $H_R[\frac{rad}{m}]$
- Deformation shape*
- Fitted geometric primitive*, see Sec. 3.3.3
- Events Fixed/Fall
- Counters for Exploration/Failure

These properties allow for the planning of manipulation tasks and prediction of object behaviour during manipulation. Properties marked with a star (*) can only be acquired with the foam-based sensor. The manipulability is determined from the events “fixed/fall”, and the friction force serves as an estimate for the required effort. If the manipulator cannot control

lateral object motion, drift and rotation pose limits to the possible pushing distance. The expected reaction of an object is expressed by its static friction, deformability, drift, rotation and the "fall" event. Furthermore, the deformability allows for an estimate of the material type, which is important, for instance, to adjust grasping parameters. Simple articulated objects – e.g. a wing of a door with hinges – are modeled based on rotation and deceleration time. In the following, the haptic tag is used for planning of pushing operations, see Sec. 4.2.1.

Acquisition of a haptic tag during a pushing manipulation is separated into different phases, triggered by events: Event *contact* – phase *(a)* – event *motion* – *(b)* – platform stops – *(c)*. If an event does not trigger, the corresponding phase is skipped, and some haptic properties remain undefined. The visuo-haptic sensor measures distances between the robot, object and the environment (floor), see Sec. 3.5. As soon as the foam starts to deform, i.e. δ increases, the event *contact* is triggered. Once the deformation remains constant (or decreases) while the platform continues to move, the event *motion* is triggered. Data are stored during exploration and later processed separately per phase.

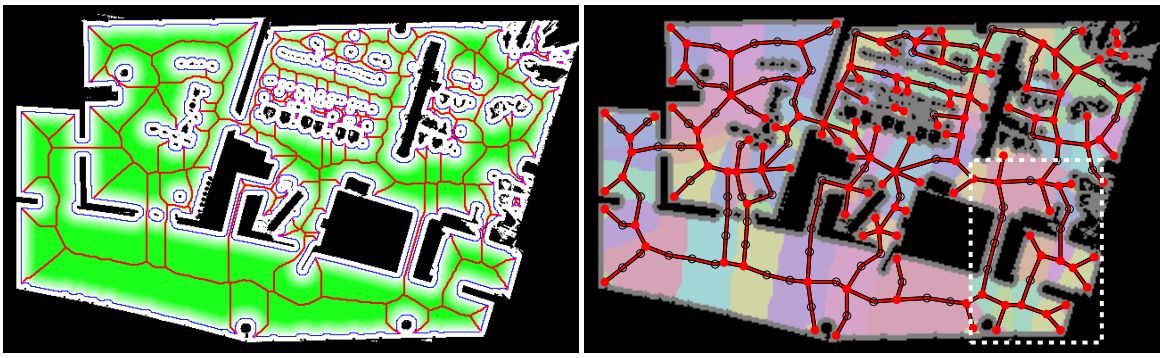
Phase *(a)* — The static friction is the maximum force value measured during this phase. Furthermore, the object deformation is determined by $H_D = 1 - \frac{\text{foam deformation}}{\text{platform motion}}$. For rigid objects, the two distances are equal and $H_D = 0$. Distance and pressure values are taken when they are largest, and a linear compression model is assumed, thus $H_D = const$. The relative measure H_D may be converted to a stiffness parameter $[\frac{N}{m}]$ by considering the pushing force $H'_{F'}$ and the absolute deformation. Objects with a negligible static friction force, such as open wings of doors or balls, trigger the *motion* event immediately, skipping this phase.

(b) — Dynamic friction, drift and rotation are acquired. The dynamic friction is the average force while the object is moving. Drift H_S is the shift of the central contact point perpendicular to the pushing vector, normalized by the pushing distance. Object rotations indicate instability during the manipulation and are estimated using feature points tracked on the surface, see Sec. 3.5. Rotation on the floor plane is comparable to the drift, while rotation around the y-axis of the camera image indicates that the object falls over.

(c) — For most household objects friction forces dominate and their inertia cannot be directly observed. In this case, after the platform stops, the measured force remains constant, and $H_I = 0$. Only some objects show an observable deceleration behavior and continue their motion after stopping. The duration of continued motion is taken as an estimate for deceleration. Object motion is determined from the change of foam deformation and by feature tracking. For instance, the wing of a door usually continues to move after being pushed, detaching from the foam to finally stop somewhere in front of the platform, due to small friction in the door hinge.

Finally, the two counters are used for an estimate of the success probability, when the object is manipulated several times. A failure is triggered when fast feature motion is detected, e.g. when the object falls or is destroyed.

Haptic tags are assignable to different visual representations of the object: Most importantly, they are assigned to the footprint of the object in the map at the contact point of



(a) Skeleton (red), distance map $\mathcal{M}_d$ (green) **(b)** Navigation graph

Figure 4.1: Skeleton, distance map *(a)* and navigation graph *(b)* extracted from a map of several rooms and a hallway. Additional connection nodes are encircled in black. Regions assigned to a navigation node are depicted in pale colors. The detail marked by the dashed white box is used in following figures. Figure adapted from [6]. Map adapted from [46].

exploration. This means that haptic tags stick to objects as long as they are not moved, even when the robot is in a different part of the building. Furthermore, haptic tags can be associated to object identities in an object database used for visual search with feature descriptors, see Sec. 2.3.5. That way, if a known and explored object is recognized visually anywhere in the room, its haptic tag can be used for manipulation planning without further exploration. This idea can be extended such that haptic tags are loosely associated to visual classes of objects, such as "bottles". When an object instance is detected, the haptic tag allows for an estimate of its manipulation properties – see also Sec. 6.3.3.

4.2 Graph-based planner

Topological navigation graphs model navigable space within buildings for a mobile platform in a very efficient way. Edges represent all path segments that are accessible by the platform, such as hallways, doors and any free space between obstacles. Nodes are decision points that connect these segments, but they can also be seen as a representative for a certain region in the map. Modeling is limited to 2D planes for (piece-wise) planar indoor scenes, even though topological graphs could also represent height. In the following, it is described how a navigation graph is obtained from an occupancy map, making use of established techniques, which are reviewed in Sec. 2.3.6. The platform exhibits a laser scanner, which scans a range of 240° in a plane parallel to the floor. The Karto Visual SLAM system [50] fuses range data to build a 2D map of the entire operation space and localize the robot therein. Additionally, an inertial unit (IMU) provides a motion prior. A graph-based representation is obtained from the occupancy grid map using the following steps: (*a*) Determination of a trivalent

state (occupied/free/unknown) for each cell; (b) generation of a distance map $\mathcal{M}_d$ that shows the distance to the nearest occupied cell (green in Fig. 4.1a); (c) calculation of a cost map $\mathcal{M}_c$ that penalizes proximity to obstacles, see below; (d) thinning of the reachable map area, i.e. $\mathcal{M}_c < \infty$, yields a skeleton (red paths in Fig. 4.1a); (e) conversion of the skeleton to a graph with edge costs $d \cdot \mathcal{M}_c$, edge length d. The costs $\mathcal{M}_c$ of cells represent the inverse of the possible speed of the platform and are infinite for unreachable cells:

$$\mathcal{M}_c = \begin{cases} \infty & \mathcal{M}_d < r_{rob} \\ v_{max}^{-1} & \mathcal{M}_d > 3r_{rob} \\ [v_{min}^{-1} \dots v_{max}^{-1}] & \text{otherwise} \end{cases} \tag{4.1}$$

The obtained topological graph depicts the structure of the room and is further simplified: Short stub paths introduced by the skeleton are removed. Nodes with exactly two edges are connection nodes that can be deleted, except in cases where their removal would also remove a loop. Corresponding edges are merged together, summing their costs. The remaining nodes are located on the intersection points of the skeleton, see Fig. 4.1b. For visualization purposes, some connection nodes are kept at regular intervals, such as 1 m. The edge costs are based on skeleton segments in the center of the room and are thus in general overestimated. The platform motion is fastest and most reliable along such paths, yet there will be shorter paths in most cases. For a global planner, however, this approximation is feasible. The graph is a much more efficient representation than a grid map, since it only exhibits one node for each choice around obstacles.

A mapping between occupancy grid cells and nodes of the graph (and vice versa) is required to map a path in the graph to regions in the grid map. The assignment should be based on the shortest path between grid cells and nodes. It must take the cost map $\mathcal{M}_c$ into account and especially the fact that obstacles may block access to free space from a close-by node. Each node is considered a representative of its assigned map segment, as depicted in Fig. 4.1b. The assignment is performed with Algorithm 1. A fast-marching approach [32] with multiple start positions is utilized. The start positions correspond to the node locations and represent multiple minima in the potential field $\mathcal{M}_P$. Furthermore, a segment map $\mathcal{M}_R$ is obtained, which assigns each grid cell to a node id (pale colors in Fig. 4.1b).

The potential map $\mathcal{M}_P$ also allows for fast shortest-path planning from any point in the map to the closest node, simply by iteratively moving to the neighbor cell with the lowest potential. This path is used by the local planner to reach the final goal.

4.2.1 Integration of haptic information

Occupancy grid maps and the extracted graph are solely based on visual data. Any visible obstacle is considered to definitely block platform movement, regardless of its weight and shape. Using haptic knowledge about obstacles – specifically whether an object is movable

Algorithm 1: Region assignment based on the fast marching method

Data: Navigation graph $\mathcal{G}$, cost map $\mathcal{M}_c$
Result: Map of region assignments $\mathcal{M}_R$, potential map $\mathcal{M}_P$

Initialize: $\mathcal{M}_P \leftarrow \infty, \mathcal{M}_R \leftarrow 0, U \leftarrow \emptyset$;
for $i \in nodes(\mathcal{G})$ **do**
    $\mathcal{M}_P(x_i) \leftarrow 0$; // x_i: Location of node i
    $\mathcal{M}_R(x_i) \leftarrow i$;
    $U \leftarrow U \cup \{x_i\}$; // Update set
repeat
    for $u \in U$ **do**
        $U \leftarrow U \setminus u$;
        for $n \in \mathcal{N}(u)$; // Neighbour cells of u
        do
            $q = \mathcal{M}_P(u) + \mathcal{M}_c(n) \cdot d(n, u)$;
            if $q < \mathcal{M}_P(n)$ **then**
                $\mathcal{M}_P(n) \leftarrow q$;
                $\mathcal{M}_R(n) \leftarrow \mathcal{M}_R(u)$;
                $U \leftarrow U \cup \{n\}$;
until $U = \emptyset$;
return $\mathcal{M}_R, \mathcal{M}_P$

by the robot platform – additional paths involving manipulation of obstacles may become available. For instance, a toy ball or lightweight paper bin blocking a passageway can be easily pushed away. Such movable obstacles are represented as a new type of nodes, the "manipulation nodes", associated with a manipulation operation and connected to neighboring nodes. Haptic tags (Sec. 4.1) are stored in these nodes to estimate manipulation parameters. The visual navigation graph is extended but not modified to keep a consistent representation. Now, a path in the extended visuo-haptic graph translates to both navigation and manipulation tasks, which are mainly pushing operations on a mobile platform, see Sec. 4.3. This extended graph represents the environment on a more semantic level.

For the following process, a list of explored objects with haptic tags is assumed to be known together with an assigned region in the map, see Fig. 4.2a. Sec. 4.1 discusses the properties in the haptic tag. The list of haptic tags is acquired as discussed in Sec. 4.4, which presents an exploration plan for obstacles. The visual graph is extended to a semantic visuo-haptic graph as follows:

1. Set the occupied grid cells of the explored object o_e to "free" in the original map
2. Using the updated map, recalculate the cost map $\mathcal{M}_c$ in the vicinity of o_e according to Eqn. (4.1)
3. Add a node n_e to the navigation graph $\mathcal{G}$, located at the center of obstacle region o_e

4. In region map $\mathcal{M}_R$, assign all unassigned cells connected to n_e and with $\mathcal{M}_c < \infty$ to n_e (yellow in Fig. 4.2b)
5. Determine all neighbor regions of n_e in $\mathcal{M}_R$ and add edges between the corresponding nodes and n_e
6. Calculate edge costs between node n_e and its neighboring nodes based on the updated cost map $\mathcal{M}_c$
7. Remove those edges again that do not significantly reduce any path costs

The extended graph provide a time estimate for navigation operations. The costs for a manipulation operation t_{mani} are expressed consistently, taking into account the following components:

(a) Platform-specific constant setup/finishing time,

(b) manipulability given the current plan,

(c) duration of object motion t_{push} based on the required effort and push length,

(d) penalty for possible manipulation failures t_{fail}.

The calculation of costs is based on the haptic tag. The manipulability (b) is a binary decision based on events "fixed/fall" and lateral motion. The latter depends on the required push length and the estimated drift (H_S·push length) and must not exceed a certain threshold. Failure probabilities are derived from the exploration counter in the haptic tag if the object was explored several times. The pushing speed and length yield t_{push}. Speed is adapted to the required effort determined by the friction force and depends on the platform capabilities. Any path via a manipulation node includes two edges connected to this node, so costs $\frac{1}{2}t_{mani}$ are added to all its edges. The time for approaching and leaving the movable object is already expressed in the edge costs calculated in step 6.

A path in the extended graph $\mathcal{G}^H$ between the current position and a goal position can be planned as usual, e.g. by the $A^\star$ algorithm [41]. Alternative paths can be determined based on task-specific side conditions, such as most information gain about the scene, most reliable path or avoidance of manipulation tasks. Each navigation node within a path represents a high-level navigation task, e.g. "go left/right around obstacle" or "enter hallway". A corresponding initialization is given to the local planner, which searches a feasible and short path within the regions associated to the nodes in $\mathcal{M}_R$. Each manipulation node represents a task to push an object out of the way. The local planner for this task is described in Sec. 4.3.1 and attempts to move the obstacle such that it no longer blocks the robot. After the manipulation the scene map is updated, and $\mathcal{G}^H$ must be adapted to that change by converting the manipulation node to a navigation node and removing costs $\frac{1}{2}t_{mani}$ from the associated edges. From time to time, when the robot rests, the full process for graph generation is restarted. Local planners may fail – especially for the manipulation tasks – necessitating replanning on the global level, with costs set to ∞ for edges associated to the failure.

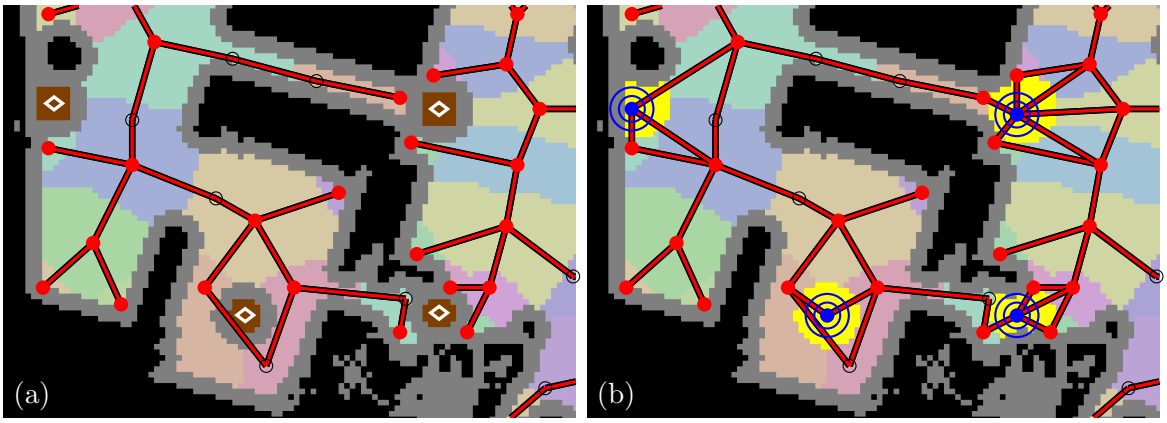


Figure 4.2: *(a)* Four movable obstacles (brown/⋄-symbol) are added to the map. The two on the right block access to passages – note how the connectivity in the graph is reduced compared to Fig. 4.1b. *(b)* The extended visuo-haptic semantic graph has additional manipulation nodes (blue circles) with corresponding regions assigned (yellow). Access to the hallways is again possible through these nodes. Figure adapted from [6].

The benefit of a new manipulation node n_i is determined by the maximum reduction of path costs between any of the adjacent nodes $\mathcal{N}(n_i)$. This value is used to determine the order of exploration, see Sec. 4.4:

$$\Delta t = \max \{\text{costs}_{\mathcal{G}^H}(n_k, n_l) - \mathit{costs}_{\mathcal{G}}(n_k, n_l) \quad \forall n_k, n_l | n_k, n_l \in \mathcal{N}(n_i)\} \tag{4.2}$$

4.3 Local planning

Local planners perform low-level platform control for a specific task. Usually, they must fulfill realtime constraints (e.g. to generate motion commands at a rate of 50 Hz) and therefore exhibit only a limited level of intelligence. Generally, they ignore topology or other global information and instead rely on current sensor readings, potentially a local costmap, as well as information provided by the global planner. Each node in the navigation and manipulation graph is associated with a local planner. The planner is activated, configured, cancelled and stopped by the global planner as soon as the robot gets into the region associated with the node. The result of the local task – which is success, failure or timeout – is returned to the global planner. The local planner must provide a rough estimate of execution time to the global planner. Navigation tasks are performed by a standard local planner [68]. This section discusses local planners for manipulation tasks.

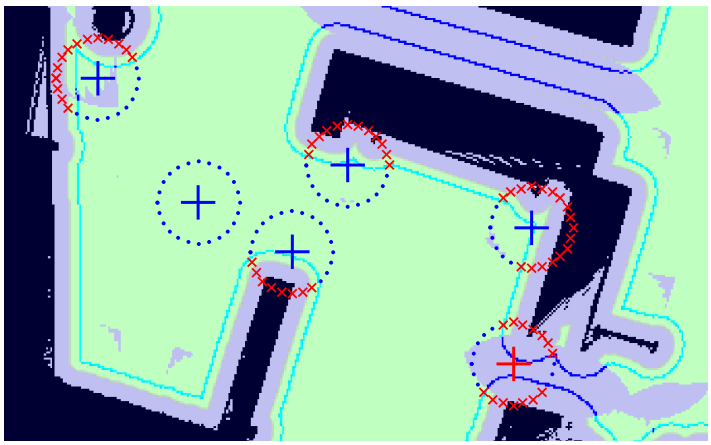


Figure 4.3: To search for valid push goals y, paths encircling potential goal positions (blue +) are searched for collisions (red $\times$/blue $\cdot$). Only the position in the door frame (bottom right) introduces a blockade, since it exhibits 2 occupied subpaths (red). All possible goal points according to the map $\mathcal{M}_g$ are shown by a green overlay. Figure adapted from [6].

4.3.1 Direct push

The direct push planner moves obstacles that block a desired path. Its goal is to determine an adequate and reachable goal position as well as a push trajectory. The global planner provides the obstacle to be pushed as well as the map and the desired path in the vicinity of the obstacle. The latter information is required to find a goal position that unblocks the desired path as well as an effective contact point. Task plans are defined by the parameters contact point x, pushing path $\mathbf{v}$ and push goal y, which are determined here based on the object's footprint in the map. One solution is selected and sent to the platform controller. If a haptic tag is acquired during manipulation, the process described in Sec. 4.1 is started.

Possible contact points between the robot and the obstacle must fulfill multiple conditions – solutions x *(a)* lie on a path encircling the object, similar to the path π discussed below, *(b)* are "free", i.e. $\mathcal{M}_d(x) > r_{rob}$, *(c)* can be reached by the robot, i.e. the path planner finds a path to x, and *(d)* allow for stable manipulation. The latter condition depends on the end-effector used. In the presented system, $\mathbf{v}$ should go through the connecting line between the object's center of mass and the floor, since the object would otherwise start to rotate. It is most stable if the normal of the object's footprint at the contact point is parallel to $\mathbf{v}$, thus the platform should approach the surface in a perpendicular fashion. The center of mass is assumed to be in the center of the footprint. If this assumption is wrong, the contact point can be corrected after the first exploration using H_D and H_R from the haptic tag.

The push goal point y must be chosen such that the new location is reachable, unblocks the blocked path and does not introduce any new blockades. The shortest path π on which an object can be encircled is defined by all points, which are r_{rob} from the closest point of

the object's footprint. If, for a potential object location y on the map, each point on π has $\mathcal{M}_d(\pi) > r_{rob}$, there is also no other obstacle that blocks the platform, and y is a valid goal point. However, there are more solutions, such as pushing the object close to a wall or another static obstacle, which is very feasible in practice. To find these solutions, π is split into subpath π_i at points $\xi \in \pi$ where $\mathcal{M}_d(\xi) = 0 \wedge \frac{\delta}{\delta\xi}\mathcal{M}_d(\xi) \neq 0$. In case y is close to only a single obstacle, there is exactly one subpath for which $\mathcal{M}_d(\pi_i) < r_{rob}$. If multiple of such subpaths exist, location y is close to two or more obstacles and would thus introduce a blockade of possible trajectories. A map of all possible goal points is generated for a local neighborhood within the manipulation radius r_{mani}:

$$\mathcal{M}_g(y) = \mathcal{M}_d(y) > r_{obj} \wedge |\{\pi_i | \mathcal{M}_d(\xi) < r_{rob} \; \forall \xi \in \pi_i\}| \leq 1 \tag{4.3}$$

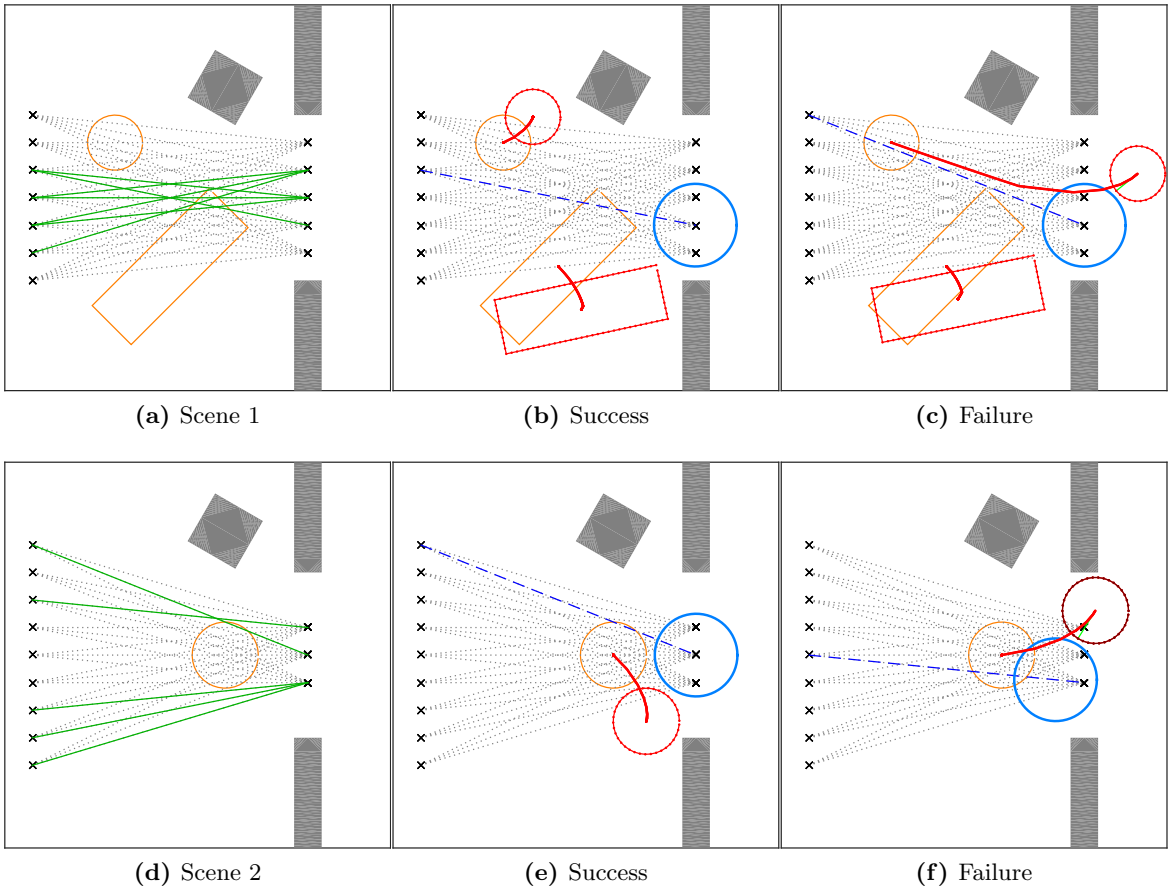
Fig. 4.3 shows this map and also exemplifies points on π for several goal points y located in free space, close to walls or corners and in a narrow doorway. Note the different detected subpath π_i for these goals. For simplicity of illustration, the footprint is approximated by the smallest enclosing circle, such that π is a circle as well. It is not required to test the costs of any points between π and y if $r_{obj} \geq r_{rob}$.

A set of path primitives – such as curves of different radii used in a local planner – is used for pushing path candidates. Their validity is verified by checking $\mathcal{M}_d$ at all points along a path, and for the remaining paths it is checked whether they establish connections between pairs of contact points and goal points. The resulting triples (contact point, path, goal point) are possible solutions for the manipulation problem. However, since the presented platform does not provide any lateral stability when pushing an object, only a straight pushing path ensures reliable manipulation. The straight line connects between contact point, center of mass and goal point. Such a feasible push plan is typically found if there is enough free space on opposite sides of the object.

4.3.2 Sideward push

While the direct push planner accurately controls the trajectory and goal of an obstacle, it is a time-consuming process. Apart from the slow speed during the push, the platform interrupts its navigation trajectory and needs to reorient. In some cases, a more efficient manipulation of obstacles may be performed, which is even integrable with the navigation trajectory: Instead of approaching the obstacle frontally, it is rather touched by the side of the (round) mobile platform, resulting in forward and sideward force components towards the movable obstacle. The latter component increases while the object slides along the curved side of the platform, resulting in a curved trajectory towards the left/right side of the platform.

The task plan is parametrized by a starting/contact point, an end point and a connecting trajectory, the motion of the platform. Contrary to the above planner, however, the relation between the platform and the object is not fixed and must be simulated before the task



(a) Scene 1 **(b)** Success **(c)** Failure

(d) Scene 2 **(e)** Success **(f)** Failure

Figure 4.4: Planning of the sideward push manipulation to unblock a path in two scenes *(a)*, *(d)* with static obstacles (gray), movable objects (initial pose in orange) and the mobile platform (blue). Unsuccessful candidate plans are depicted with a dotted line, successful ones with a green line. The remaining figures show simulation results for one robot trajectory highlighted in blue. Object trajectories and their final poses are shown in red. The plan in *(c)* fails since the round object is still in contact with the robot when it reaches its goal, and in *(f)*, the object collides with the gray wall.

Algorithm 2: Local planner for sideward push

Data: Platform trajectory; object o with centroid o_x^0, orientation o_α^0, contour o_j, mass o_m, moment of inertia o_i; environment e; contact range d_c
Result: Feasibility of plan (boolean); Object trajectory o_x^k, o_α^k

```
contact ← 0; k ← 1;
for x ∈ trajectory do
    (d, j) ← min_j(||o_j − x|| − r_rob);        // o_j: Closest contour point
    if d < d_c then
        contact ← 1;
        n ← normal(o_j);                        // Surface normal at contact point
        f ← −(1 − d/d_c) · n;                   // Pseudo-force perpendicular to surface (no friction)
        f ← scale(f);                           // Scale for adequate step-width
        o_x^k ← o_x^{k−1} + f o_m^{−1};
        ω ← (o_i × f) o_i^{−1};
        o_α^k ← o_α^{k−1} + ω_z;
        if collision(o, e) then
            return Failure: collision;
    else if contact then
        return Success;                         // Robot and object no longer in contact
    k++;
return Failure;                                 // Robot and object still in contact
```

is performed. For successful completion, the object must loose contact with the platform ("drop point") before the end point is reached and without hitting any other obstacle along the way. Feasible contact points are located at a distance r_{rob} away from the obstacle, within the region in front of the manipulation node (seen from the current position of the platform). Similarly, end points are located within the region(s) behind the manipulation node. The distance of those points to the manipulation node corresponds to the maximal acceptable sliding distance. Here, we use a value of 1 m, which corresponds to about $6r_{rob}$. Larger values are infeasible, since prediction of sliding (see below) becomes increasingly unreliable. Sets of starting and end points are generated by regular sampling of points on the map that fulfill the aforementioned conditions. Candidate plans are obtained by generating all possible tuples of starting points, end points and path primitives. Paths that reach inaccessible areas of the map are immediately rejected. Note that one end point is reached by different combinations of start points/path primitives. The associated trajectories exhibit only slight differences. Therefore, it is sufficient to use the straight line as the only path primitive. Additional primitives (curves) only need to be checked if all straight lines fail.

Each candidate plan is evaluated by simulation of the corresponding object trajectory. The simulation, see Algorithm 2 and Fig. 4.4, determines the object trajectory based on the given path of the platform, the shape of the object, as well as other obstacles like walls. The

algorithm iterates over the points of the robot trajectory and calculates a pseudo collision impulse on contact with objects. For details about modeling rigid body dynamics, see [100]. Here, a friction coefficient between the platform and the object of zero is assumed, and the impulse is perpendicular to the surface at the contact point. The simulation is quasi-static, i.e. objects are assumed to be static on contact. The collision impulse results in a translational and rotational pose update, which depends on mass and moment of inertia of the object. Dynamic effects are negligible for the considered speed range. The inertial moment is still required for separation of translation/rotation and is calculated assuming an equal mass distribution. By keeping the number of task candidates small (e.g. lower than 30), the complexity of the simulation is very low. It is sufficient to use a low sampling density for the contact/end points, since corrections in the low cm-range may still be performed during manipulation (see below). The quasi-static 2D motion simulation (line 9 ff.) is simple, and a sampling density of about 2 cm is sufficient. Furthermore, many plans are rejected quickly because of an inadequate contact point or an early collision with another obstacle.

Costs of feasible plans are determined by a constant setup time similar to the direct push planner and the duration of the navigation operation, considering a lower velocity limit than in free space. The cheapest feasible plan is selected and executed. The push trajectory remains near the ideal navigation path, and there is no additional pause after the manipulation ended. Therefore, the costs of a successful sideward push are considerably lower than for the direct push. During manipulation, object motion is compared to simulation by tracking the center of the contact region. Deviations may be caused by inaccurate modeling of friction or inertia and are compensated by slight corrections of the platform trajectory. (Typically, this is only required if the object is closer to the front of the platform than predicted, since an earlier drop point poses no problem.) Large deviations or unforeseen collisions with other objects result in a manipulation failure.

4.3.3 Transparent objects

Transparent objects in a room – such as windows, glass doors, tables or vases – result in inconsistent, missing or incorrect range values from a laser scanner. Similar to the Kinect (Sec. 5.1), there may be missing data ("shadows"), refraction effects, shine-through of the background and jumping range values[1] Since Visual SLAM methods rely on consistent measurements for successful integration, transparent objects may show up as distorted data or are eventually removed from the map. Even if they are successfully inserted into the map, new data may result in their deletion. Still, transparent objects may be detected by other means: The platform might randomly hit transparent objects and detect their shape and position using the tactile contact shape from the sensor described in Chapter 3. Also, a transparency detector as described in Sec. 5.2.2 could detect a transparent object using the Kinect or even

[1] Note that some high-end laser scanners detect multiple reflections, which may result from transparent objects. Multi-reflection is not considered here, due to the high costs of such scanners.

the laser scanner. Unlike data from the laser scanner, however, detection results come as irregular events.

If a transparency detection result is available, the transparent object is stored by adding a transparency node to the navigation graph. A safety region associated with this node is added to the region map $\mathcal{M}_R$, replacing existing regions at the same location. Unless the global planner specifically decides to do so, the safety region is not entered by the platform, avoiding collisions with the transparent object. Like that, there is no need to represent or integrate the object in the visually acquired map. Additionally, $\mathcal{M}_d$ is updated around the transparency node in order to avoid high platform velocities in its vicinity. Haptic exploration can be used to learn more about a visually detected transparent object. In case a haptic tag for the transparent object becomes available, the transparency node is treated as a manipulation node. If an assumed obstacle is not confirmed by haptic exploration, the transparency node is removed. In case the node was created based on a visual detector, a negative visual detection – e.g. a negative result form the transparency detector – also triggers removal.

4.3.4 Special operations

The navigation and manipulation graph offers great versatility by extending it with additional node types and local planners for specific tasks relevant during navigation of mobile platforms. For instance, door wings are common obstacles, which require specialized manipulation. The corresponding task is represented by an appropriate manipulation node. Detection of doors in the visual domain may be based on door frames and handles, see [83]. This approach also detects doors with a transparent wing, as long as there is an opaque frame. While pushing against a rotating door wing, in the haptic domain, a descriptive contact shape pattern is observed: The haptic tag shows a constant rotational component (H_R), a constant motion of the contact center as well as a planar shape primitive. This pattern may be used both for detection of a door wing and for state estimation during the manipulation task. Local planning for opening a door is fairly straight-forward with this state estimate: The door wing is pushed open far enough to allow the platform to fit through. If it does not move, it is blocked, and the manipulation task fails. Contrary to visual approaches, haptic state estimation also works on transparent doors.

Elevators, hallways in public buildings and production halls often exhibit automatic sliding, potentially transparent doors. In order to navigate through such doors, the state of their wings must be determined. Again, in case of transparent doors, this may require haptic exploration of the door wing. In case the door is closed, a trigger must be activated, such as pressing a button, moving in front of a motion detector or sending a wireless command to a building automation system. Obviously, some manipulation steps related to doors or elevators go beyond the capabilities of simple mobile platforms. A robotic arm is required in order to push down a door handle, press the button of an elevator or pull a door wing. If an appropriate arm is not available, the capabilities of a mobile platform is limited to pushing

Feature	PC tower	Stool	Cushion*	Paper bin	+ content	Table	Door*	Vase	+ water	Cleaner*
Force [N]	15.3	8.1	19.4	1.5	6.5	15.9	2.8	0.6	0.9	1.3
Dyn. Friction [N]	–	7.5	–	1.4	6.1	15.9	–	0.6	0.9	–
Deformability	0	0.08	0.2	0	0	0	–	0	0	–
Deceleration [s]	–	–	–	–	–	–	0.6	–	–	0.7
Drift	–	3e-2	–	3e-2	3e-2	3e-2	1e-1	3e-1	1e-1	–
Fixed/Fall	y/-	-/-	y/-	-/-	-/-	-/-	-/-	-/-	-/-	-/y
In Fig. 4.5a	(α)	(β)	(β)	(γ)	(γ)	(δ)	(ζ)	(ϵ)	(ϵ)	Fig. 3.6

* *Door:* Push against the open wing of a door, which continues its motion due to its inertia.
Cushion: Softer top of the (fixed) stool.

Table 4.1: Haptic tags of various objects obtained during the experiments

(a) Office scene **(b)** Foam-based sensor, object ϵ

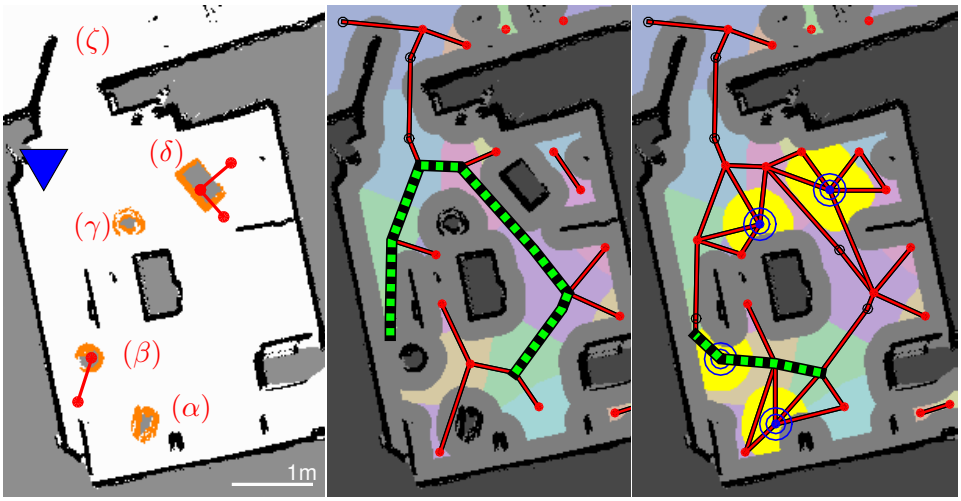
Figure 4.5: The proposed exploration system is tested in an office scene *(a)* with multiple movable objects (Greek letters). The foam-based sensor obtains haptic data even from transparent objects *(b)*. Figure adapted from [6].

door wings that are not snapped – a common scenario in public hallways. In other cases, the platform needs to wait for external help from a human or another robot. For instance, the platform could wait in front of an elevator until someone enters or exits.

4.4 Experiments

The proposed navigation and manipulation scheme is implemented on a robotic platform, integrating the proposed approaches, i.e. mapping, acquisition of haptic tags, generation of the joint navigation/manipulation graph and local task planning. At first, the platform is driven manually around the scene to have the Visual SLAM system acquire a map[2] as outlined

[2]Of course, the map can also be loaded from a previous exploration



(a) Scene map **(b)** Navigation planning **(c)** Joint planning

Figure 4.6: *(a)* Map of the office scene showing candidate objects for exploration (orange), feasible direct push plans (red) and the location of the photo in Fig. 4.5a (blue triangle). *(b)* Navigation graph built from the vision-based map. The shortest path between two nodes is shown in green/black and involves a long detour. *(c)* Manipulation nodes are added to the graph (blue) and allow for a much shorter and faster path. Figure adapted from [6].

in Sec. 2.3.6. For autonomous mapping, an automatic approach for exploration planning would be required, e.g. based on the detection of unexplored scene parts [32].

Next, the map is searched for explorable objects, i.e. objects that look like they could be movable by the platform. They are selected based on a simple scheme using the 2D map: First, connected regions corresponding to the footprints of objects are extracted. Since the visual map does not necessarily show the complete footprint, the convex hull is calculated. The main selection criteria is the size $s_{x,y}$ of the footprint: Large objects are usually heavy and cannot be pushed by small platforms, while small objects either fall when they are pushed, or belong to larger structures, such as tables. Only regions with $s_{x,y} \in [0.1\,\text{m}, 0.7\,\text{m}]$ are kept and explored by the platform one by one based on proximity and the possible cost benefit according to Eqn. (4.2). Fig. 4.6a shows the filtered regions in orange.

Haptic tags are obtained according to Sec. 4.1 for the selected objects, see Table 4.1. The following important object properties can be obtained from the tag: *Fixed* objects (PC) are too heavy to be moved by the platform; *falling* objects (cleaner) react in a sudden movement when pushed and can thus not be reliably manipulated; *deformable* materials retreat when pushed by the foam (here, the back of the cushion is fixed to apply a large force). The remaining objects are *movable*, but require significantly different efforts. Note how a larger weight of the same object (paper bin, vase) results in an increased friction force. The door exhibits a large drift, since it rotates around its hinge and continues its motion after the

platform stops. Finally, a large drift is observed during the first exploration of the vase: Its rectangular footprint was touched at a corner, such that the object rotated during the push. Haptic tags are associated to the map, such that the acquired haptic knowledge is stored and can be refined over time by additional explorations.

After haptic exploration, the visuo-haptic graph is generated according to Sec. 4.2.1. The navigation graph built directly from the map, see Fig. 4.6b, shows that there is no direct connection between nodes/regions in the lower left part of the room – due to a blockade by the stool (β). A path between nodes that are close-by in the map is thus quite long (depicted in green/black). Next, manipulation nodes are added to the graph (blue, Fig. 4.6c) for all the candidate objects, with node costs determined from the haptic tag. As the stool is detected to be movable during exploration, it is connected to other nodes with costs $t_{mani} = 7\,\mathrm{s}$. The push plan obtained for this object is depicted by a red line in Fig. 4.6a. Now, a much shorter and less costly path is found between the same two nodes (green/back in Fig. 4.6c). The lengths for the two paths are 6.0 m for pure (visual) navigation or 1.8 m for the combined navigation/manipulation plan. Navigation costs are determined based on a maximum platform speed of $v_{max} = 0.3\frac{m}{s}$. Manipulation is performed at a lower speed of about $v_{mani} = 0.05\frac{m}{s}$ and also requires constant costs of 4 s, which is typically needed to position the platform correctly.

In the top right of the room there are two detached nodes (Fig. 4.6b) because of a blockade by object (δ). One of the feasible push plans (red in Fig. 4.6a) can remove the obstacle and enable access to that part of the room. Note that the manipulation nodes for object (γ, α) exhibit a high connectivity to neighbors – yet, due to the manipulation costs, the associated edges would not be chosen by a path planner.

Visual processing for the proposed sensor runs at 30 Hz camera frame rate even on larger images (1600×896 pixels) using an Intel i7 platform. This is due to the fact that tracking relies on individual interesting points (such as the foam contour or the object region), instead of using the entire image. The high-level graph-based planning algorithms are implemented in MATLAB and require a few seconds for processing on larger maps as presented in Fig. 4.1b. Note, however, that there are no strict realtime requirements for high-level planning.

4.5 Summary

Indoor navigation of robots is typically based on occupancy grid maps, which represent space as “free” or “occupied” in 2D. Maps are generated using visual sensors, such as laser scanners. Graph-based representations can be derived from these maps and allow for path planning even in complex buildings. An abstract plan is extracted from the navigation graph, while platform commands are generated by a local planner that considers current sensor readings.

In this chapter, an extension of navigation graphs built from vision with haptic information about obstacles is proposed. Obstacles like glass doors that are invisible to a laser scanner as

well as movable obstacles that can be pushed away are integrated in the navigation graph as manipulation nodes. From the extended navigation/manipulation graph, a plan is generated, which integrates navigation and manipulation operations that are related to navigation. This representation avoids collisions with invisible objects, allows for shorter paths by moving an obstacle instead of taking a detour and enables access to areas that are otherwise blocked by a movable obstacle. Local planners are presented to control platform motions during manipulation operations.

5 Visuo-haptic geometry fusion

The acquisition or reconstruction of models from real objects is an important problem in robotics and other fields like augmented reality or gaming. Multi-view methods, such as KinectFusion or OctoMap, see Sec. 2.2.1, create a full geometric surface model of an object based on data acquired with a depth sensor from many views around the object. Accurate object models are important for robotic tasks like detection, manipulation and semantic planning. GraspIt [73], for instance, relies on accurate object models to find stable grasp configurations. In Chapter 6, object models are used to obtain local stiffness by simulation. For many methods, it is essential that the geometry is correct on the entire object – holes or large errors cannot be tolerated.

Multi-view reconstruction methods build high-quality 3D models by fusing measurements, thus reducing sensor noise and extending the observed space. One basic assumption in the fusion process is consistency of the observed 3D structure. This means that objects in the scene must not move and exhibit diffuse (Lambertian) reflection for all viewpoints. Only small deviations, e.g. a small component of specular reflectance, can be accepted. Transparent objects, however, cannot be reconstructed with such methods, since their image formation model is more complex than for diffuse reflection. Regardless of whether an active visual 3D sensing technique or a passive stereo camera is used – the appearance of transparent objects depends on multiple factors such as the background behind the object, viewing angle, object geometry and local reflectance.

Transparent regions cause errors or gaps (holes) in the reconstruction process, which must be detected in order to obtain complete and correct object models. To this end, an algorithm for transparency detection is proposed, which searches depth maps for geometric inconsistency effects caused by transparency. Consistent scene parts are filtered out. The detection result represents a rough indicator for transparent regions – it can at best reconstruct the convex hull of such regions. Similar to the above reconstruction methods, the scene must be acquired from multiple views by moving the camera along a trajectory. Detected inconsistencies are accumulated in a 3D volume. The proposed method detects objects with a smooth, curved surface exhibiting dominant refractive effects and, with limitations, surfaces with specular reflection. The following discussion focuses on transparency effects – with according arguments for reflective objects.

In order to obtain details about the geometric structure within transparent regions, i.e. to "fill holes" of geometric models, a haptic exploration scheme is proposed. Points from the object surface are acquired by touch with a single point contact probe mounted on a robot

arm. It is assumed that at least a partial visual object model is available, i.e. the object has some non-transparent areas. The result of transparency detection serves as an initialization, providing a region in which to conduct haptic exploration. Haptic exploration steps consist of moving the probe along a line, until contact with the object is detected. These steps are planned such that they provide most information gain about the object surface, according to an estimator. A surface is reconstructed using radial basis functions (RBF), a method for fitting and extrapolation of surfaces. Fitting is based on a set of so-called support points. Initially, this set consists only of valid points from the available visual model, which are located at the edges of the hole. There are no data points within the region of the hole, such that initial extrapolation results are inaccurate. Subsequently, additional "haptic" (i.e. haptically explored) points within this region are added to the set of support points by each exploration step. Extrapolation is refined by new points in an iterative fashion. The certainty of the reconstructed surface is estimated with a distance measure, based on the proximity to support points. Haptic exploration is then performed within the most uncertain region, i.e. along a line through the surface point with the largest distance to any support point. Finally, a complete object model is obtained from the visual model together with mesh extrapolated from haptic points.

This chapter is party taken from [4].

5.1 Effects of transparency

Image formation for transparent objects is based on a multitude of effects, as discussed in [47]. In this work we mainly exploit the "lens effect" caused by refraction of light passing through a curved transparent object. This is the dominant effect for clear, smooth materials hit by light rays with a non-acute incident angle. Looking at a transparent object, an image of the background is observed, which is distorted by refraction of the light ray as it passes through the air-surface boundaries of the transparent object. In this distorted image, depth estimation methods see a "virtual object" whose depth is shifted from the real depth of the background, see Fig. 5.1. The depth distortion depends on a multitude of variables, such as the real depth of the background, angle and position of the refracting surfaces (i.e. pose and geometry of the transparent object), camera position, refraction index and thickness of the material. As the main purpose of our approach is to find unknown transparent objects, only very weak assumptions about their geometry can be made. Therefore, it is impossible to predict or model the expected background distortion. For instance, even a simple bottle refracts a light ray four times on its inner and outer surfaces. Due to these considerations, we do not attempt to model the expected distortion, but rather focus on detecting *any* background distortion. This approach is also more robust when incorrect or no depth data are measured, and it can deal with additional depth distortion effects like specular reflection.

For active sensors, such as the Kinect, refraction on the object does not only occur for

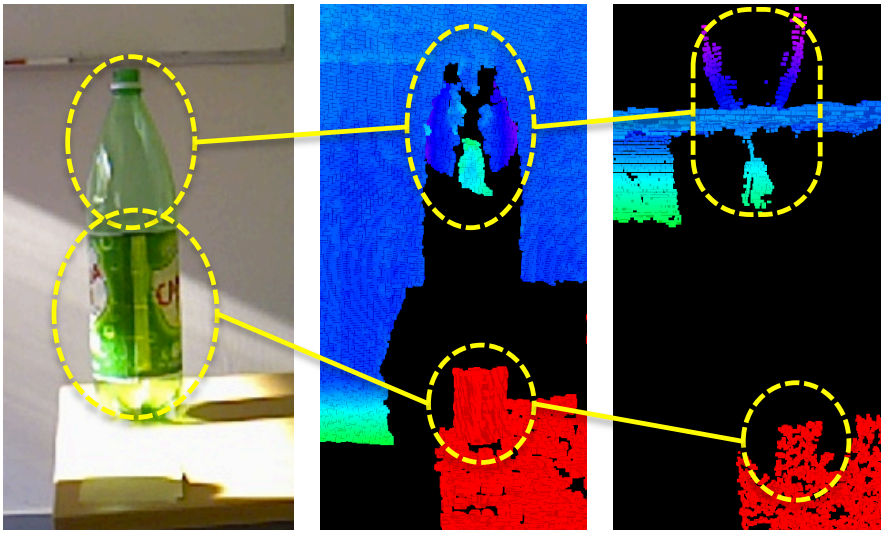


Figure 5.1: A partly transparent bottle in front of a wall is observed by a Kinect/Xtion sensor *(left)*. The corresponding point cloud (*center,* rendered above the viewpoint of the real camera) is correct around the label, whereas the transparent bottleneck is missing. Instead, this part of the object distorts the background by refraction, clearly visible in the top view *(right)*. Figure adapted from [4].

incoming light rays (traveling towards the camera), but also for outgoing light rays (from the projector). Like that, a partly distorted pattern is projected onto the background. Depending on camera-projector baseline and scene geometry, refraction at the transparent object may affect the measurement in three ways: Distortion of incoming rays, of outgoing rays or of rays in both directions. In addition, due to the strength of the projected rays, a significant portion is reflected on the transparent surface, leading to specularities in the IR image.

The Kinect sensor requires the projected pattern to be intact in a certain local neighborhood for successful matching. This is the case if refraction along a smooth surface is the dominant effect. However, if the pattern is distorted by too many effects – such as diffuse reflectance, attenuation or sub-surface scattering – pattern matching will fail, yielding an invalid depth (or hole in the depth map). Objects exhibit different dominant effects in different regions of the surface and depending on the camera pose. In a single view, the depth image of the transparent object is often "sparse", i.e. (distorted) depth data are only available in some areas. Thus, measurements from multiple viewpoints must be combined.

5.2 Transparency detector

The detection method is based on a search for geometry distortions caused by transparent objects. These distortions are geometrically inconsistent from different viewpoints, as they depend on multiple variables discussed above. In contrast to [65], which also detects

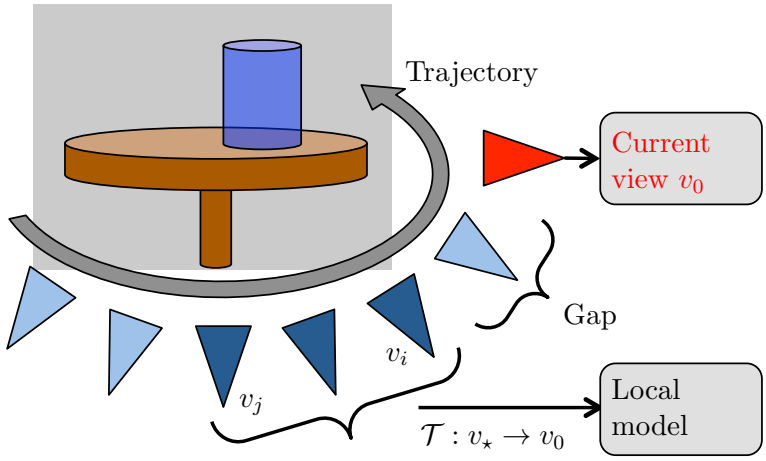


Figure 5.2: Proposed setup: While the sensor is moving around a transparent object (blue) in the scene, a background model is generated and compared to the current observation. Figure adapted from [4].

transparency with a Kinect, see Sec. 2.2.2, the presented method processes seemingly "valid" measurements from the sensor. It is therefore not required to provide geometric models of the transparent objects. The sensor is moved along a trajectory, which should ensure a broad coverage of viewpoints onto the relevant scene parts (space of interest, Fig. 5.2). On a robot, this step could be performed during navigation or while positioning the arm. Camera poses are determined by a visual tracker, see Sec. 2.3.2. During the movement, the expected, stable scene geometry is predicted from past measurements (background model) and compared to the current observation.

5.2.1 Local background model

A local scene model, yielding a geometry estimate along with a reliability measure, is built based on depth maps taken from multiple viewpoints. These viewpoints are located on the sensor trajectory, close to the current observation both in time and space. We work with eight views roughly 0.02 m apart along a distance of 0.2 m. Compared to a global model, this approach offers several advantages: The model is generated online, there is no need for a separate, time-consuming exploration step, as changes in the scene cause only local distortions and there is no need for global geometric consistency.

The model must deal with a number of error sources. First of all, the Kinect sensor exhibits a certain noise level over time and space. Under ideal conditions (e.g. a bright matt planar surface), as shown in [53], the noise can be modeled by a Gaussian with standard deviation σ_k dependant on depth z (in meters). Furthermore, there are small tracking and calibration

errors, which are modeled by a Gaussian with σ_t. Together, these components determine the standard deviation of the expected minimum sensor noise σ_s, with σ_s from [53] and σ_t measured from experiments:

$$\sigma_s(z) = \sqrt{\sigma_k^2 + \sigma_t^2} = \sqrt{\left(\frac{1}{2} 2.9 \cdot 10^{-3} z^2\right)^2 + 0.01^2} \tag{5.1}$$

On the other hand, scene parts such as edges, fine structures or certain material types exhibit a much higher noise level. For instance, depth edges are depicted frazzled and flicker between background and foreground over time due to the matching neighborhood used by the sensor. In other cases, the sensor yields mostly invalid data. This is the case for surfaces hit by the projected rays with a very acute angle, some active light sources, surfaces with a very low albedo, scene parts which are beyond the measurement range as well as for transparent materials. Even though the (few) valid observations of these scene regions might be consistent, their reliability is low. Finally, errors may be caused by jumps in the estimated pose or losses-of-track, which temporarily invalidate all measurements.

A model for each current viewpoint $c = v_0$ is built from depth maps taken at nearby poses $v_i, \dots v_j$ with $j > i, i > 0$. Depth maps are projected into the view c, see Fig. 5.2, using poses obtained by the tracker [10]. Hence, $(j - i + 1)$ measurements are available for each depth pixel whereof some provide invalid data, leaving a lower number n of valid measurements. Assuming a Gaussian distribution, a mean depth $\overline{D^{(c)}}$ and a standard deviation $\hat{\sigma}^{(c)}$ can be estimated with a maximum likelihood estimator. The estimation bias of $\hat{\sigma}^{(c)}$ is corrected by $\sigma^{(c)} = \sqrt{\frac{n}{n-1}}\hat{\sigma}^{(c)}$. As n is quite low, it is infeasible to estimate more complex models – such as a Gaussian mixture model – even though they might be more realistic.

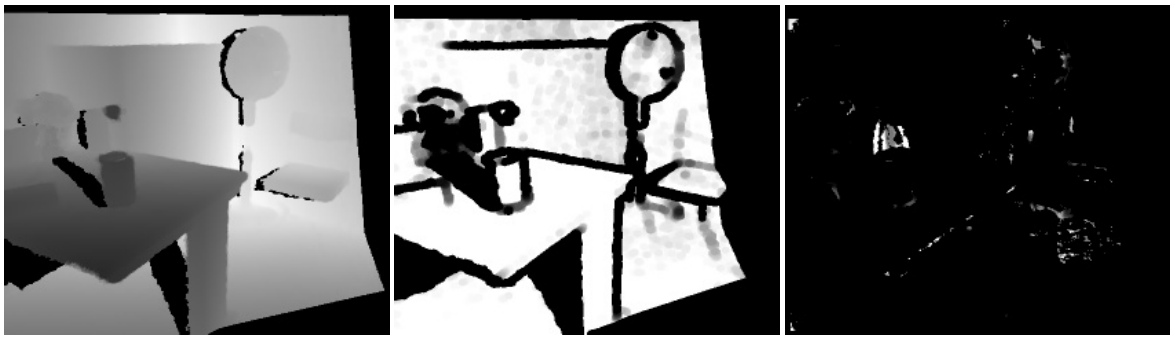
The observed Gaussian distribution $\mathcal{N}_{\sigma^{(c)}}$ is compared to the expected error model $\mathcal{N}_{\sigma_s}$ given by Eqn. (5.1). Reliable regions in the scene exhibit a standard deviation $\sigma^{(c)}$, which is lower or equal to the expected error. The reliability is quantified with a score determined by comparison of the two distributions using the squared Hellinger distance. This distance is commonly used to quantify the similarity between two probability distributions, see [80]. For two Gaussians with equal mean, it is calculated according to:

$$H^2 = 1 - \sqrt{\frac{2\sigma_1\sigma_2}{\sigma_1^2 + \sigma_2^2}} \tag{5.2}$$

The reliability score is evaluated per pixel as follows:

$$\mathcal{S}_r^{(c)} = \ell\left(1 - wH^2\right) = \ell\left[1 - w\left(1 - \sqrt{\frac{2\sigma_s\sigma^{(c)\prime}}{\sigma_s^2 + \left(\sigma^{(c)\prime}\right)^2}}\right)\right] \text{ if } n \geq 3 \tag{5.3}$$

$$\text{with } \sigma^{(c)\prime} = \max\left(\sigma^{(c)}, \sigma_s\right)$$



(a) Model depth $\overline{D^{(c)}}$ **(b)** Reliability $\mathcal{S}_r^{(c)}$ **(c)** Error $\mathcal{S}_E^{(c)}$

Figure 5.3: Local background model and error score/image for a view $c = v_0$ of the scene with object "Bottle-G", see also Fig. 5.6a. The model is generated from past views taken right of v_0, leaving a region of low reliability that appears as a shadow left of the object. Figure adapted from [4].

The value of the score is 1 for reliable regions where $\sigma^{(c)} \leq \sigma_s$ and 0 for $n < 3$ or very unreliable regions. The weight w is chosen such that $\mathcal{S}_r^{(c)}$ drops to 0 for $\sigma^{(c)} \approx 4\sigma_s$. Function ℓ clamps the lower bound of the score value to 0 in case of large errors. An erosion operation is applied to the reliability image $\mathcal{S}_r^{(c)}$ in order to add a margin to the unstable regions.

The views $v_i, \dots v_j$ are taken in a dense fashion from the recent sensor trajectory. A gap of about 16 cm is maintained between the closest view v_i and the current view v_0 in order to avoid model distortions in the same image area as in the current observation, caused by the transparent object itself. For instance, assume the camera turns around the space of interest, as depicted in Fig. 5.2. The most recent views are distorted by the transparent object in almost the same image area as where the object is currently seen. Older views, on the other hand, provide a valid estimate of the current background, as the distortion from a transparent object is projected to other image regions, see the "shadow" in Fig. 5.3b.

The model for the current view, see Fig. 5.3, consists of $\left(\overline{D^{(c)}}, \sigma^{(c)\prime}, \mathcal{S}_r^{(c)}\right)$, which is a straightforward and sufficient representation of the expected scene geometry at the current viewpoint, merging information from past views $v_i, \dots v_j$. It can be directly compared to the depth map at $c = v_0$. Processing of depth maps is fast, especially compared to operations on volumetric or point-based data. While many scene reconstruction methods try to find the best guess in noisy data, the proposed method suppresses any unstable regions, in order to allow for a reliable rating of the current observation.

5.2.2 Detection of inconsistent geometry

Transparency is detected by comparing the current depth observation z to the scene model while moving the sensor around the space of interest. The comparison is performed at regular intervals (such as 2 cm) and should be carried out over a preferably large range of different

viewpoints. Given the model $\left(\overline{D^{(c)}}, \sigma^{(c)\prime}, \mathcal{S}_r^{(c)}\right)$ and the observation $z' = \left|z - \overline{D^{(c)}}\right|$, an error score is derived from the probability $p\left(|Z \sim \mathcal{N}_{\sigma^{(c)\prime}}| > z'\right) = 1 - \text{erf}\left(\frac{z'}{\sqrt{2}\sigma^{(c)\prime}}\right)$, which identifies observations that contradict with the model. Noise is suppressed and stability is increased by only considering high probabilities, using a mapping of $p \in [\theta, 1.0] \rightarrow [0, 1]$ and $p < \theta \rightarrow 0$ with $\theta = 0.9$. Together with the model reliability $\mathcal{S}_r^{(c)}$ this yields the error score/image:

$$\mathcal{S}_E^{(c)} = \mathcal{O} \cdot \mathcal{S}_r^{(c)} \cdot \ell \left\{ \frac{1}{1-\theta} \left[1 - \text{erf}\left(\frac{z'}{\sqrt{2}\sigma^{(c)\prime}} \right) - \theta \right] \right\} \tag{5.4}$$

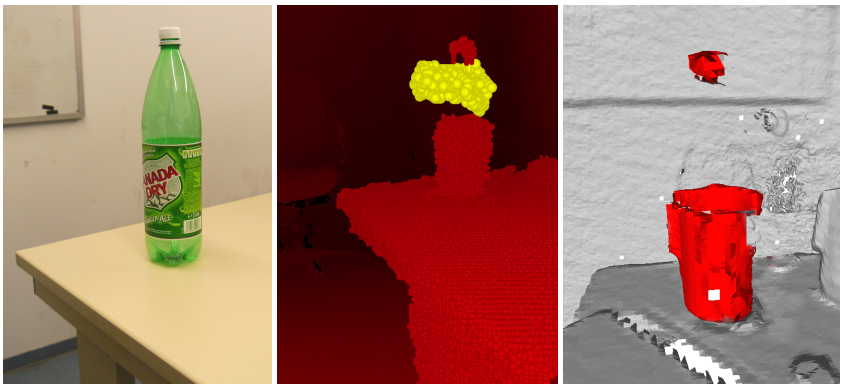
The Boolean term $\mathcal{O}$ determines whether the observed 3D point was already in the field of view during model generation. Like that, an object that suddenly comes into view of the camera does not trigger an error. A large error signal identifies geometric inconsistency caused by transparent objects, see Fig. 5.3c: The model predicts the depth $\overline{D^{(c)}}$ of the bottleneck with a high certainty $\mathcal{S}_r^{(c)}$ at the background, which is *not* confirmed by the current observation. Thus, this region triggers a large error signal $\mathcal{S}_E^{(c)}$. The error signal does not trigger on inconsistent regions caused by static effect other than transparency or specular reflection, as those regions are assigned a low reliability value $\mathcal{S}_r^{(c)}$ during model generation. Moving objects, however, would result in a large value of $\mathcal{S}_E^{(c)}$. Yet, there is some tolerance to this case, as the error will not concentrate in a certain region of space. In general, however, static scenes are assumed.

If a large error signal is found, no assumptions can be made about the real depth at the affected pixel. Each pixel corresponds to a ray in space, and all we know is that there is something unreliable along that ray. Therefore, using the known intrinsic and extrinsic camera parameters, the 2D error signal is reverse-projected into a 3D "transparency volume," which covers the space of interest. While the camera moves along its trajectory, error rays with $\mathcal{S}_E^{(c)} > 0$ are cast into this volume, accumulating at voxels in or near the unreliable space, i.e. near the transparent object. If enough viewpoints are integrated, ideally, the convex hull of the object is regenerated within the transparency volume. For a good location and rough shape estimate of the object, the viewing angles of the space of interest should cover 90° or more. Measurements with large erroneous regions are ignored, as they are typically caused by scene movements or temporary tracking errors.

Clusters of high values in the transparency volume correspond to a single object or a connected transparent part within that object. The outer voxels of a cluster form a point cloud that shows the rough shape of the object.

5.3 Geometry estimation in transparent regions

As discussed, transparency poses a major problem to visual geometry reconstruction methods, such as KinectFusion or OctoMap [44, 76]. Fig. 5.4 shows reconstruction results of a partly transparent bottle, subsequently referred to as "visual geometry": Transparent regions mostly



(a) Object "Bottle-G" **(b)** OctoMap **(c)** KinectFusion

Figure 5.4: Geometry reconstruction of a partly transparent bottle *(a)* with OctoMap *(b)* or KinectFusion *(c)* results in an incomplete model (red), with holes in transparent regions. Additional objects were placed on the table for KinectFusion to ensure correct tracking. Note the errors from refraction effects behind the object in *(c)*. In *(b)*, the result of transparency detection is shown in yellow. *(b)* Figure adapted from [4].

do not show up in the reconstructed geometry, since they do not represent consistent geometry. In some cases, there may be erroneous reconstructions, caused by effects described in Sec. 5.1. Multi-view reconstruction and transparency detection are run in parallel in order to obtain registered results. This is possible since both approaches rely on the same input data, i.e. depth data acquired along a free camera trajectory.

Results from geometry and transparency reconstruction can be combined at two levels: First, processing can be performed on individual views, effectively filtering out invalid depth values from each depth map before casting them into the reconstruction volume. Detected transparent pixels in the error score, see Fig. 5.3c, would just be removed from the depth maps. However, it must be ensured that (almost) all invalid depth values are removed reliably. Otherwise, reconstruction quality might even deteriorate, since individual incorrect values are no longer contradicted by other measurements and thus become part of the reconstruction. This requirement is not fulfilled by the presented transparency detector, since individual error images are often sparse and only show parts of the transparency region. It would be necessary to grow the regions in the error images, for instance by filling surrounding holes and by combining results from multiple views.

Instead, it is favored to combine the final reconstruction results in 3D. Like this, by integration of many different views, geometry reconstruction removes already many inconsistent regions, and the transparency detection process creates a dense volumetric representation. Fig. 5.4b shows how the two detection results complement each other. The transparency region may be considered a failure indicator of geometry reconstruction as well as a rough estimator of the object geometry with in this failure region. Yet, there is a gap between the

two regions, which must be considered in further processing. Additionally, a small misalignment is observed due to shadowing effects of the projector-camera system. This bias vanishes if views from all around the object are considered.

5.3.1 Surface extrapolation

Extrapolation is a straight-forward way to fill holes in surface models based on nearby or sparse samples of the surface. The underlying assumption is that the surface extends smoothly beyond the known parts, and higher-order parameters – such as curvature – are preserved. Yet, it is obviously impossible to predict features within the region of the hole, which do not extend beyond it. Here, radial basis functions (RBF) are used for surface extrapolation, as proposed in [16]. This method enables surface smoothing and extrapolation, even over large gaps. Also, it exhibits robustness to noise. RBFs would allow representing the entire object in a single model, including both visually and haptically obtained parts. Yet, this would go along with high computational costs, so instead, RBF is just used to fill regions of holes.

Extrapolation based on RBF is a three-stage process:

1. Definition of a set of so-called "support points" with normals $\mathbf{X}_s$, which lie on the searched surface. RBF does not impose constraints on the sampling of these points. For smoothing applications, point sets are very dense, while extrapolation usually goes along with sparse point sets.
2. Extension of the support set by off-surface points $\mathbf{X}_{s'}$, which are located along the normals at a given distance d from the original support points.
3. Fitting of the RBF r to the extended support set. RBFs represent an implicit surface by a signed distance function, i.e. $r(\mathbf{X}_s) \stackrel{!}{=} 0$ for the support points, and $r(\mathbf{X}_{s'}) \stackrel{!}{=} d$ for the off-surface points.
4. Extraction of the isosurface (zero-crossing) of the RBF. The surface is generated with Marching Cubes [62], which creates a mesh for implicit functions in an efficient way. Based on the function values sampled at the corners of a cube, a triangular mesh element is created using interpolation and a lookup table. This process is repeated for all cubes in a regular grid, within a region obtained by growing the detected transparency region.

The RBF functions have the general form

$$r(\mathbf{X}) = p(\mathbf{X}) + \sum_{i=1}^{N} \lambda_i \phi \left(|\mathbf{X} - \mathbf{X}_i| \right), \tag{5.5}$$

where p – low-degree polynomial, $\mathbf{X}_i$ – support points and ϕ – basic function. Several choices exist for $\phi(x)$, such as the thin-plate spline $x^2 \log(x)$, the Gaussian $\exp(-cx^2)$, the biharmonic spline $\phi(x) = x$ and the triharmonic spline x^3, which is used here. Fitting RBFs to a set of

support points (step 3) may be considered a training process. The coefficients in Eqn. (5.5) are obtained by solving a linear system, which respects certain side conditions, see [16].

Support points $\mathbf{X}_s$ are taken from the known part of the visual model, i.e. from non-transparent areas. It is sufficient to select only those parts of the geometry, which are in the vicinity of the hole – far-away structures are typically not relevant for extrapolation. Also, the size of the support set should not exceed 1000 points, in order to allow reasonable training times of a few seconds. To this end, the set of support points is chosen as a "band" on the known geometry around the hole. The geometry of this band is extended into the hole, so to speak, but it cannot be expected to reconstruct geometric details that do not show up clearly outside the hole. Note that the transparency region is typically a bit smaller than the visual object model. In order to find the support points near the hole, the transparency region must be enlarged until it touches the visual model. This can be achieved by locally growing the transparency region until it overlaps with known points.

5.4 Haptic exploration of geometry

Missing regions (holes) of geometry models obtained by visual measurements are complemented by points that are acquired haptically, i.e. by touch. In most cases, touch allows the acquisition of data in regions where vision failed. Haptic exploration is the process of planning where to touch the object. Here, single point contacts are considered, i.e. a robot arm with a sensor tip is moved until contact with the object is detected. In order to obtain a surface, points from haptic exploration $\mathbf{X}_H$ are used as support points for surface extrapolation according to Sec. 5.3.1.

Extrapolation is incapable to predict geometric details without any additional data. Even though the predicted surface may be correct in simple cases, in general, the certainty of the extrapolated surface decreases with increasing distance to support points. We seek to obtain an estimator of uncertainty for the extrapolated surface, assuming that the surface is locally smooth, and certainty decreases with increasing distance from support points. Also, it is assumed that "simple" surface areas (such as planes) have greater range of certainty than complex (e.g. highly curved) areas. If these assumptions are invalid for a given object, the exploration is not performed in an optimal way. It might require a few more iterations, but coverage of the entire surface of the object is still ensured.

The estimator is based on a distance measure, which models the proximity of a surface point $\mathbf{X}$ to any support point $\mathbf{X}_s$. Here, a measure is required, which is fast to compute for many point pairs and considers the global shape as well as the connectivity of the mesh. Several distance measures for meshes have been proposed, see e.g. [60]. The geodesic distance measure is based on the shortest path between two points on the surface. Despite its simplicity, this approach has some drawbacks: Distances are not smooth and they are sensitive to small changes, since only a single path is considered. Moreover, computation for many point pairs

is complex. Other popular measures consider the entire connectivity between any two points, not just a single path: The diffusion distance is based on a heat diffusion process, and the commute-time distance is based on the round-trip-time of a random walker. Diffusion processes have a time parameter, which allows emphasising either local or global properties. Finally, the biharmonic distance [60] is related to the latter two measures and represents both local and global features in a balanced way. It is used here due to its beneficial properties and its ability to provide distances for all point pairs in an efficient way. In the continuous case, the biharmonic distance is expressed as follows:

$$d_b(\mathbf{X}_1, \mathbf{X}_2) = \sum_{k=1}^{\infty} \frac{\phi_k(\mathbf{X}_1) - \phi_k(\mathbf{X}_2)}{\lambda_k} \tag{5.6}$$

Where ϕ, λ are the eigenvectors and eigenvalues of the Lapace-Beltrami operator. We calculate the uncertainty score from the biharmonic distance as follows:

$$\mathcal{S}_U(\mathbf{X}) = \min \{ d_b(\mathbf{X}, \mathbf{X}_s) \; \forall \mathbf{X}_s \in \text{ support points} \} \tag{5.7}$$

The score is evaluated on the entire extrapolated mesh. Since the biharmonic distance is evaluated for all point pairs of the mesh, there is hardly any performance degradation for larger support sets.

Haptic exploration is performed by moving a single point probe along a straight line with a robot arm until an obstacle is detected. The exploration process is parametrized by a starting point, a line of exploration and a constant motion velocity. These parameters are determined by the maximum position of the uncertainty score Eqn. (5.7). The maximum corresponds to the "least supported" surface point, which is expected to provide most information about the real object surface. The line of exploration is chosen such that it goes through this maximum $\text{argmax}_X \, \mathcal{S}_U(\mathbf{X})$ and intersects the estimated central height axis of the object perpendicularly. This way, exploration is limited to a plane parallel to the floor. The starting point is chosen at a "safe" distance away from the extrapolated mesh, e.g. a distance equal to the estimated radius. Contact with the object is determined by a conventional force sensor – or by the beam-based visuo-haptic sensor presented in Chapter 3. The position of the sensor tip at the contact event corresponds to a haptically acquired surface point $\mathbf{X}_H^{(h)}$. After transformation to the visual coordinate frame, $\mathbf{X}_H^{(v)}$ is added to the set of support points. The transformation $\mathcal{T}_{(h)}^{(v)}$ between the haptic and visual coordinate frames must be known with great accuracy, i.e. the two frames must be registered. A tracker such as Sec. 2.3.4 may provide $\mathcal{T}_{(h)}^{(v)}$ by simultaneously tracking templates on the object and the manipulator. It also compensates for motions of the object caused by haptic interaction.

Exploration planning for multi-point contacts, such as humanoid hands, requires more sophisticated approaches, due to the complexity of grasp patterns. Yet, the proposed scheme can be extended to multi-point contacts in an efficient way using a greedy search: Initially,

Figure 5.5: Office scene used for the evaluation of transparency detection. Transparent objects are placed on the table, and the tripod with an Xtion depth sensor (right) is then moved around the table.

the starting point for the first finger is planned as discussed. This imposes constraints on the configuration of other fingers, i.e. the reach of each other finger is limited. An updated score $\mathcal{S}'_U$ is calculated, which considers the influence of the first contact point. It is not required to recalculate the score – instead, the point is just added to the support set. Next, the starting point for the neighboring finger is determined by finding the maximum value of the updated score within the reach of this finger. The process is repeated until starting points are found for all fingers.

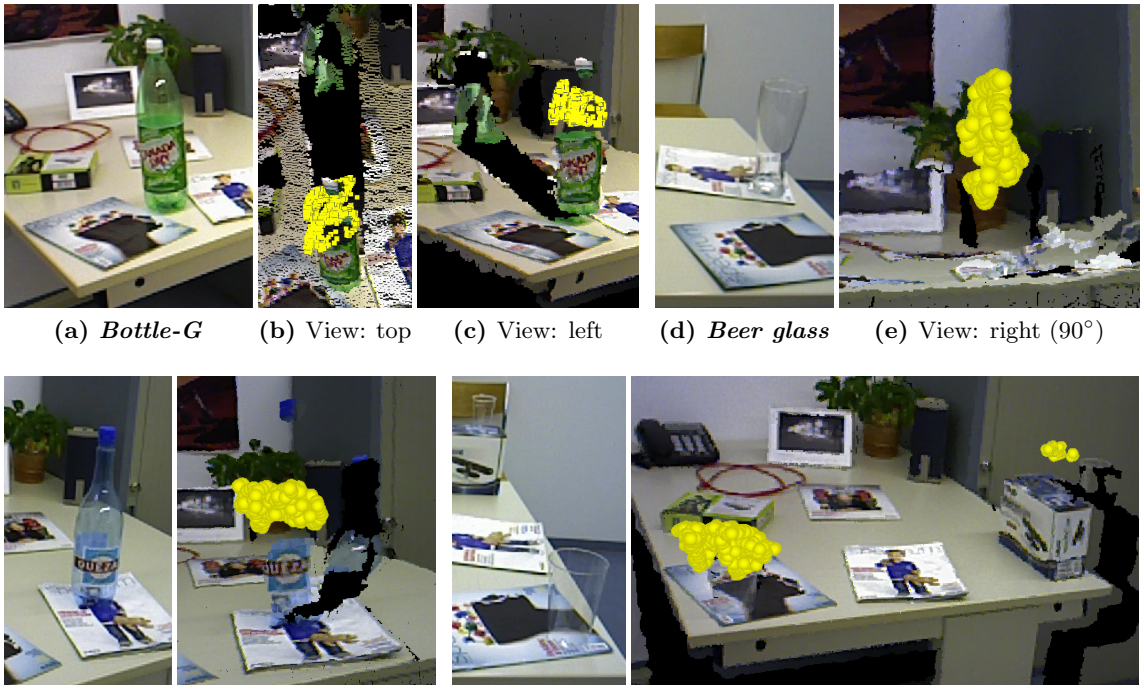
5.5 Experiments

This section presents results of transparency detection for several objects in a real-world scene. Haptic exploration and surface extrapolation are analysed in a simulation environment, using a large database of object models.

5.5.1 Transparency reconstruction

The method for transparency detection is evaluated in a cluttered office scene, see Fig. 5.5. Different transparent objects made of glass or plastic are placed onto the table, and depth maps are acquired with an Xtion/Kinect camera along a circular trajectory around the scene. The space of interest covers the space on and above the table and is set to $1\,\mathrm{m}^3$. The camera pose is estimated continuously using the tracker from [10]. As the tracker offers an online learning mode, it is not required to build any model of the scene beforehand.

Reconstruction results are shown in Fig. 5.6. The detected transparent regions are depicted as yellow “transparency” point clouds, together with a registered colored point cloud obtained directly from the Kinect. The bottles “Bottle-G” and ”Bottle-B” exhibit non-transparent surface areas with Lambertian reflection at the printed label, which are measured correctly by



(a) ***Bottle-G*** **(b)** View: top **(c)** View: left **(d)** ***Beer glass*** **(e)** View: right (90°)

(f) ***Bottle-B*** **(g)** View: slight right **(h)** ***Cups*** **(i)** View: right

(j) ***Wine glass*** **(k)** View: right **(l)** ***Mirror*** **(m)** View: top left

Figure 5.6: The proposed transparency detector is tested in a cluttered scene with six different objects, see camera images in *(a, d, f, h, j, l)*. Remaining pictures visualize the detected transparent parts as yellow point clouds, together with the colored raw point clouds from the sensor. The rendered viewpoints are shifted as indicated compared to the physical camera position, resulting in unknown areas that appear in black. Figure adapted from [4].

the Kinect, see Fig. 5.6c and 5.6g. The upper transparent parts, however, are measured incorrectly and distort the background (see area near the plant in Fig. 5.6c). The proposed method accurately detects these regions as transparent parts. The objects "Bottle-B" in Fig. 5.6f and "Wine glass" in Fig. 5.6j exhibit relatively fine transparent structures at the bottle neck and the stem, respectively. These are too small for detection – only the larger transparent parts are reconstructed. Furthermore, an example of specular reflection ("Mirror") is shown in Fig. 5.6m. Here, the sensor measures the depth of the scene shown in the mirror, i.e. the wall behind the camera, shown in bright blue. This is a geometric inconsistency as well, appearing as a moving hole in the wall. As expected, this reflection is detected as well.

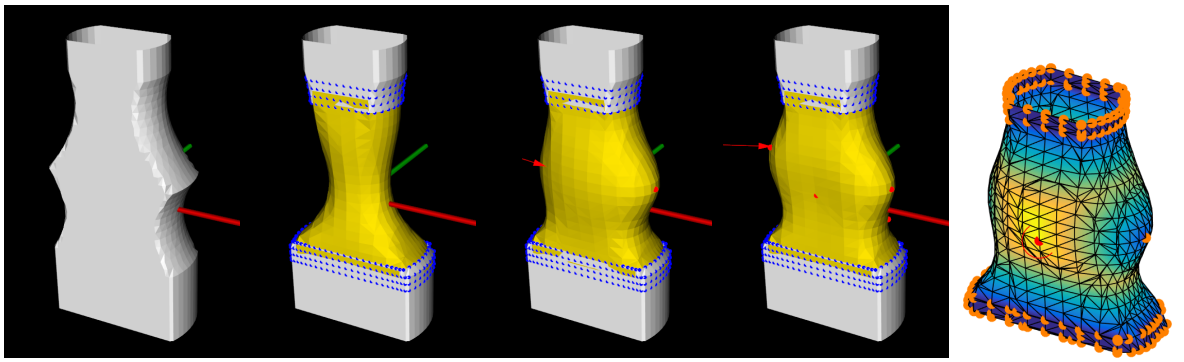
Finally, Fig. 5.4b shows the reconstruction result of object "Bottle-G" with OctoMap [44] as a red point cloud. This approach combines multiple views to reconstruct Lambertian scenes in a compact volumetric representation. Here, a voxel resolution of 5 mm is used, and the camera pose is taken from the above-mentioned tracker. The scene is accurately reconstructed, including the opaque label of the bottle. However, the transparent part of the bottle is missing in the reconstruction. Transparency detected by the proposed method is shown as a yellow point cloud and complements the 3D model of the scene.

Our approach is implemented in C++ using the ROS framework [82] and tested on a current Intel i7 machine with 4 physical cores. Eqn. (5.3) and (5.4) are approximated by a quadratic polynomial for faster processing. The measured runtime for model generation and detection is 180 ms for 8 model views and 250 ms on average for processing of the transparency volume running in a parallel thread. Using the suggested frame distance of 2 cm, realtime processing is possible for a motion speed of approximately $8\frac{\text{cm}}{\text{s}}$. The tracker [10] runs in parallel at camera frame rate, partly using the GPU.

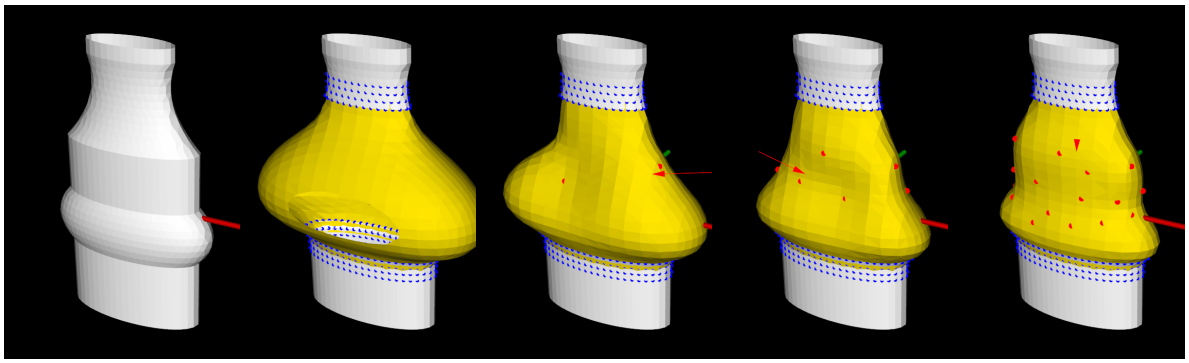
5.5.2 Simulation of haptic exploration

The proposed scheme for geometry estimation by haptic exploration is evaluated in a simulation environment. This experiment was partly conducted together with [31]. A dataset of 700 artificial object models is generated using the same approach as described in Sec. 6.1.2. These models comprise variants of bottles with round, elliptic and partially planar shapes. They are assembled from several "rings" with varying heights and curvatures. Fig. 5.7 shows some of them in the first column. These complete models serve as the "reference geometry". Next, a large part in the center of each model is cut out to simulate a transparent region (hole). The remaining geometry (upper and lower parts) represents the "visual model" and is depicted in gray in Fig. 5.7.

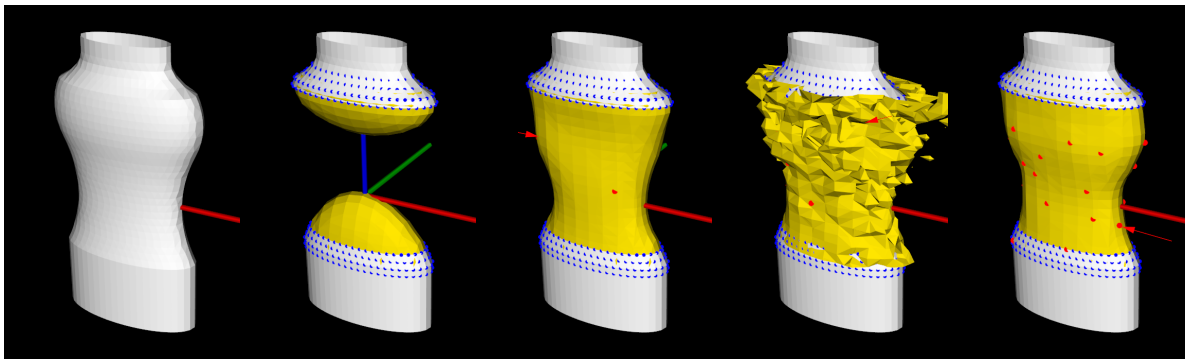
Extrapolation according to Sec. 5.3.1 is applied to fill the hole. The support points are sampled from regions in the "visual model" that are close to the hole. Results are depicted in the second column of Fig. 5.7, with support points shown in blue. The extrapolated mesh (yellow) complements the visual model and looks "smooth" – yet, significant errors are observed: The geometry is too small for object A, and too large for object B. For object C,



Object A Visual extrapolation Iteration 2 Iteration 6 Score $\mathcal{S}_U$ (it. 2)



Object B Visual extrapolation Iteration 5 Iteration 10 Iteration 30



Object C Visual extrapolation Iteration 3 Iteration 19 Iteration 30

Figure 5.7: Simulation results of haptic exploration for three artificial objects. The complete geometry is shown on the left, whereof the central part is subsequently removed and estimated by extrapolation. The remaining pictures show intermediate extrapolation results (yellow – extrapolated surface, blue – visual support points, red – haptic support points). The uncertainty score is shown on the top right (orange – support points, red – detected maximum).

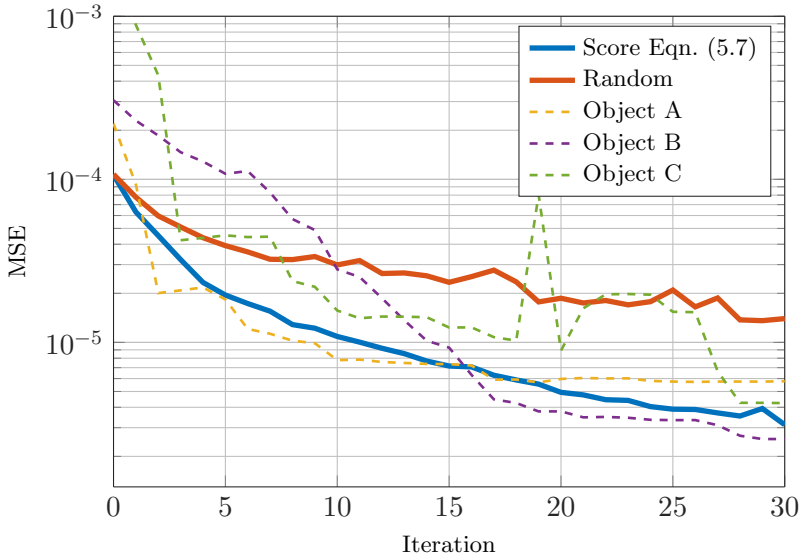


Figure 5.8: Error of the extrapolated geometry obtained by simulated haptic exploration. The blue curve shows the average for the entire dataset.

the extrapolated geometry is entirely wrong. These problems may be attributed to wrongful hints in the support region, i.e. the geometry does not continue as in the support region.

Next, haptic exploration is performed as discussed in Sec. 5.4. The uncertainty score $\mathcal{S}_U$ is evaluated for the extrapolated mesh as presented in Eqn. (5.7). A haptic exploration step is planned based on the location of the maximum uncertainty score. Exploration is performed along a line and yields one point on the object surface from the reference geometry. This point is added to the set of support points. Surface extrapolation is repeated with the updated support set. "Haptic" support points are shown in red in Fig. 5.7. The process is iterated, each time improving accuracy of the extrapolated mesh by one additional support point. Fig. 5.7 (top right) shows the uncertainty score obtained after two exploration steps. The current location of the maximum (red point) determines the third exploration step.

Considerable improvements of the extrapolated mesh are obtained already by the first few exploration steps. Geometric details of the three objects become visible after 6/10/3 exploration steps, respectively. An extrapolation error is observed in iteration 19 for object C, probably due to a numerical instability. This problem disappears in the next iteration.

Finally, the quality of the extrapolated mesh is evaluated. To this end, the distances between the reference geometry and the extrapolated surface are calculated at regularly sampled locations. Distances are squared and averaged within the extrapolation domain to obtain the mean squared error (MSE). The MSE is plotted in Fig. 5.8 over the number of iterations. Curves are shown for the three objects of Fig. 5.7 as well as for the average of the entire dataset of 700 objects. On average, an improvement of one order of magnitude is observed after 10 single point exploration steps. For the three exemplary objects, the MSE drops

rapidly for A, C, and for B, there is a continuous improvement. Also, note the peak for object C at iteration 19. These observations are consistent with the images in Fig. 5.7. For comparison, the average MSE is also given for random exploration, i.e. exploration steps are performed along random directions. The exploration planner performs considerably better, rapidly improving the geometry in the beginning and further refining it in higher iterations.

5.6 Summary

Existing methods for geometry reconstruction, such as [44, 76], cannot cope with transparency. A detector for transparency is presented, which detects the rough position and shape of fully or partially transparent objects. It is based on a search for geometric inconsistencies between the current observation and a local background model. These inconsistencies stem from refraction effects, which are observed on any curved transparent surface. Data are acquired with the same setup as for the above reconstruction methods, by moving a depth sensor along a trajectory around the scene. Experiments are presented, showing reconstruction results for various bottles and glasses made of different materials like plastics or glass. Existing methods for transparency detection are either based on pre-learned models, specularity features, structured scenes and lighting, or they rely on specialized sensors, which for instance scan the object at different wavelengths [47].

In order to obtain a complete geometry model of an object, a haptic exploration scheme is presented. The model uses visual geometry where available, and fills holes by surface extrapolation with radial basis functions (RBF). Haptic exploration is performed with a robot arm, which moves a single point probe along a line until a touch event is detected by a force sensor. Exploration is planned in such a way that the respective "most uncertain" point of the surface is touched. Surface extrapolation uses the explored points as support points, together with visual support points at the edges of the hole. Like that, arbitrary surfaces can be estimated. Certainty is expressed based on the so-called biharmonic distance between extrapolated points and existing support points.

The proposed scheme is verified in an extensive simulation-based experiment. It works with dense and sparse point set and is thus used for both visually and haptically acquired points. The idea of fitting surfaces to points acquired haptically is a common principle, used e.g. in [70]. Yet, most existing methods do not work with generic surface models, but rather fit simple surface primitives such as planes, curves or spheres.

6 Deformation models for thin-walled objects

Deformable objects change their shape when a force is applied to them during a manipulation process. This behavior must be considered for planning of manipulation tasks, such as grasping. In this chapter, an object model for thin-walled objects is proposed, which integrates deformation and stiffness characteristics with surface-based object representations as used for visual reconstruction and object classification/recognition. These "haptic" models, see Fig. 6.1, are developed for and demonstrated in two applications: Sec. 6.2 presents an approach to obtain stable grasp configurations, and Sec. 6.3 introduces a method for object classification based on haptic exploration, both applied to plastic bottles and similar objects.

Haptic classification is considered as a subsequent process following vision-based recognition and typically discriminates between a low number of model candidates. A robot explores haptic features such as the local stiffness on a real object and compares them to simulated data. The presented models are especially suited for objects with varying local stiffness along their surface. This applies to thin-walled objects, such as plastic bottles, cups or other containers, which exhibit deformation characteristics that are strongly correlated with their surface geometry. Objects made of soft homogeneous materials, on the other hand, exhibit a more uniform surface stiffness that is determined by the inner structure.

Deformation characteristics of an object are obtained by elasticity simulation based on the Finite Elements Method (FEM). The simulation is performed offline for a large object database, or during a training phase for individual objects. A set of representative grasp patterns is applied to the surface of each object. Simulation provides *deformation shapes*, i.e. the changed geometry resulting from a given force or grasp pattern. So-called *local stiffness maps* as well as stiffness features are derived from a set of deformation shapes.

Elasticity is a volumetric property, necessitating the generation of volumetric object models. It is not possible to directly acquire complete volumetric models from an object in an automated fashion with sensors that could be mounted on a (humanoid) autonomous robot. Instead, volumetric models are created from artificial surface models, using simplified assumptions about the volumetric structure, such as homogeneous contents or predefined volumetric structures like thin walls. The training of a classifier for haptic object recognition requires multiple variants of such models in order to avoid overfitting. Such variants are created by parametric object models.

Parts of this chapter have been published in [8, 9].

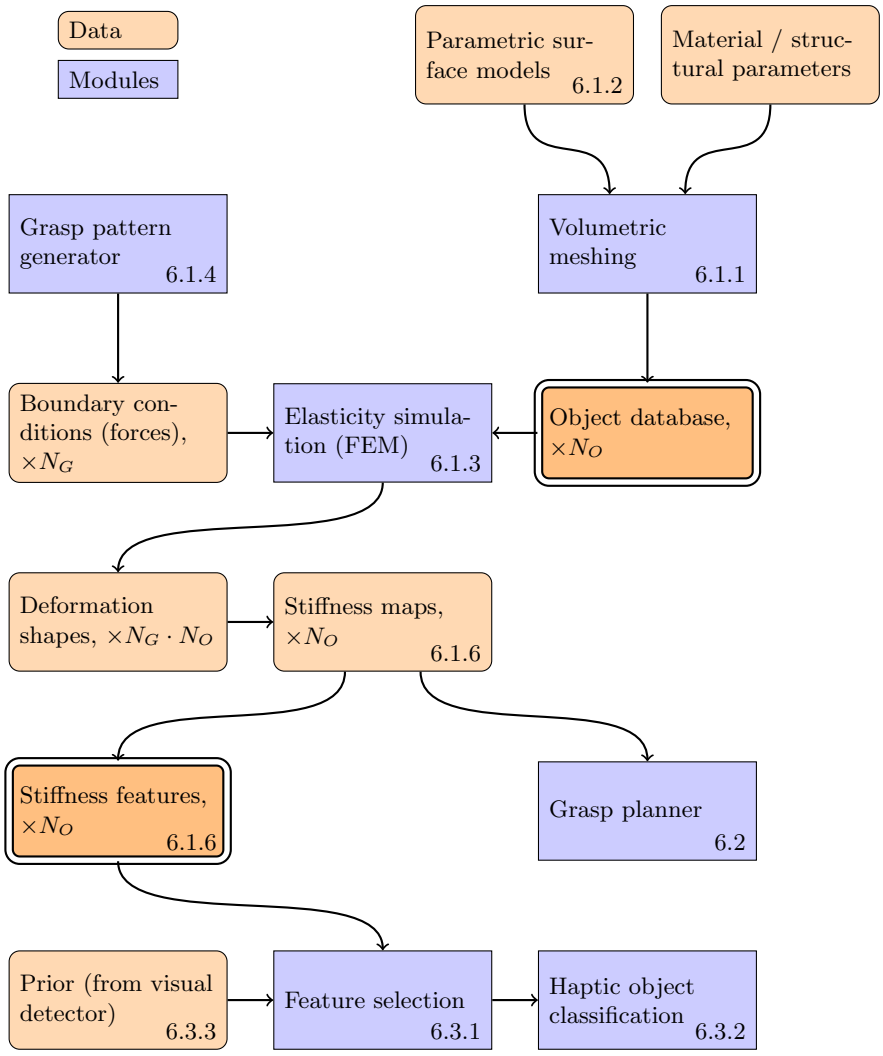


Figure 6.1: Overview of object classification and grasp planning with deformable objects models. During training, N_O object models are generated, and their deformation is analyzed at N_G grasp points. References to the respective sections are given.

6.1 Simulation of object deformation

This section discusses the generation of artificial object models, their use in an FEM-based simulation, generic grasp patterns as well as the modeling of liquids in containers. Furthermore, the proposed stiffness features and stiffness maps are presented.

6.1.1 Volumetric object models

Simulation of deformation is performed with the Finite Element method (FEM) and requires volumetric object models. Such models represent surface geometry and inner structure as well as elasticity parameters of the material, see also Sec. 2.2.4. In computer vision (as well as in computer graphics), object models usually represent only the surface geometry using triangular meshes, which are optionally textured. Surface models are used in numerous manual and automatic modeling processes as well as for visual recognition and classification. Automatic reconstruction systems such as [76] build accurate surface models even from a handheld moving depth camera, see also Sec. 2.2.1. In order to benefit from this exiting infrastructure and from existing object databases, volumetric models are built here based on surface models. This process relies on additional information or assumptions about objects, which may be available from priors, scene context, object databases or, to some extent, from measurements. These assumptions are ambiguous, such that several volumetric models may be derived from a single surface model.

In very simple cases, a volumetric representation may be obtained by filling the inside of a surface model homogeneously. Yet, most artificial objects have an inner structure that is more complicated. In this work, we consider deformable containers or packages – such as plastic containers, bottles, bowls or cups for food, drinks, cleaning agents and other liquids. These objects are highly relevant for autonomous robots that perform manipulation operations in household and office scenes. They exhibit thin outer walls (or "shells") and are often produced by injection molding. This technique leads to relatively homogeneous materials within the walls and allows for a great variety of shapes – which results in a large variety of deformation behavior. Additionally, the overall stiffness varies within a large range, based on material type and wall structure.

The material within the walls (the "contents") is typically homogeneous and consists of air or liquids. It must be distinguished between a gas or liquid that is enclosed inside the container ("closed") and an "open" container, from which gas can escape. These different states have a considerable influence on deformation behavior and can be modeled in a simple way: The content exerts a pressure to the inner surface of the shell caused by a volume reduction of the gas, see Sec. 6.1.5. This inner pressure can be fed into the FEM simulation as a boundary condition. Thus, as long as dynamic behavior is not of interest, additional mesh elements are not required to model the contents. Solid structures inside of containers, on the other hand, cannot be modeled in such a simple way.

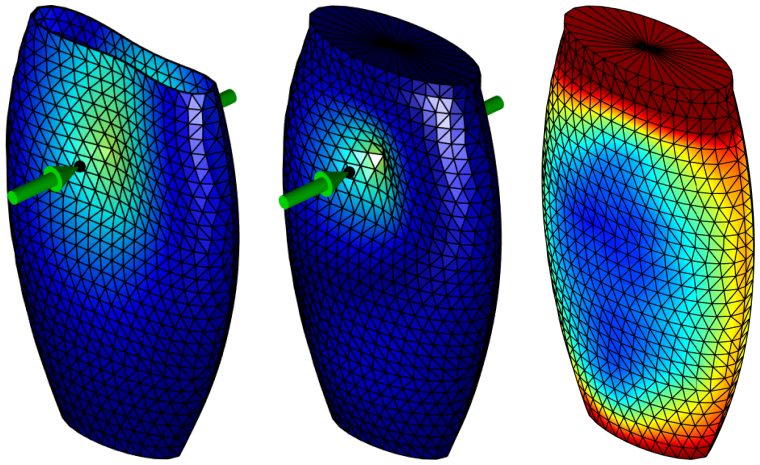
Object models may be generated artificially, i.e. with a construction program, or measured from a real object. The typical FEM workflow relies on the former type with accurate (noise-free) geometry as well as complete knowledge of the inner structure and material parameters. For objects in household and office environments such models are usually not available. Instead, however, surface models are often available, for instance from online databases, see e.g. [56]. Volumetric models are therefore built here from surface models, extended by a description of the wall structure, material properties, etc. These parameters are specified manually during the training process, obtained from priors (e.g. semantic information), assumptions or from haptic exploration of real objects. Generally, there are multiple options for structural or material parameters, such that multiple volumetric models are built from one surface model. The container models are built from parametric models, see Sec. 6.1.2, which allow for the generation of numerous object variations. In order to ensure accurate results from FEM simulations, volumetric meshes must obey certain quality criteria. There are established tools that refine meshes accordingly – such as [91] for tetrahedral meshes.

Alternatively, volumetric models could be generated from a visually acquired surface mesh. This approach, however, has a number of drawbacks. Multi-view methods for model generation apply smoothing in order to create "nice-looking" surfaces, which results in the removal of sharp structures or edges. This can be tolerated for many applications in computer graphics, as inaccuracies may be compensated by texture information. Yet, deformation behavior changes considerably if such sharp structures are removed. On the other hand, if no smoothing is applied, surfaces become rough or noisy, which also distorts deformation characteristics. Furthermore, multi-view modeling approaches are time-consuming, making it unfeasible to create large object databases. Multiple variants of each object would be required and be scanned separately.

6.1.2 Parametric model generation

Parametric models generate volumetric object models based on geometric parameters, structural parameters and material parameters. The geometric parameters control sections or control points of a parametrized object model. By sampling of this parameter space, many variants of an object can be created. These variants are needed for several reasons: Deformation behavior shows a great variability depending on the extent of certain structures and on basic object parameters. Even for visually similar objects, geometric details or small structures may have a large impact on deformation characteristics. Furthermore, training processes used in machine learning approaches require a large number of variations to obtain realistic representations of a certain object class and to avoid overfitting.

In this work, a dataset of artificial parametric surface models for typical containers and bottles is created manually. Objects are built on a base area, which is an ellipse, a "flattened" ellipse or a circle (resulting in a solid of revolution). The side walls consist of multiple sections of splines, which are either straight, bent to the inside or bent to the outside. The



(a) No rim **(b)** Top rim **(c)** Stiffness map

Figure 6.2: Simulated deformation shapes of two models obtained from a force stimulus by the indicated grasp pattern. There is a reinforcement structure (cap/rim) on the top of object *(b)*, which is missing in *(a)*. In *(c)*, the local stiffness map of the object in *(b)* is shown (blue – soft, red – hard). *(c)* Figure adapted from [9].

parametric model generator used here is illustrated in Fig. 6.3a. Generating geometries from such simple primitives makes it easy to create a large number of object variations by changing the relations of the fundamental dimensions, the positions of salient points or turning points, local curvature as well as the extent of convexities or concavities.

Despite their simplicity, the generated models show great variations in deformation behavior. Accurate and regularly sampled surface geometries can be generated from parametric models – a clear advantage compared to models obtained from visual observation. The thin-walled structure is built with a second surface (the inner surface of the shell), which is created by pushing all vertices to the inside by the amount of the wall thickness along the surface normal. Additionally, to ensure realistic representations, typical structures of containers are modeled – such as bottle caps, dents or reinforcement rings (rim). The latter structure is an essential feature for grasp planning often found at the top end of bottlenecks or plastic cups. Additional object instances are generated, exhibiting variations of these structures.

6.1.3 FEM-based elasticity simulation

An FEM model is built from the volumetric object models as discussed in Sec. 2.2.5. First, a stiffness matrix is built for each tetrahedral element based on an elasticity model that relates force and deformation (displacement). The system matrix assembled from element matrices relates forces and displacements for all nodes of the object. A solver determines the deformation shape for a given configuration of boundary conditions. Boundary conditions are specified on surface points of the volumetric mesh. Displacements or deformations δ

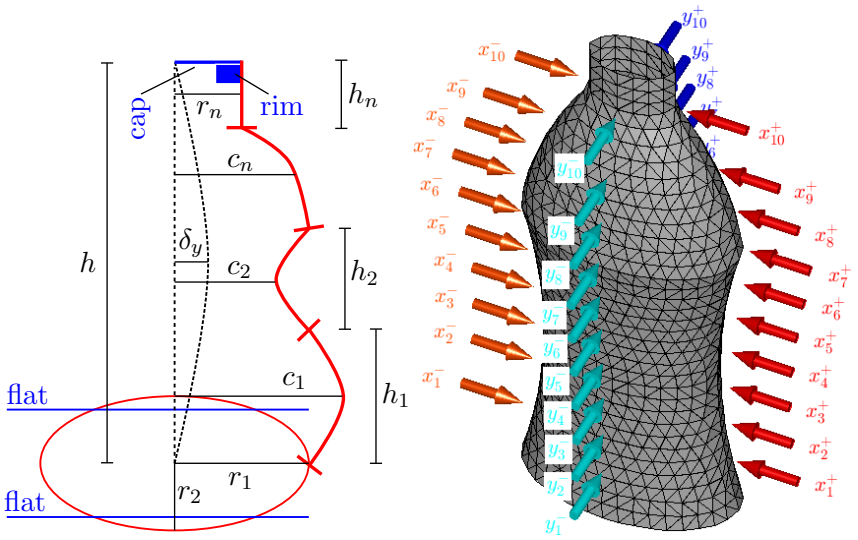
represent Dirichlet boundary conditions, forces F represent Neumann (second type) boundary conditions. The object must be fixed at some points, so for all vertices of the base plate, $\delta_{x,y,z} = 0$. (Without this fixture, a physics simulator would be needed to calculate object motion.) For points on the outside surface of the wall, $F_{x,y,z} = 0$, except on points where a grasp is applied to. The time scale for time-dependent deformations is small compared to the duration of the grasping process. Therefore a steady-state simulation is performed, yielding the new rest configuration of the object when a constant force is applied at certain points.

The goal of deformation simulation is to obtain a versatile representation of object deformations for all feasible grasp configurations. Therefore, in an offline process, a deformation database is built by applying an exhaustive number of grasp configurations to the outer object surface, see Sec. 6.1.4. The object's reaction, i.e. the displacement of all nodes, or deformation shape, is obtained for each grasp configuration. The surface near the grasp points moves towards the inside, whereby the extent of the affected area depends on support structures and object geometry, see Fig. 6.2. Spatially distant areas typically do not deform much, or they are even slightly displaced to the outside. Even though linear elasticity models are used, deformation behavior of the object may be non-linear. For instance, a large change in geometry may lead to the weakening of certain support structures. To account for this effect, each grasp pattern is simulated with multiple force values. Results from all deformation shapes are subsequently integrated in a stiffness map, see Sec. 6.1.6.

6.1.4 Grasp patterns

Generic grasp patterns are used during simulation to generate boundary conditions for the FEM. These patterns are applied to the surface either in a dense fashion to obtain a complete stiffness map, or only to the feature locations (see below) to allow for faster simulation during online processing. The simplest realistic grasp configuration that provides a stable grasp is represented by two single points of contact on opposing sides of the object, such as shown by the green arrows in Fig. 6.2. The line connecting these two points goes through the central height axis of the object. The forces at the two contact points are directed in opposing directions and thus result in a zero global force. Fig. 6.3b visualizes a feasible set of such grasp patterns, see below. These patterns can easily be applied to real objects by using a parallel two-finger gripper with short rods or "finger tips".

Other than a simpler single point contact, this dual-tip configuration represents realistic grasps: Due to the counterforce applied on the opposing point, the object does not globally deform "away from" the contact point. Like that, a characteristic local deformation and stiffness is observed. Grasp patterns with more complex grippers or hands require more sophisticated models and are not evaluated in this work. Still, the dual-tip configuration is representative for more complex grasps: The counterforce could be applied by any number of fingers or even a laminar contact on the back side, without significantly changing the local stiffness at the frontal contact. Any grasp requires a balance of forces, which is already a



(a) Model generator **(b)** Feature locations

Figure 6.3: *(a)* Parametric model generator: The red curve is rotated along the elliptic/flattened (*flat*) base to create the object surface. It consists of sections with height/curvature h_i, c_i and base/neck radii $r_{1,2,n}$. Special structures, such as *cap*, *rim* or a non-symmetric shift δ_y may be added to the geometry. *(b)* Locations of stiffness features on the object, with a sampling pattern of 4×10. Opposing arrows (e.g. x_{10}^+ and x_{10}^-) correspond to one dual-tip grasp pattern. *(a)* Figure adapted from [9].

fundamental property of the dual-tip configuration. Local stiffness, on the other hand, would change considerably if an additional finger is pushed onto the surface within a certain neighborhood around the first finger. The extent and shape of this neighborhood depends on the object geometry and could be represented by a coupling matrix. There is no coupling between two opposing contact points. Also, to represent more complex grasps with multiple contact points, the deformation shapes from multiple dual-tip configuration can be superimposed.

In simulation, one grasp pattern corresponds to two point stimuli applied to two opposing nodes on the outer surface of the object. The Neumann boundary conditions for these points become nonzero. Forces are applied to the selected point $\mathbf{X}$ on the outer surface and its opposing point $\mathbf{X}'$, see Fig. 6.2a. The line $\mathbf{XX}'$ intersects the central height axis of the object in a perpendicular fashion. Forces are applied along $\mathbf{XX}'$, according to the dual-tip grasp pattern. The simulation is repeated for all points on the outer surface, yielding one deformation shape for each point $\mathbf{X}$ and each grasping force value.

6.1.5 Modeling the contents of containers

The contents within containers or bottles is not considered by the volumetric modeling introduced above. Instead of explicit modeling, the properties of the contents my be represented

by additional boundary conditions. Changes of the inner volume caused by deformation result in a pressure, if the contents is enclosed and consists of gases or liquids. Pressure is isotropic and results in a force on each node of the inside wall. Note that in fluid mechanics, FEM is frequently used to simulate flow and pressure, yet this requires different types of finite elements. The following models are used to model contents of containers:

Air, non-enclosed If there is an opening in the container, the air inside escapes when the object is compressed. This is the case for empty bottles if the closure or cap is open. In steady state, pressure on the inside wall equals the pressure on the outside wall, thus $F_{x,y,z} = 0$ on the inside wall. The ambient air pressure is not of interest, since it is exerted to all surfaces equally.

Air, enclosed In this case, pressure obeys the Boyle-Mariotte law $P \cdot V = const$ – i.e. it increases with decreasing volume. In rest configuration, the volume enclosed by the inner wall is V_0, and the inside pressure is assumed to be equal to the ambient pressure of $P_0 = 10^5$ Pa. The inner pressure $P_I = \frac{V_0}{V_1} P_0$ is obtained for the deformed object with inner volume V_1. Due to isotropy, this pressure acts everywhere on the inside surface. The force applied onto a triangle of the inside wall with area A_{tri} and normal $\mathbf{n}_{tri}$ is

$$\mathbf{F_{tri}} = (P_I - P_0) A_{tri} \mathbf{n}_{tri}. \tag{6.1}$$

Since boundary conditions must be applied to vertices, the force is split equally to the vertices of the triangle. Contributions from different triangles are summed up.

Deformation shape and inside pressure are mutually dependent. Thus, Eqn. (6.1) could be integrated into the FEM system matrix Eqn. (2.7). Yet, this would break its sparsity, and also FEM solvers are usually integrated as "black box" systems. Instead, Eqn. (6.1) is integrated as boundary conditions that are updated in an iterative fashion. In order to avoid oscillations, a damping factor c is introduced:

$$P_I[t] = c \cdot P(V_1) + (1 - c) \cdot P_I[t - 1]. \tag{6.2}$$

Iteration stops when the amount of volume change drops below a predefined threshold.

Liquids Water and similar liquids are almost incompressible, i.e. they preserve their volume, but not their shape. Typically, there is always some amount of air inside the container in addition to the liquid. The pressure within the volume of the enclosed gas behaves as discussed above, with updated volume terms:

$$P_G = \frac{V_0 - V_L}{V_1 - V_L} P_0, \tag{6.3}$$

with the constant volume of the liquid V_L. For the non-enclosed case, $P_G = 0$.

Within the liquid, the pressure increases slightly with depth d due to gravity, yielding $P(d) = P_G + 1000\frac{kg}{m^3} \cdot 9.81\frac{m}{s^2} \cdot d$. Thus, the forces applied to the inner vertices (see above) depend on the object height. Yet, the gravitational effect is low for small objects, such that it is neglected here. One exception to this are containers with thin, unstable walls, such as plastic bags.

6.1.6 Stiffness maps and features

The deformation behavior of a single point on the surface, without considering any deformation beyond that point, is referred to as *local stiffness*. This value is an important property for grasp planning and haptic classification of deformable objects. For a given point $\mathbf{X}_A$, local stiffness is obtained from the deformation of this point when a force is applied to the same point – i.e. only a single displacement is extracted from the deformation shape that corresponds to the grasp at $\mathbf{X}_A$ (and its opposing point). The grasp pattern also determines the direction of the force, which is roughly perpendicular to the surface. Deformations along other directions can be neglected for the objects considered here. For linear elasticity models, the local stiffness has the unit $^{\mathrm{N}}/_{\mathrm{m}}$ and is used as in Hooke's law. A dense representation of the local stiffness is created for all points on the object surface. This so-called (local) *stiffness map*, see Fig. 6.2c, allows predicting the deformation of each surface point, given the applied force (or vice versa).

Higher-order elasticity models are feasible to represent objects in a more realistic manner. Multiple deformation-force pairs are obtained by running the simulation for each grasp pattern with multiple grasping force values. This results in a deformation-force curve for each surface point, which does not necessarily follow a linear elasticity model. Higher-order models are required especially in case of large deformations or for objects with a counter-pressure from the inside, see Sec. 6.1.5. A linear and a quadratic model are fitted to the observed curve. Fitting is performed both for the full range and for a reduced range of deformation of up to 3 mm. Stiffness may vary greatly between these ranges, which are adapted to different application scenarios. Three coefficients are obtained from the fitting process for each point of the stiffness map:

1. Linear elasticity coefficient, low deformation (i)
2. Linear elasticity coefficient, full deformation (I)
3. Quadratic elasticity coefficient (normalized), full deformation (II)

The quadratic coefficient is normalized by a typical nominal grasping force and deformation of (50 N, 1 cm) for numerical reasons.

The stiffness map is a detailed haptic representation of an object, yet its calculation is time-consuming, and a direct acquisition from the real object is infeasible. Therefore, a lower-dimensional (sparse) representation of stiffness is required, the *stiffness features* $\mathcal{F}$.

They are determined by sampling of the stiffness map at descriptive locations, capturing extremal points of the stiffness map. Sampling points must be aligned with the peaks/valleys of the stiffness map. A fixed pattern of feature locations is required, which fulfills these criteria and is adequate for all of the investigated objects. Elliptic objects exhibit two valleys and two peaks around their circumference, which results in a minimum of four sampling points. These points must be aligned with the extremal points, i.e. with the axes of the ellipses. Along the height axis, 16 sampling points are chosen in order to obtain an accurate representation of the stiffness profile. Thus, the feature vector $\mathcal{F}$ consists of $16 \times 4 \times 3$ elements, with three elasticity coefficients $(\mathcal{F}^i, \mathcal{F}^I, \mathcal{F}^{II})$ for each point. Feature locations correspond to the grasp patterns, see Sec. 6.1.4 and Fig. 6.3b.

Features are normalized for scale-invariance, which generally improves the performance of detectors. Normalization parameters are stored alongside the feature vector. This allows for the identification of the best fitting database object after classification. First, a straightforward normalization scheme is applied to the linear coefficient (i) as follows:

$$\hat{\mathcal{F}}^i = \frac{\mathcal{F}'}{||\mathcal{F}'||}, \quad \text{with } \mathcal{F}' = \mathcal{F}^i - \text{median}(\mathcal{F}) \tag{6.4}$$

The same normalization is applied to the coefficients for the full deformation to obtain $\hat{\mathcal{F}}^I$. However, the calculation of the norm and the median is infeasible during exploration, since it requires the full feature vector. Therefore, a second normalization scheme is proposed, based on an estimate derived from a low number of feature points, which can be measured from the real object within a reasonable time:

$$\tilde{\mathcal{F}}^i = \frac{\mathcal{F}^i}{\overline{\mathcal{F}^i(x_8^+, x_8^-, y_8^+, y_8^-,)}} - 1 \tag{6.5}$$

The four normalization features are taken from the center of the object (feature index 8 out of 16) on opposing sides with two grasp patterns. Third, an even simpler normalization scheme based only on a single location x_8^+ is applied to obtain the feature $\tilde{\mathcal{F}}^{i,(1)}$.

Simulation and exploration of real objects are brought together at the level of stiffness features: Features are measured directly from real objects using a robotic arm, see also experiments in Sec. 6.4.2. Two opposing single point tips are attached to the two fingers of a gripper and touch the object according to the grasping pattern on two opposing points (dual-tip) as discussed in Sec. 6.1.4 and depicted in Fig. 6.3b. The number of force-deformation samples is much larger than during simulation, which allows for the compensation of sensor noise and other perturbing effects. Features are extracted in the same manner as discussed, by fitting linear/quadratic models to the obtained curve. The manipulator is moved one by one to the locations of the feature sampling points to obtain the full feature vector. The entire process is referred to as "haptic exploration" of an object and takes several seconds for each pair of features. Therefore, a full exploration of the object is only feasible during training.

During the classification stage, feature selection is used to extract the relevant features from the object, see Sec. 6.3.1.

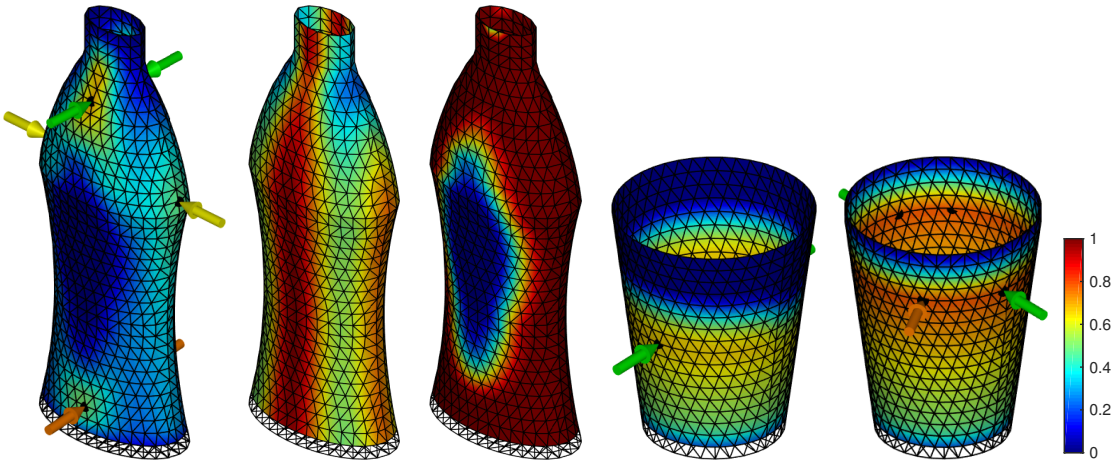
Furthermore, a speed-up is possible at the expense of accuracy by an alternative exploration scheme: Since the feature locations are arranged along two pairs of lines, the full feature vector for linear elasticity may be acquired by exploration along a curve or line on the object surface. For that purpose, the robot arm moves along an appropriate trajectory, sliding the contact points along the surface. At the same time, the gripper is in force control mode and pushes onto the object with a force F_p, adapting to the shape of the object. Curve-based exploration offers substantial speed benefits, since the arm is constantly moving and repositioning movements are avoided. The sliding speed must be chosen low enough to avoid dynamic effects from inertia and surface roughness. Friction between the single point contact and the object must be low – i.e. by using a smooth metal or wooden surface on the contact point. In order to avoid motion of the object during the exploration, it is explored from top to bottom while being fixed on the ground. Calculation of linear stiffness features requires at least two pairs of force/deformation per sampling location. Since the resting shape of the object (which corresponds to a force of zero) is assumed to be unknown, the curve-based exploration must be performed twice with two different values for the force F_p. Alternatively, a low-frequency offset can be added to F_p: Like that, deformation values for different forces are obtained for neighboring points on the surface during a single run. Stiffness estimation is based on the assumption that local stiffness remains almost equal within a close neighborhood.

6.2 Grasp planning for deformable objects

The goal of grasp planning is to find "stable" grasp configurations for a given object and gripper. The object pose is determinate and remains fixed with respect to the gripper during the grasping process. Grasp forces must be chosen large enough to avoid slipping or tilting of the object. To this end, surface friction, weight and center of mass of the object must be considered. At the same time, forces must be small enough to avoid the destruction of the object. Most importantly, the deformability of objects must be considered: The chosen grasp configuration must ensure that the deformation stays below a maximum value.

Here, a grasp planner for deformable objects is presented, which determines feasible grasping points based on the presented stiffness maps. Additionally, it integrates simple local geometry features to find stable grasp points.

Candidates of grasp configurations correspond to the two-finger grasp primitives on the entire surface, as introduced in Sec. 6.1.4. Note that these primitives ensure that forces are balanced and directed towards the major object axis. There is exactly one grasp configuration per surface point. The feasibility of each configuration is determined by a score $\mathcal{S}_G$. It is evaluated for each surface point $\mathbf{X}$ and consists of four components listed below. Each component has an associated coefficient $w_\star$:



(a) $\mathcal{S}_G$, cleaner **(b)** $\mathcal{S}_n$ **(c)** $\mathcal{S}_d$ **(d)** $\mathcal{S}_G$, cup **(e)** $\mathcal{S}_G$, cup-rim

Figure 6.4: Grasping scores for a complex bottle ("cleaner") and a plastic cup with/without reinforcement ring ("rim") on top. Feasible grasp patterns are located at maxima of the grasp score and are indicated by arrows. Figure adapted from [8].

Contact A preferably perpendicular object-gripper contact is desired to ensure a determinate object pose. The object surface normal $\mathbf{n}(\mathbf{X})$ and the direction of the applied force $\mathbf{F}$ should be preferably parallel. Otherwise, lateral forces would cause object rotations and slipping of the fingers on the surface. If there is a laminar contact between the finger and the surface, the finger surface normal should be parallel to $\mathbf{F}$. The contact score $\mathcal{S}_n$ considers a neighborhood $\mathcal{N}$ of points within d around the current surface point $\mathbf{X}$:

$$\mathcal{S}_n(\mathbf{X}) = \frac{w_n}{||\mathcal{N}_d(\mathbf{X})||} \sum_{\xi \in \mathcal{N}_d(\mathbf{X})} \left(1 - \sqrt{||\xi - \mathbf{X}|| d^{-1}}\right) \mathbf{F} \cdot \mathbf{n}(\xi) \tag{6.6}$$

Furthermore, a merely partial overlap between the finger and the object is penalized – as visible near the very top of the object in Fig. 6.4e.

Curvature Similarly, grasps on points within a locally flat or concave surface area are less susceptible to slipping. The curvature score $\mathcal{S}_c$ is calculated for each surface point $\mathbf{X}$ using the following steps:

- Select points within a neighborhood around $\mathbf{X}$
- Rotate this point set, such that the normal $\mathbf{n}_\mathbf{X}$ is oriented along the Z-axis
- Fit a second-order polynomial surface ($z = p_{20}x^2 + p_{02}y^2 + p_{11}xy + \ldots$) to the point set, using least-square minimization
- The curvature score is derived from the quadratic components of this polynomial:

$$\mathcal{S}_c = 1 + w_c \min\{p_{20}, p_{02}\}$$

If both coefficients are non-negative, the surface is flat or concave, and $\mathcal{S}_c$ is clamped (see below) to 1.

Height bias An object may tilt around the grasp axis while it is grasped due to a torque around this axis. Grasp configurations with two contact points, as used here, are especially susceptible to this problem. Static or dynamic torque may occur while the robot arm moves a grasped object. An uncontrollable tilting of the object is easily avoided, if the grasp point lies above the centroid of mass. A height bias $\mathcal{S}_h$ favors grasp points that lie closer to the top of the object:

$$\mathcal{S}_h(\mathbf{X}) = (1 - w_h) + w_h \frac{\mathbf{X}_z}{\max(\mathbf{X}_z)} \tag{6.7}$$

Deformation The object model provides a maximal acceptable deformation δ_{max} and a required grasping force F_G, see below. Based on the stiffness features $\tilde{\mathcal{F}}^i$, it is straightforward to obtain the expected deformation for each point. It must be ensured that deformation remains below a given upper threshold. A score is calculated as follows:

$$\mathcal{S}_d(\mathbf{X}) = -\frac{1}{2} \tanh \left[w_d \left(\frac{1}{\tilde{\mathcal{F}}^i(\mathbf{X})} \cdot F_G \cdot \delta_{max}^{-1} - 1 \right) \right] + \frac{1}{2} \tag{6.8}$$

The score is 1 for small deformations, and 0 for large ones. A soft transition is ensured by the tanh function, whereby the transition width is adjusted by w_d.

In the above equations, the surface point density is assumed to be constant. Coefficients $w_\star$ are chosen according to the physical capabilities of the gripper. For instance, a finger with a soft laminar rubber surface has a very high friction and thus tolerates relatively large lateral forces, while a metallic tip hardly offers any lateral stability. The size of the neighborhood around the contact point $\mathcal{N}_d(\mathbf{X})$ is adjustable, primarily based on the alignment accuracy that the grasping system provides. Here, we use $w_d = 5.0, w_n = 1.0, w_c = 5.0, w_h = 0.5, \delta_{max} = 5\,\text{mm}$ and $d = 1\,\text{cm}$ for the neighborhood size. The final grasp score is obtained according to:

$$\mathcal{S}_G = \prod_{\mathcal{S} \in \{\mathcal{S}_n, \mathcal{S}_c, \mathcal{S}_h, \mathcal{S}_d\}} \ell\left(\mathcal{S}\right) \tag{6.9}$$

The individual scores are clamped to the range $[0; 1]$ by the ℓ-function. During manipulation of a real object, the score is calculated online from the stiffness map and the geometry, after the correct object model has been identified, see Sec. 6.3. Typically, multiple feasible grasp configurations with a large enough score (e.g. $\mathcal{S}_G > 0.5$) exist. These configurations represent the set of possible goals for the arm controller. To find the optimal grasp configuration, the corresponding costs of the arm trajectory are considered.

The required grasping force F_G as well as the acceptable deformation δ_{max} are not determined by the planner itself. Instead, these values are specified semi-automatically as param-

eters of the object model. To determine F_G, first of all, the friction coefficient μ between the gripper and the object surface must be considered. Its value depends on the materials that come into contact as well as their surface roughness. For a contact steel-polyethylene, $\mu = 0.2$, while contacts of many materials with rubber exhibit $\mu \approx 1.0$. If the contact point is located below the center of mass, in addition, the friction torque must be considered to avoid tilting of the object. Second, the weight of the object must be estimated, based on the volumetric model, the density of the material as well as the contents within the container objects, which is most significant for plastic containers. An upper limit of the weight of the contents is calculated based on its density and the volume of the container. Feasible values of the grasping force can be calculated automatically during training of the generation of the object database. Yet, additional assumptions are necessary – such as the typical surface roughness of materials, the fill level of liquids in containers and correspondingly, the acceptable deformation δ_{max}. Such prior values are given manually per object class.

In the presented grasp planner, object geometry is considered based on simple geometry features. This simplification is applied in order to demonstrate the effects of integrating deformation models in a straight-forward way. Even though the grasp planner works with the presented objects, in general, more aspects of object geometry must be considered. Complex grasp patterns – such as those from a humanoid hand – require more sophisticated models. Grasp planning is an active field of research, even for rigid objects, and many approaches have been presented for different scenarios. GraspIt [73] – a state-of-the-art system applicable to many kinds of objects – determines stable grasps by a Monte Carlo simulation: Numerous grasp configurations are generated randomly, and a physics simulator is used to verify their stability. It is not the goal of this work to propose a new, versatile grasp planner. Instead, the proposed score based on the stiffness maps could be integrated into these existing methods.

6.3 Haptic object classification using stiffness features

Object classification is a second application of the proposed stiffness features. In this section, a method for feature selection, online exploration planning and classification based on classification trees is presented. Following, relevant scenarios for haptic classification are discussed, and a process for joint visuo-haptic classification is outlined.

6.3.1 Feature selection and exploration planning

Feature selection methods are widely used in machine learning – mainly in order to speed up processing with complex detectors, see e.g. [19], sometimes also to improve the classification performance. They are based on various principles, such as the mutual correlation of features with the predicted variable, correlation between features, subspace methods or information gain. One straight-forward approach is to apply subspace decomposition with principal component analysis (PCA) to the feature space. In [1], a method for a greedy feature search based

on the a posteriori detection accuracy is proposed. Neighboring features are often highly correlated, such that one "representative" for each neighborhood of features is sufficient. The stiffness features clearly show that property, see also Fig. 6.7a. Feature statistics rely on data available during the training process, which is why a large number of representative samples are required. Many approaches require the full feature vector as an input, i.e. each element must be known before the selection process.

In point-based haptic exploration, one is faced with the situation that each element of the feature vector needs to be obtained individually by touching the object at its respective location. It is not possible to simply circumvent this limitation by a multi-contact sensor, since local stiffness is influenced by nearby touch points. The acquisition of each element of the feature vector comes with costs of several seconds, and therefore, it is the goal of feature selection to identify a possibly low number of features (e.g. less than 10 elements), which are sufficient for reliable classification. On the other hand, the speedup of classification or other computational processes is of lower interest: There is enough time for computations while the robot is moving, and the dimensionality of the stiffness feature is small anyway.

Classification trees have been identified as a favorable approach for haptic classification, which simultaneously provides feature selection, exploration planning as well as classification. They are trained rapidly online, given a subset of classes from a previous detector that provides a prior, see Sec. 6.3.3. Feature selection is performed even twice: First, during the generation of the tree (training phase), a subset of features is identified, which is sufficient for accurate distinction of all given class candidates. Consequently, with a lower number of classes due to a stronger prior, the number of required features is reduced. Second, during classification, only those feature elements are acquired, which lie along the chosen path through the tree. If an object is clearly distinguished by a single feature, only a single exploration step is performed. It should be noted, however, that other methods for classification, such as SVM [17], exhibit a better classification performance than classification trees.

In the notation of classification trees, each object type is a *class* of which several *observations* exist, corresponding here to variations of material, geometric details, etc. The goal is to predict the class based on *prediction variables*, which are elements of the feature vector. Here, we use the MATLAB implementation of binary classification trees, which is based on the CART method. (For an introduction, see [61].) Binary trees split the data into two subsets at each level, based on the comparison of a single predictor variable with a threshold. The construction of the tree from training data is performed as follows:

1. Start at the root node with all observations.
2. Find a split between the observation set, which minimizes the sum of the *impurities* of the two child nodes. The split is based on comparison of a single element k of the feature vector with a scalar θ_k. Both the scalar and the element index are obtained by optimization, minimizing the impurity. Intuitively, the two subsets in the child nodes represent the two most dissimilar subsets of observations.

3. Repeat the above step recursively for each child node. Stop splitting once a stopping criterion is reached, e.g. all observations in a node belong to the same class. Leaf nodes are associated with a class.
4. Prune the tree to a smaller size by minimization of the misclassification error. An estimate of the error is obtained by cross-validation.

The impurity of a node is determined by Gini's Diversity Index $gdi = 1 - \sum_c p^2(c)$, were the sum is performed over classes c that reach the node and their respective probability p (i.e. share of observations) at the node. A node with only a single class exhibits the minimal gdi of 0. The split parameters (k, θ_k) are determined by optimization:

$$(k, \theta_k) = \underset{k' \in \mathcal{F},\, \theta \in \mathbb{R}}{\operatorname{argmin}} \sum_{n \in n_1, n_2} \left[1 - \sum_{c \in c_n(k', \theta)} p^2(c) \right] \tag{6.10}$$

Where $n_{1,2}$ are the two child nodes and c_n is the set of classes that arrives at node n, which depends on the split parameters.

For classification, the tree is traversed from the root node until a leave with an associated class is reached. At each node, one element from the feature vector must be measured. This is the element that provides most information gain, given the existing knowledge. Its value is compared to the learned threshold and determines whether to continue to the left or right child node. The classification uncertainty is reduced, and the next most relevant feature element is obtained. Due to these properties, the classification tree also serves as an exploration planner. Each node is associated to one feature location (one element of the feature vector) such that the path through the classification tree determines the exploration strategy of the object. All knowledge from previously acquired features is integrated at any location within a path. Feature elements are only measured as they are "requested" by a node, while all irrelevant features (i.e. those that are not on the chosen path) are ignored. Note that there may also be decisions based on a previously acquired feature with a different threshold. The exploration step can be skipped in such cases. In [26], a method is presented that deliberately prefers already acquired features to reduce classification time.

6.3.2 Scenarios for haptic exploration

Haptic exploration provides additional object knowledge in the context of (visuo-)haptic classification and detection. In the following, some scenarios are presented, in which it is feasible to obtain information relevant for manipulation by haptic exploration.

Object variants There are numerous variants of objects that cannot be distinguished by visual detection due to a similar appearance or unspecific visual models. These variants include different materials, wall thicknesses, reinforcement structures, dents or other geometric details. Each variant is represented by one or several haptic models. For

instance, reinforcement rings/rims on the top of cups or bottles result in significant differences of local stiffness and consequently in different grasp plans, see Sec. 6.2. Models are distinguished by haptic exploration as outlined in Sec. 6.3.1. In many cases, a low number of features suffices for discrimination.

Object state For a given object identity, there may exist multiple internal discrete or non-discrete object states. In case of a bottle, states may refer to the cap (open/closed) and the fill-level of the contents. State changes result in changing haptic properties, such that haptic exploration can be used to identify the state value. The relations between states and haptic properties are either predicted by models, see Sec. 6.1.5, or they are learned from the real object. Since properties may only differ for large deformations, exploration speed should be lowered and the number of exploration steps should be as low as possible. Therefore, the object state is determined subsequent to class-level recognition. Also, a (visual) verification method might have to be used to avoid destruction of the object.

From the detected object state, certain limitation for grasping and manipulation plans may be derived. For example, depending on the state of a bottle, the following constraints must be considered:

- A closed bottle filled with water is relatively stiff and heavy. The grasping force can be larger than required, ensuring a stable grasp. The grasping point is determined based on geometry and does not need to be very accurate. There are no constraints on the trajectory, and fast manipulation is possible.
- Grasp plans for open bottles filled with liquids require additional considerations. If such objects are grasped at points of low stiffness, the resulting deformation might cause an overflow of the contained liquid. Safe grasping configurations are thus limited to stiff regions of the object. Also, to avoid shaking, the grasp and the subsequent motion trajectory should be executed with a lower speed. Furthermore, the bottle must always point up, imposing further constraints on the trajectory.
- An empty bottle is much lighter and is thus grasped with a relatively small force. The resulting deformation remains low on most parts of the surface, such that there are fewer constraints on the grasp points.

Model refinement Simulation-based models typically do not represent all details of a real object. A new, more accurate model may be derived from a generic object model by refining model parameters, geometry and local stiffness. More accurate models improve the performance of grasp planning and object classification. First, for parametric models, the continuous parameters – such as wall thickness and elastic modulus – are optimized to improve the fit between simulation results and observation. Also, linear combinations of models may be created. Second, simulation-based models are refined by integrating sparse measurements from real objects. To this end, local stiffness values

are measured at predefined or critical locations on the object. The simulation-based stiffness map is corrected at the respective locations. In between the sparse samples, an interpolation scheme is applied.

Haptic object classification with prior As discussed in Sec. 6.3.3, vision-based object classification should precede haptic classification or recognition. Results from vision serve as a prior, which generally exhibits uncertainty and comprises several possible object classes (candidates). Furthermore, one "visual" class may correspond to multiple haptic models. Robots require a higher classification certainty than other applications, since incorrect decisions may result in destruction of objects. The correct model is determined by haptic exploration, see the experiment in Sec. 6.4.2. A haptic exploration plan must be created, which allows for the optimal discrimination between the candidates. The classification result can be used, for instance, to find feasible grasp configurations, or as an input to a semantic planner.

6.3.3 A process for visuo-haptic classification

In this section, a process for joint visuo-haptic classification or recognition feasible for robotic applications is presented. It consists of three stages: The first stage relies on data collected using "passive" vision-based methods, which work remotely from the current perspective or trajectory. This allows for parallel recognition of many objects in a room or scene, and often one major constraint is the acceptable runtime of classification algorithms. Next, an optional "active" visual exploration stage is started, which actively controls the robot trajectory to acquire information about an object. One example for active exploration is driving around an object to build a complete 3D model. Finally, the robot explores the object in the haptic modality – i.e. it collects data by touching or manipulating the object. Active exploration methods, both in vision and haptics, require the robot to focus on a single object or a small group of nearby objects. The duration of data acquisition itself poses a major constraint. Therefore, it is essential to consider the information acquired in the earlier, less costly stages for the planning of subsequent stages with higher effort. Haptic exploration should be minimized as much as possible (e.g. by reducing the number of touch points), or even skipped when possible. A purely haptic exploration system is infeasible, except in extreme environments, such as in muddy water, or in very small workspaces.

Visual recognition (for specific objects) or classification (on a class level) is based on one of the methods discussed in Sec. 2.3.5. There are significant differences in the result characteristics of these methods, with strong implications for the haptic classification process:

- Using local texture features, objects of completely different shapes and functionality (semantics) may be confused due to common design patterns or logos. While these objects are visually similar by some measure, they might require entirely different manipulation

and grasping strategies. If an object does not exist in the database, the match with the closest texture features may yield a completely unrelated object.

- Contour-based or other class-level recognition approaches do not provide the specific geometry, shape or size of an object. This is done by construction, since the detector is generated and trained for generalization. Training is performed with multiple variations of size and shape, such that a classification of a previously unseen object is also possible. Consequently, a visual class may correspond to multiple different haptic models, which usually belong to the same semantic class though.
- Geometry-based classification typically yields objects with a similar shape, regardless of the object class. Similar shapes may still imply different haptic properties – for instance due to the material: There are cups built from ceramics, glass, thin steel sheets, plastics or paper. The latter materials result in a much softer and lighter object, due to considerably lower elasticity moduli. Furthermore, stiffness depends on the wall thickness, which is hard to detect by vision. Reconstruction methods fail, since the inner surface is hard to observe, and depth noise is often higher then the wall thickness. Finally, subtle geometric detail, such as dents or reinforcement structures, are hard to see, but result in considerable changes in local stiffness. Note that these effects occur with almost any visual detector, if there is no additional knowledge about an object.

In general, haptic classification may rely on prior information from vision, which reduces the number of possible object candidates considerably. The remaining candidates, however, may be very diverse concerning their haptic properties. Obviously, the distinction of large differences in stiffness or even shape is relatively simple in the haptic modality and requires only a low number of measurements. For more subtle differences, more precise and consequently more time-consuming measurements may be required. However, two objects with a high "haptic similarity" will also be manipulated in a similar way.

Information about objects is stored in various databases. First, a global object database contains representations of thousands of objects and provides the basis for classification. It is created by a central instance to provide generic capabilities of object classification to robots. The required object models depend on the application area of the robot (such as household, industrial production, service, outdoor) and should allow both for class-level recognition as well as for the detection of specific objects. In case of a household robot, the database consists for instance of the products available in the national supermarkets. Feature extraction and training can be performed by the creators of the database. In order to construct features for various visual detectors, object models must comprise geometry and texture. Besides visual representations, haptic models are associated to each object. These models include stiffness maps, stiffness features (Sec. 6.1.6), or – for less detailed models – haptic tags (see Sec. 4.1). Haptic properties are obtained by physics and FEM simulation from the geometric model as outlined Sec. 6.1. Specific visual object representations for recognition are associated with one specific haptic model. Note, however, that multiple results may be provided by the detector

in the first stage. Representations on class-level are associated with a set of haptic models, as discusses for the different cups.

Second, a robot builds local databases, which store visual and haptic knowledge of objects in a specific scene and accumulate local scene knowledge over time. It comprises of objects that are not present in the global database, additional or more specific information as well as context information from the environment, such as the last known position of an object. Building new object models is a time-consuming process: For the visual modality, a complete geometric and texture-based model is acquired by driving around the object. Haptic properties must be determined by touching the object at many points. Depending on the application, simplified models may be used, as presented in Chapter 4. Once a model comprising visual and haptic information has been built, however, rapid recognition of the respective object is possible. For instance, visual recognition can be performed based on local texture features extracted from the texture model. The haptic model or haptic tag is loaded together with the visual model, rendering the haptic classification stage unnecessary in case of unique classifications. Ambiguous classifications are less likely than for the huge global database, and they could even be resolved using the scene context. That way, the classification time is reduced, and the robot becomes faster in familiar scenes. New objects may eventually be shared with the global database, if they are of general interest.

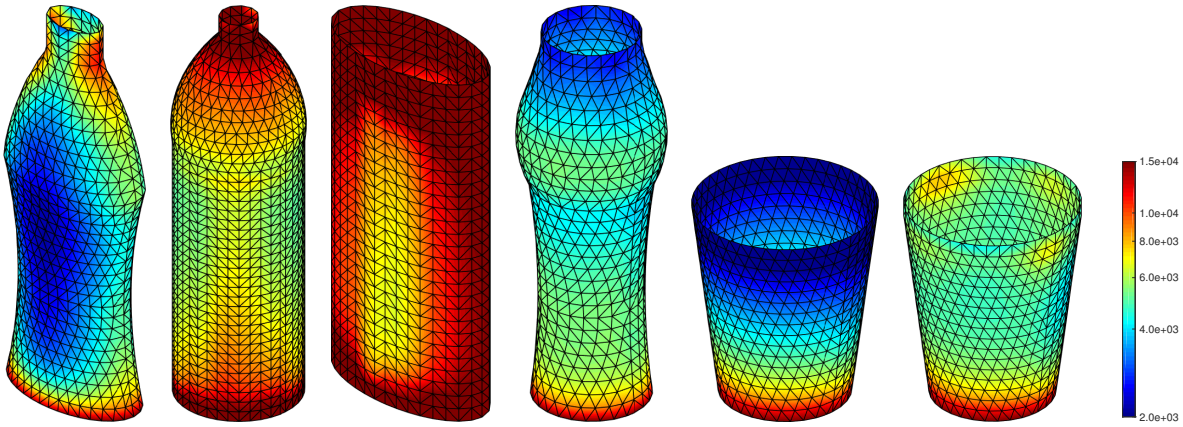
6.4 Results and experiments

This section presents results and experiments for the proposed methods. First, the simulation-based process of object model generation, feature extraction and classifier training is outlined. Next, stiffness data are acquired from various real objects and used for testing of the trained classifiers. Finally, the grasp planner is demonstrated on real objects.

6.4.1 Simulation-based models

An object database is created based on artificial, parametric object models as outlined in Sec. 6.1.2. The database includes various bottles, plastic cups and other containers. Geometries and stiffness maps of the most important classes are shown in Fig. 6.5. Object variants are generated automatically by variation of several object parameters, such as diameter, height as well as the curvature and height of object sections, see also Fig. 6.3a. For instance, the object shown in Fig. 6.5a and Fig. 6.5d consists of three sections stacked upon each other. Other variations include the wall thickness, material parameters as well as hidden geometric structures or details. There are 50–100 variants within each class.

Stiffness maps and features are calculated as outlined in Sec. 6.1.6 with FEM simulation based on the VegaFEM library [93]. It can be seen that local surface curvature has a strong influence on local stiffness, see e.g. Fig. 6.5a. Strong “support structures” in areas with a high curvature result in a high stiffness, while large, flat areas are rather soft. Rotational



(a) Cleaner **(b)** Bottle **(c)** Elliptic **(d)** Smoothie **(e)** Cup, no rim **(f)** Cup, rim

Figure 6.5: Stiffness maps of exemplary models in the database (with unit $\frac{N}{m}$). Figure adapted from [8, 9].

solids like the cup in Fig. 6.5e are expected to show a local stiffness that gradually changes as a function of height. The bottom part is most stable, since it is supported by the base plate. Adding a stiff structure on top of this object – such as a small reinforcement ring ("rim") – results in a significant increase of local stiffness on the upper part, see Fig. 6.5f. Note, however, that the stiffness map is not completely rotationally symmetric. This error, which is also observed in Fig. 6.5b, is caused by inaccuracies of the FEM simulation. The tetrahedral elements are relatively ill-conditioned due the low wall thickness. At the same time, the mesh resolution cannot be increased significantly due to memory limitations. Improvements can be expected with different types of finite elements, which would require another library, or the use of surface-based models, such as thin shell models [13]. The latter approach, however, would limit the object types that can be modeled.

Normalized stiffness features $\tilde{\mathcal{F}}^i$ are extracted from the stiffness maps. In order to verify their distinctiveness, features of all classes are cross-correlated using NCC in Fig. 6.6. Most features exhibit a small mutual correlation coefficient, indicating that they are relatively distinct. It can be expected that objects can be well distinguished based on these features. However, the feature vectors of elliptic bottles (bottom right in the figure) show a high mutual correlation. Even though these objects do exhibit characteristic differences in their stiffness maps, see Fig. 6.5a and Fig. 6.5c, the features show great similarities. A cross-validation within the database showed that it is still possible to distinguish these features – yet transferability to real objects is critical, see below.

Classifier training Next, the generation of the classification tree, see Sec. 6.3.1 is analyzed. In Fig. 6.7a, features $\tilde{\mathcal{F}}^i$, see Eqn. (6.5), are shown for four different object classes. The intraclass variability of the feature vector is indicated by error bars. It is evident that some

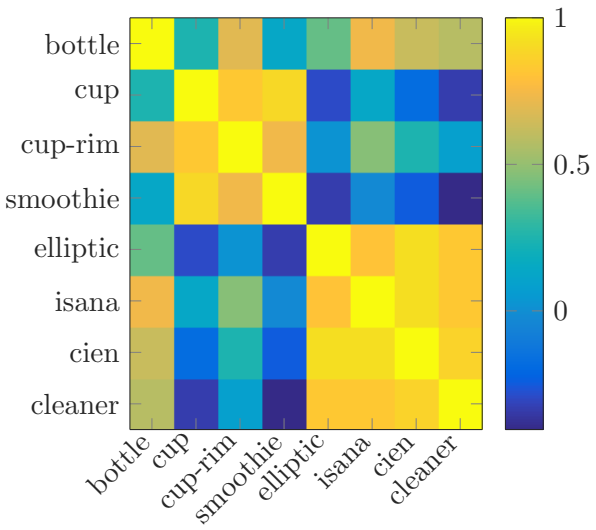


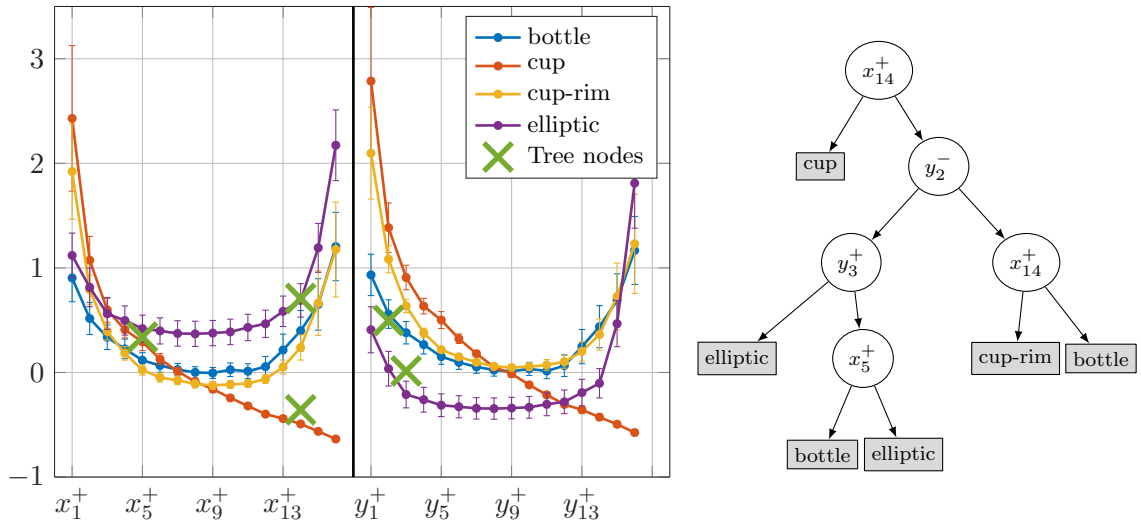
Figure 6.6: Correlation between simulation-based feature vectors $\tilde{\mathcal{F}}^i$ of different classes

elements of the feature vector allow for a good distinction of objects, namely those with a large separation gap between classes. On the other hand, elements with almost identical values for different classes are unsuited to discriminate different classes. The classification tree obtained for this case is depicted in Fig. 6.7b. Obtained split parameters (k, θ_k), see Eqn. (6.10), are marked in the plot. The selected features allow for good discrimination of object classes. The first selected feature x_{14}^+ already separates "cup" from other classes, based on its low stiffness on the top. The second selected feature y_2^- lies near the blue curve and thus splits the bottle class in two. This is why "bottle" appears in both subtrees derived from this feature node.

Additional classification trees are trained for combinations of two or three classes, see Table 6.1. As expected, only a very low number of features is needed for discrimination. This kind of classification task is relevant for instance when an object geometry has been identified with high certainty by a preceding classifier, but there are still two or three candidates, representing different geometric details, such as "cup/cup-rim". Also, a classifier is trained based on all classes from the database, which results in a larger tree with 9 feature elements. The obtained trees are used for classification of real objects, see below.

6.4.2 Exploration of real objects

In order to explore real objects, a haptic exploration system is built with a two-finger Schunk gripper mounted onto a KUKA LWR robot arm. Each finger is equipped with force sensors and a wooden tip, see Fig. 6.8. Like that, it is ensured that objects are touched at two single points with low lateral friction. The gripper is opened, positioned around the object and then slowly closed, until a maximal force is reached. Contact with the object is detected by an increasing force value. The position and force readings acquired during the closing process



(a) Features $\tilde{\mathcal{F}}^i$ from simulation

(b) Decision tree

Figure 6.7: *(a)* Simulation-based stiffness features for four relatively distinct classes. Error bars show the standard deviation of features to indicate the intraclass variability. *(b)* Decision tree trained for this subset of classes. Its nodes (i.e. the feature indexes and decision thresholds) are also marked in the plot *(a)* with a ×-symbol. (Features on x^-/y^- are mapped to x^+/y^+.)

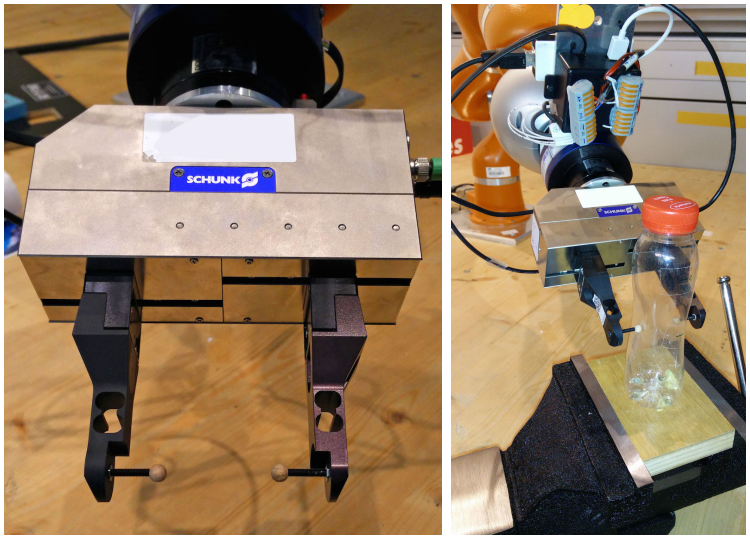


Figure 6.8: The haptic exploration system with a KUKA LWR arm and a two-finger gripper. Objects are touched/pressed with an increasing force by two sensor tips from opposing directions at the feature locations, see Fig. 6.3b.

#	Classes	Feature	Loss	Selected features k [Threshold θ_k]
1	cup, cup-rim	$\tilde{\mathcal{F}}^i$	0.0%	x_{12}^+ [-0.28]
2	bottle, elliptic	$\tilde{\mathcal{F}}^i$	0.0%	y_3^+ [0.01]; x_5^+ [0.34]
3		$\tilde{\mathcal{F}}^{i,(1)}$	2.9%	y_3^+ [-0.12]; y_3^- [-0.36]; x_2^+ [0.21]
4	bottle, cup, elliptic	$\tilde{\mathcal{F}}^i$	0.5%	x_{14}^+; y_3^+; x_5^+
5		$\tilde{\mathcal{F}}^{i,(1)}$	2.7%	x_{10}^+; y_3^+; y_3^-; x_2^+
6	bottle, cup, cup-rim,	$\hat{\mathcal{F}}^i$	2.9%	7 nodes, 7 features
7	elliptic, smoothie	$\tilde{\mathcal{F}}^i$	1.2%	7 nodes, 6 features
8		$\tilde{\mathcal{F}}^{i,(1)}$	3.7%	11 nodes, 9 features
9	cien, isana, cleaner	$\tilde{\mathcal{F}}^i$	0.8%	x_2^+ [1.15]; x_{16}^+ [3.45]

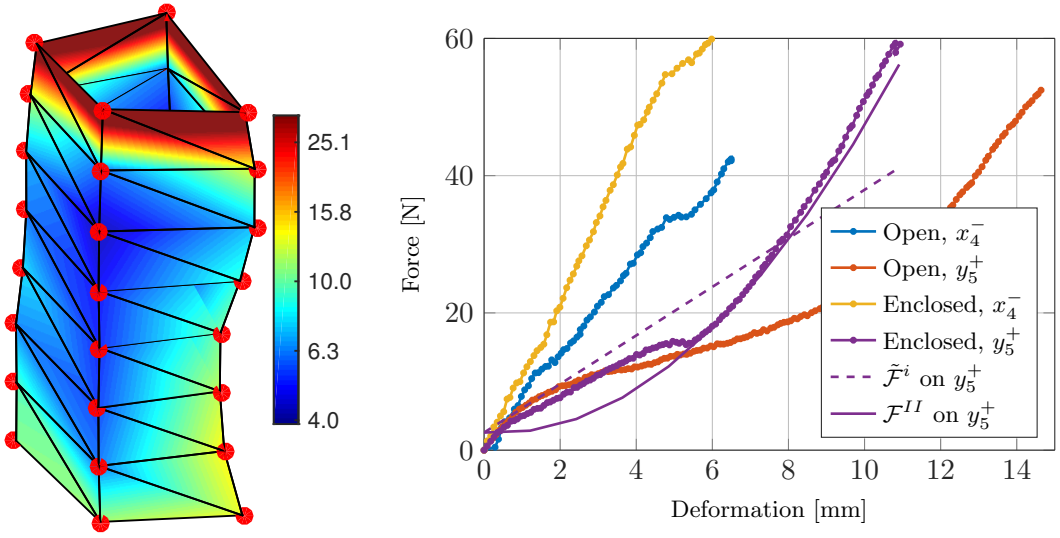
Table 6.1: Decision trees trained with different subsets of classes from the object database

(a) Bottle, smoothie **(b)** Cut-bottle, cup-rim, cup **(c)** Cien, isana, palmo, cleaner

Figure 6.9: Real objects used in the experiments for haptic exploration and classification. Four plastic cups in *(b)* were put into each other, since the force sensor was not sensitive enough for a single one. The cut bottle also belongs to the "cup" class. Bottles in *(c)* all belong to the "elliptic" class. Figure adapted from [9].

provide the deformation-force curve for the respective point on the surface, see Fig. 6.10b. In our system, force/position readings are acquired with about 30 Hz, and the fingers move with a closing speed of 5 mm/s. Data from both fingers are measured independently, but a roughly symmetric force on the object is ensured. To this end, the position of the gripper is corrected such that both fingers touch the surface roughly at the same time. The process is repeated for all feature points, see Fig. 6.3b, to obtain the full feature vector $\tilde{\mathcal{F}}^i$ as in simulation. Curves for a pair of opposing points are acquired simultaneously by the two fingers of the gripper. Stiffness features are calculated from the measured deformation-force curve as outlined in Sec. 6.1.6. Fig. 6.10a shows the stiffness map obtained from the "cien" bottle in Fig. 6.9c.

In this setup, object poses are pre-known. Objects are fixed on their bottom side to prevent motion during the exploration process, see Fig. 6.8. This approach ensures more consistent measurements and allows measuring non-symmetric object stiffness. In realistic scenarios,



(a) Stiffness map [N/mm]

(b) Measured deformation-force curve for two points

Figure 6.10: Exploration results of the "cien" plastic bottle, sampled at feature locations according to Fig. 6.3b. The strain-stress curve is given for two points on the curved right side (x^+) and the flat frontal side (y^+) of the object. Note the pronounced differences of stiffness. If the bottle is closed, enclosing the air inside, the object becomes much stiffer for large deformations.

fixtures cannot be used. Yet, object motion would still be limited, due to a lower number of touch points and the symmetric grasp pattern. Moreover, the object pose should be tracked, for instance with ICP, in order to refine the location of touch points. Note that only stiffness features are considered in the following experiments. Additional haptic features as considered in the "haptic tag" – such as geometry, shape or contact events – may improve performance.

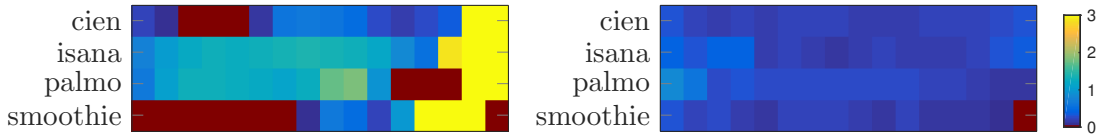
Classification Haptic object classification is performed based on measured stiffness features and the trained trees. First the two feature pairs used for feature scaling according to Eqn. (6.5) are obtained. The following exploration steps are determined by the classification tree – for instance, with the tree shown in Fig. 6.7b, a cup with a soft top could be distinguished from the other objects based only on feature x_{14}^+. Otherwise, exploration continues according to the right sub-tree. The structure of the trained tree depends on the class candidates selected by the prior. Wrong priors result in incorrect classifications, since there is no "other" class in the decision trees.

Table 6.2 shows the classification results for all objects depicted in Fig. 6.9 with different trees from Table 6.1. Tree 7 is based on the largest training set and shows a good classification performance. The elliptic bottles are all detected as the "elliptic" class, showing that generalization of features works well. Only the smoothie object is not detected correctly, despite its characteristic stiffness map. Tree 9, which is trained to distinguish different elliptic

Object	Classification tree/feature type					
	4/$\tilde{\mathcal{F}}^i$	**5**/$\tilde{\mathcal{F}}^{i,(1)}$	**6**/$\hat{\mathcal{F}}^i$	**7**/$\tilde{\mathcal{F}}^i$	**8**/$\tilde{\mathcal{F}}^{i,(1)}$	**9**/$\tilde{\mathcal{F}}^i$
cup	✓	✓	✓	✓	✓	–
cup-rim	–	–	✓	✓	✓	–
cut-bottle	×	×	✓	✓	✓	–
bottle	✓	×	✓	✓	×	–
smoothie	–	–	×	×	×	–
isana*	✓	×	✓	(✓)	×	(✓)
cien*	✓	×	✓	✓	✓	×
cleaner*	✓	×	✓	✓	×	✓
palmo*	✓	✓	✓	✓	✓	–

–: Classifier not trained for object; ×: Detection failed;
✓: Detection successful; (✓): Detection successful in 2 out of 3 experiments

Table 6.2: Classification results of haptic exploration on real objects with six decision trees from Table 6.1. Objects marked with a star (*) are of class "elliptic".



(a) Real objects **(b)** Simulation

Figure 6.11: The values of the quadratic feature $\mathcal{F}^{II}$ are compared for the same object in the two states "open" (o) and "closed" (c) (with air enclosed inside). The depicted difference $\mathcal{F}_c^{II} - \mathcal{F}_o^{II}$ is expected to be positive on most feature locations. Values are shown within $[0, 3]$, whereby any value ≤ 0 is shown in red.

bottles, shows a weak performance. Obviously, objects with more subtle differences cannot be distinguish with the proposed features, as already inferred from Fig. 6.6.

Additionally, the proposed schemes for feature normalization are compared: Feature $\tilde{\mathcal{F}}^i$ using simplified normalization (tree 7, Eqn. (6.5)) performs similarly well as feature $\hat{\mathcal{F}}^i$ (tree 6, Eqn. (6.4)), which requires the full feature vector for normalization and thus cannot be used with sequential exploration planning. Normalization based only on a single value ($\tilde{\mathcal{F}}^{i,(1)}$), however, results in an insufficient classification performance (tree 8). The training process already unveils the weakness of this feature, see Table 6.1: The loss is consistently higher, even though a larger number of features is selected.

Object state Next, deformation behavior is analyzed when the content inside of the bottles is considered[1]. In principle, the quadratic feature could just be added to the feature selection process, yet for practical reasons, a separation is feasible. Here, two different object states

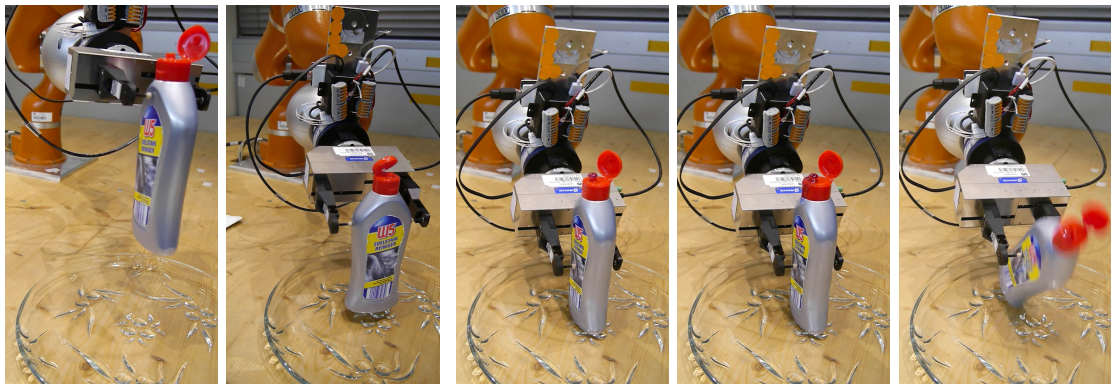
[1]This experiment was conducted together with Jingyi Xu

"open" and "closed" are compared, whereby in the latter case, counterpressure from the compressed air enclosed inside the object is considered, see Sec. 6.1.5. In an opened bottle, air can escape, such that there is no pressure to the inside wall. The resulting difference is seen from the plot in Fig. 6.10b, which depicts the measured deformation curve for both states on two surface points. The curve of the (softer) point y_5^+ shows a stronger quadratic component for the closed state (violet) than for the opened state (red). Additionally, the fitted models for features $\tilde{\mathcal{F}}^i$ and $\mathcal{F}^{II}$ are shown. The (harder) point x_4^- shows a difference between the two states (yellow/blue) in the linear component. Since the difference is smaller, a robust distinction of the states is infeasible at this point. Obviously, the counter-pressure from compressed air has a much stronger effect on large and flat surface areas.

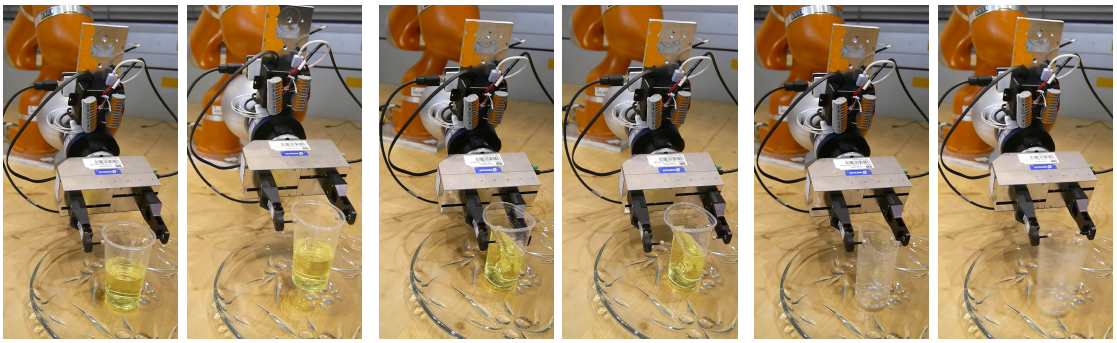
The behavior of the quadratic feature components is analyzed for multiple real and simulated objects. Fig. 6.11a shows the difference of the $\mathcal{F}^{II}$ feature for the two object states within the relevant y^+ region. In general, it is expected that a significant difference is observed within soft areas, i.e. in the mid/lower region of y^+. The observation of the "isana" and "palmo" objects adhere to this expectation, while for the "cien" object, the difference is largest within the upper part. The high difference towards the top is attributed to a secondary effect, i.e. the stronger support of a closed lid. For the "smoothie" object, other effects seem to dominate within the lower region, which is also quite stiff. In Fig. 6.11b, the same comparison is shown based on simulated data. While simulation confirms the general trend of a larger quadratic feature value, it is incapable to identify optimal feature locations. Also, the magnitude of difference determined by simulation is much smaller. Thus, unless a more realistic modeling is available, the optimal feature location for discrimination of the two object states should be determined by training on the real object. An approach for discriminating object states with a trained classifier based on a 6D tactile feature has been proposed in [20].

Grasping Finally, a grasping experiment, see Fig. 6.12, is conducted with objects "cleaner", "cup-rim" and "cup", for which predicted grasp scores and plans are shown in Fig. 6.4. The two predicted grasp configurations for the (open) cleaner, near the lid and on the side, work successfully, see Fig. 6.12a. The grasp on the side, however, is sensitive to alignment errors, as can be seen from the low curvature score $\mathcal{S}_c$. A grasp with 20 N at the (soft and flat) center of the object on the other hand, see Fig. 6.12b, results in an overflow of the liquid contents and in falling over of the object. For "cup-rim", the predicted grasp on top also succeeds, see Fig. 6.12c. Fig. 6.12d illustrates that a grasp with the same force in the center results in a large, permanent deformation of the cup. Without the rim on top (Fig. 6.12e), a force of only 5 N results in a large deformation, and the object slips out of the gripper, even if it is empty.

Contrary to classification problems, correct scaling is essential for grasp planning. If grasping is performed after classification as outlined above, the correct model must be identified from the set of models in the detected class. The best model can be selected based on feature elements used for scaling or any other "stable" feature. Note that the same two-finger gripper with two tips as in the previous experiments is used. Other finger designs – such as



(a) Cleaner (30 N, top/side) – ✓ **(b)** Cleaner (20 N, center) – spills over, falls over – ✗



(c) Cup-rim (10 N, top) – ✓ **(d)** Cup-rim (10 N, center) – ✗ **(e)** Cup (5 N, top) – ✗

Figure 6.12: Grasp configurations calculated by the proposed grasp planner are verified in an experiment, see also Fig. 6.4. Predictions of stable and unstable locations are confirmed. Figure adapted from [8].

fingers with a soft laminar material – might work better with the presented object. Yet, note that each mechanical design has its own strengths and limitations, which must be considered accordingly by parameters in the planning process.

6.5 Summary

In this chapter, a model for thin-walled deformable objects is presented and used in the context of haptic object classification and grasp planning. This haptic object model represents the local stiffness of the object surface (local stiffness map) and is obtained by simulation. To this end, generic grasp patterns are applied to the object in an exhaustive fashion, and the deformation is determined by the Finite Element Method (FEM). Pronounced locations in the stiffness map, which can also be measured on real objects, are referred to as stiffness features. Many variants of an object are generated in an automated fashion from surface geometry models in order to generate large object databases as required by machine learning methods. These variants include variations of surface geometry, material parameters, subtle

geometric details as well as the contents within objects, such as enclosed air. The approach is especially suited for objects with a distinct stiffness profile on their surface, such as plastic bottles, cups and other containers.

A grasp planner is presented, which considers the deformation in order to determine valid grasp points. It calculates a grasp score for the object surface, based on the stiffness map as well as stability criteria of the local geometry. Current approaches, such as [73], focus on complex object geometries and grippers, but they do not consider deformable objects. The feasibility of the detected grasp points is demonstrated, and the need of considering stiffness during grasping is illustrated.

Furthermore, haptic object classification is considered as a subsequent step after a vision-based classifier. In order to feasibly extract stiffness features from a real object, a low number of relevant features must be determined. This feature selection process is performed by classification trees, which are at the same time used to create an exploration plan and to detect the object class. Experiments are presented, demonstrating the performance of the proposed classification scheme. Multiple small sets of object classes are distinguished by a low number of local stiffness features. An adapted and reliable manipulation or grasping plan can be determined from the object class. However, the detector is incapable to distinguish objects with similar stiffness profiles. This is mainly due to the limited accuracy of simulation-based features compared to data from real objects.

7 Conclusion and Outlook

7.1 Conclusion

Autonomous robots that operate in environments built for humans require multimodal sensing skills and rely on multimodal environment models for task planning. For the interaction with objects, the visual and haptic modality are of the greatest relevance. In Chapter 3, a visuo-haptic sensor has been presented, which uses a standard camera to obtain haptic data, i.e. force, pressure and contact shape during manipulation operations. The camera measures a deformable element, which is part of the robotic actuator. The deformation of this element is converted to a force/pressure profile based on its known deformation model. This concept is demonstrated with deformable elements made of plastic or rubber foam, which exhibit a laminar contact with objects, as well as with a beam-based structure, which provides a full 6D force-torque vector at a single contact point. Visual observations from the object, the scene or the components of the robot may be acquired by the same camera and are thus naturally coherent with haptic data. This is an important advantage over existing haptic sensors with optical readout, which rely on an optical sensor inside of a deformable structure. Dedicated sensor systems are replaced by a low-cost camera, which may be co-used by other tasks, reducing costs and system complexity. Integration of visual data is important to observe the reaction of an object during manipulation, and also to verify the state of the manipulator itself. Experiments are presented for grasping applications with the sensor mounted on a two-finger gripper as well as for haptic mapping with a mobile platform. The accuracy of the sensor is shown to be good, by comparison with an industrial force sensor. One major limitation of the presented concept is that the deformable element must not be occluded. Depending on the camera position, this results in constraints of the end-effector pose and of acceptable object geometries. Furthermore, the measurement frequency is limited by the camera frame rate and by the damping characteristics of the deformable element.

Perception also includes the process of fitting high-level models to raw sensor data. Data and level of detail provided by a model must be adapted to the available sensors and the intended applications. A graph-based environment model for navigation tasks is presented in Chapter 4. Established techniques for navigation in buildings are based on 2D occupancy grid maps, which are acquired with range sensors. These maps show the free space as well as obstacles such as wall, doors or furniture and can be represented in a more abstract way as topological graphs. The proposed environment model is based on such a topological graph and extends it with haptic information acquired by a visuo-haptic sensor attached to

a mobile platform. Haptic object properties relevant for manipulation, such as friction or lateral motion, are represented on a per-object basis. The resulting graph represents possible choices for navigation, such as turning left/right, as well as potential manipulation tasks that are linked to navigation, such as pushing obstacles aside. Manipulation nodes represent manipulable objects and are associated to specific tasks, properties and semantics. This allows for the representation of, for instance, transparent obstacles, doors or elevators. A path in that graph is an abstract joint navigation/manipulation plan. Alternative routes to reach a goal can be determined, involving for instance either a push operation or a detour. Parts of the map that are otherwise inaccessible can be reached, if manipulation operations are considered. The presented model represents objects based on a single or a low number of "haptic tags" and is thus inadequate for objects with complex geometries or locally changing haptic properties. Moreover, object properties must be known before path planning, i.e. obstacles must have been previously explored.

Chapter 5 presents an approach for joint visuo-haptic geometry acquisition of objects that cannot be fully reconstructed with visual methods. Established methods for 3D visual reconstruction typically assume surfaces with Lambertian reflection. A major cause of reconstruction failures or geometric inconsistencies in static scenes are specular reflections and transparent materials, which exhibit a substantially different image formation model. First, a method is presented that detects transparent regions of an object by searching for geometric inconsistencies using a multi-view setup. In parallel, geometry reconstruction is performed, assuming that objects are only partially transparent. The estimated transparency region is used to find missing or erroneous parts of the reconstructed geometry. Within this region, geometry is initially estimated using surface extrapolation with radial basis functions (RBF), which are fitted to surrounding reliable points (support points). Second, a method is presented for haptic geometry exploration within the missing regions using a single point probe. The certainty of the extrapolated geometry is estimated based on the biharmonic distance to the support points. Exploration steps are planned based on the surface point with the largest distance, respectively. Explored touch points are subsequently added as new support points, refining the estimated surface iteratively. An accurate estimate of the complete object geometry is obtained once there are enough points from haptic exploration. The obtained geometric model can be used for tasks like classification, grasp planning or learning. The presented approach is limited to visual errors or holes caused by transparency or, to some degree, reflection. Moreover, transparent objects must be curved, exhibit dominant refraction or specular reflection effects, and the camera must move around them along a trajectory. The exploration scheme only considers single point contacts, which results in rather long exploration times compared to laminar contacts or multi-point contacts.

In Chapter 6, a model for deformable thin-walled objects, such as bottles and other containers, is presented. At first, a large database of volumetric object models with many variations of each object is created. Based on these models, object deformations are simulated with the Finite Elements Method (FEM). A generic grasp pattern is applied to all points of the surface

in a dense fashion, yielding a so-called stiffness map, which represents first- and second-order stiffness coefficients. The local stiffness varies considerably along the surface of thin-walled objects, depending on their geometry. Two applications for these models are demonstrated: First, a grasp planner is presented, which considers stiffness as well as geometry to determine stable grasp points. Second, an object detector based on haptic exploration is presented. It allows for the distinction of object classes, variants or states, which are indistinguishable using other (e.g. visual) methods. For instance, a visual detector might not be able to discriminate between a thin plastic cup, a paper cup with a reinforcement ring or an indeformable ceramic mug. A tree-based detector is trained based on results from simulation, whereby the object variations in the database ensure generalizability. The exploration scheme selects the point on the object with the largest information gain at each step, ensuring a low number of contact points. One limitation of the presented model is that only stiffness features are considered. The integration of other relevant haptic properties, such as geometry or contact points, may improve the performance of object classification. For grasp planning, friction and weight are very important and must be given as predefined parameters. Furthermore, the presented approach is currently limited to two-finger grippers.

7.2 Outlook

Visuo-haptic sensor The concept of visuo-haptic sensors allows for great freedom in sensor design by adaptation of the deformable element. The major requirements are the observability of deformation as well as a unique and known mapping from deformation to pressure. Even structural parts of the robot could be used as deformable elements, if they are built with corresponding materials. Additionally, such a design allows for lightweight low-cost robots with increased passive safety.

On a realistic implementation of a soft robot, complex models may be required for the deformable elements. For instance, the "skeleton" of an arm may be modeled as a beam, but an additional deformation of a soft surface (the "skin") may need to be considered. In order to obtain the applied pressure, different types of trackers should be combined, such as trackers based on edges, (deformable) templates and depth data. Redundant tracking data improve accuracy and offer increased robustness. Trackers for deformable templates provide 3D deformation maps of textured soft surfaces and thus allow for the detection of lateral forces. With such extensive tracking data, it is a natural extension to also track the joint positions of a robotic arm based on image data. This yields an accurate estimate on the real end-effector pose, and even allows substituting data from dedicated joint sensors.

Finally, the application range of the sensor can be extended by a further integration of visual observations. Similar to the idea outlined above, deformations of an object can be tracked with a stereo or depth camera during grasping. This allows obtaining deformation shapes and learning the deformation characteristics of objects, see also Chapter 6. Additionally,

unexpected reactions of the object during manipulation can be detected, potentially avoiding damage. Within the contact region, the deformed object shape is known both from haptic and visual perception, allowing for geometry refinement and consistency checks.

Environment models The joint geometry models in Chapter 5 rely on the detection of visual modeling errors. A transparency detector is proposed and used as an error detector. There are many other reasons for a partially incomplete object geometry, such as surfaces with a very low reflectivity, active light sources, limited sensor range, incomplete coverage of viewpoints as well as transparent or reflective regions, which are not detected by the proposed detector. The presented detector fails for flat surfaces (such as windows), rough surfaces as well as for materials that do not exhibit dominant refraction effects, such as opal glass. In principle, specific detectors may be developed for each case. However, since this approach results in high computational load, it may be more feasible to rely on additional consistency checks, semantic guesses, or to derive uncertainty scores directly from the reconstructed 3D model. Any detected uncertainty region may be fed into the presented haptic exploration planner. In order to speed up haptic exploration, multi-point contacts – for instance, using a humanoid hand – should be considered, as outlined in Sec. 5.4. There may still be undetected visual reconstruction errors, such that a haptic verification scheme may be feasible.

The deformation models presented in Chapter 6 use so-called deformation shapes during the simulation process. They are obtained for each grasp configuration and currently combined into a single stiffness map. In future work, these shapes could be used for matching of simulation results and real observations: While a real object is being grasped, the corresponding deformation shape can be retrieved from the database and matched against the visual observation of the deformed object. Approaches such as ICP [84] allow for aligning and matching of (partial) point clouds. By checking for deviations between the expected and observed shape, grasp errors can be detected early, potentially avoiding the destruction of objects. Furthermore, if there are multiple model candidates, the correct one can by identified faster by checking deformation shapes.

The presented model can be extended to multi-point grasp configurations, if the relevant forces remain approximately perpendicular to the surface and if grasp configurations are roughly symmetric. The interaction between several contact points may be modeled by a matrix of coupling coefficients. Generally, if contact points are close-by, they excite almost the same deformation pattern, and their forces add up. Deformation shapes of far-away contact points may be superimposed, which allows for fast prediction of the deformation for a complex grasp configuration. The obtained deformations may be integrated in a grasp planning system, such as GraspIt [73].

Finally, the approach has been applied to thin-walled objects because of their pronounced relation between shape and deformation, making visuo-haptic perception especially beneficial. By modeling thin-walled objects as 2D surfaces, more efficient simulations are possible using so-called thin shell models [13]. Furthermore, results may be more accurate, since tetrahedral

meshing of thin walls either yield very large models, or tetrahedra with a bad conditioning[1]. Here, FEM has been used since it also enables to model volumetric structures, such as reinforcements. It can be applied to any object with (hyper)elastic deformation, including simpler objects that are filled homogeneously in their interior.

Versatile object models should consider additional properties, such as surface roughness, dynamic effects and vibrations after contact events, see [20]. These effects are also part of haptic perception, but not considered in this work. Many materials have a characteristic surface structure, which causes vibrations when sliding over their surface. These tactile signals have a relatively large frequency spectrum (e.g. 500 Hz for human perception) and depend also on the contact pressure and the mechanical properties of the sensor. They have been used for object classification tasks, such as the distinction of floor materials [34]. For grasping tasks, surface properties can be used to estimate friction and the required grasping force. In this work, the required force is a parameter of the object models. Also, dynamic object behaviour is not considered by the presented object models. Autonomous robots in human environments move slowly in order to avoid injury of persons, such that dynamic properties do not need to be modeled in detail for many applications. Yet, a simplified model of dynamics as part of the "haptic tag" would improve the understanding of object reactions, such as falling over.

The environment and object models proposed in Chapters 4–6 represent different haptic properties and are targeted to the needs of specific applications. It is an obvious extension to develop a single common object representation that covers the presented as well as additional relevant object properties in a generic way. Such a versatile visuo-haptic representation should address the following aspects:

Multiple domains/levels Versatile object representations are inherently multimodal, since they represent visual and various haptic properties, such as stiffness, surface friction and mass density. Many of these properties are represented by a map on the same domain, facilitating a common representation. For instance, scalar maps over the object surface may be used to represent or approximate visual texture, stiffness as well as friction, and can be acquired by surface exploration. Properties on a volumetric domain, such as the mass density or the elastic modulus, may be derived from artificial object models, but cannot be directly explored on an object. Only indirect effects are observable, such as inertial behavior or deformation shapes. Such properties describe an object on a global level and cannot be assigned to specific locations.

Partial information In contrast to visual models, there is not one particular process to acquire all relevant haptic data of an object. Rather, distinct exploration processes may be needed to obtain a complete, generic object representation. Additionally, the dense acquisition of haptic properties is very time-consuming. Versatile object models must thus deal with partial information. Missing data should be explored as needed, or

[1] The conditioning of a tetrahedron is often represented by the ratio between its shortest edge and the radius of the circumscribed sphere.

estimated if possible. A "complete" object representation can be provided by a database, or be obtained on a robot with complex manipulators by time-consuming exploration.

Sensors Different sensor sources must be registered and aligned before the provided data are fused in a common model. Retrospective refinements are necessary, if alignment parameters are corrected during exploration. Moreover, the haptic appearance of an object depends on the sensor or actuator used during the manipulation process. This is especially important for tactile data, such as surface vibrations or friction as well as for grasping contacts (e.g. single point versus laminar contact). A transformation of the "object views" from one type of sensor to another is possible, if model data are detailed enough and the sensor models are known. Versatile models should allow generating these views on request.

The models in Chapters 5 and 6 both represent complementary surface properties of objects and may thus be combined in a straight-forward manner. Geometry and stiffness data of an object can be acquired simultaneously by exploration. The deformation model, however, can only be obtained after the complete object geometry is known, i.e. after the geometry model has been built. Additionally, it requires a volumetric object representation or volumetric parameters from a priori data or from a semantic database. Integration of the navigation/manipulation models (Chapter 4) is less straight-forward, since they represent various object properties on a different domain/level. One feasible approach is to introduce intermediate object representations, such as part-based models with individual haptic properties for each part. This would allow for the prediction of object reactions to manipulation operations based on physical models, as presented in Sec. 4.3.2, rather than on simple extrapolation of observed behavior. The motion of more complex objects, such as chairs or tables, could be predicted reliably with such representations. More powerful models, however, go along with a more complex and time-consuming exploration scheme.

List of Figures

1.1 Commercial domestic robots for vacuum cleaning and lawn mowing 2
1.2 Humanoid robots with wheeled platforms or legs 3
1.3 Manipulation in unstructured environments requires visuo-haptic perception . 4

2.1 Laminar tactile sensors for robotic hands and bodies 12
2.2 Tactile sensors with optical readout 13
2.3 Scene reconstruction with KinectFusion and OctoMap 15
2.4 Tactile exploration and reconstruction, joint visuo-haptic exploration 18
2.5 Deformation map from reference configuration to deformed shape 21
2.6 Simplified example for a 1D FEM assembled from rod elements 22

3.1 Foam-based visuo-haptic sensor for a platform and a gripper 35
3.2 Experimentally determined strain-stress relations for PUR/rubber foams . . . 38
3.3 Simplified arrangement of camera and foam 42
3.4 Sensitivity of stress measurements to the image position 44
3.5 Proposed beam-based sensor 47
3.6 Visual point features tracked on the object surface and the floor 49
3.7 Comparison of force measured by the proposed sensor against a reference . . 51
3.8 Accuracy of the beam-based sensor 52
3.9 Foam-based sensors mounted on a mobile platform and a gripper 54
3.10 View from the platform camera while touching different objects 55
3.11 Office scene mapped by naïve haptic exploration 56
3.12 Transparent bottle explored by a gripper equipped with the foam-based sensor 57
3.13 Stiffness of the transparent bottle measured with the proposed sensor 58
3.14 Opening a crown cap with the beam-based visuo-haptic sensor 59

4.1 Skeleton, distance map and navigation graph extracted from a map 64
4.2 Movable obstacles are added to the map 68
4.3 Search for valid push goals 69
4.4 Planning of the sideward push manipulation 71
4.5 Proposed exploration system tested in an office scene 75
4.6 Map and navigation graphs for the office scene 76

5.1 A partly transparent bottle observed by the Kinect 81
5.2 Proposed setup: The depth sensor moves around the scene 82
5.3 Local background model and error score 84
5.4 Geometry reconstruction of a partly transparent bottle 86
5.5 Office scene used for evaluation of transparency detection 90
5.6 Results of transparency detection in a cluttered scene 91
5.7 Simulation results of haptic exploration for artificial objects 93
5.8 Error of the extrapolated geometry obtained by simulated haptic exploration 94

6.1 Overview of classification and grasp planning with deformable object models 98
6.2 Simulated deformation shapes, stiffness map 101
6.3 Parametric model generator, locations of stiffness features 103
6.4 Grasping scores for various objects 108
6.5 Stiffness maps of exemplary models in the database 117
6.6 Correlation between feature vectors of different classes 118
6.7 Stiffness features for four classes, trained decision tree 119
6.8 Exploration system with a KUKA LWR arm and a two-finger gripper 119
6.9 Real objects used in experiments for haptic exploration and classification . . 120
6.10 Exploration results of the “cien” plastic bottle 121
6.11 Quadratic feature values $\mathcal{F}^{II}$ compared for two different object states 122
6.12 Grasping experiments with real objects 124

List of Tables

2.1 Value of the Young's modulus E for various materials 20

3.1 Sequence of tasks for opening a crown cap 59

4.1 Haptic tags of various objects obtained during the experiments 75

6.1 Decision trees trained with different subsets of classes 120
6.2 Detection results of haptic exploration on real objects 122

Publications by the author

[1] N. Alt, S. Hinterstoisser, and N. Navab. Rapid selection of reliable templates for visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Fransisco, USA, June 2010. DOI: 10.1109/CVPR.2010.5539812.

[2] N. Alt, W. Maier, Q. Rao, and E. Steinbach. Semantic interpretation of novelty in images using histograms of oriented gradients. In C.-Y. Su, S. Rakheja, and H. Liu, editors, *Intelligent Robotics and Applications (ICIRA)*, Lecture Notes in Computer Science, Montreal, Canada, Oct. 2012. DOI: 10.1007/978-3-642-33503-7_42.

[3] N. Alt, Q. Rao, and E. Steinbach. Haptic exploration for navigation tasks using a visuo-haptic sensor. In *Interactive Perception (ICRA Workshop)*, Karlsruhe, Germany, May 2013.

[4] N. Alt, P. Rives, and E. Steinbach. Reconstruction of transparent objects in unstructured scenes with a depth camera. In *IEEE International Conference on Image Processing (ICIP)*, Melbourne, Australia, Sept. 2013. DOI: 10.1109/ICIP.2013.6738851.

[5] N. Alt and E. Steinbach. Visuo-haptic sensor for force measurement and contact shape estimation. In *Haptic Audio-Visual Environments and Games (HAVE)*, Istanbul, Turkey, Oct. 2013. DOI: 10.1109/HAVE.2013.6679605.

[6] N. Alt and E. Steinbach. Navigation and manipulation planning using a visuo-haptic sensor on a mobile platform. *IEEE Transactions on Instrumentation and Measurement*, 63(11), Nov. 2014. Early access: May 2014. DOI: 10.1109/TIM.2014.2315734.

[7] N. Alt and E. Steinbach. A visuo-haptic sensor for the exploration of deformable objects. In *Autonomous Grasping and Manipulation (ICRA Workshop)*, Hong Kong, China, May 2014.

[8] N. Alt, J. Xu, and E. Steinbach. Grasp planning for thin-walled deformable objects. In *Robotic Hands, Grasping, and Manipulation (ICRA Workshop)*, Seattle, WA, USA, May 2015.

[9] N. Alt, J. Xu, and E. Steinbach. A dataset of thin-walled deformable objects for manipulation planning. In *Grasping and Manipulation Datasets (ICRA Workshop)*, Stockholm, Sweden, May 2016.

Bibliography

[10] C. Audras, A. I. Comport, M. Meilland, and P. Rives. Real-time dense appearance-based SLAM for RGB-d sensors. In *Australian Conference on Robotics and Automation*, Melbourne, Australia, Dec. 2011.

[11] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, Feb. 2004.

[12] S. Benhimane, A. Ladikos, V. Lepetit, and N. Navab. Linear and quadratic subsets for template-based tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, MN, USA, June 2007.

[13] M. Bernadou. *Finite Element Methods for Thin Shell Problems*. Wiley, Chichester, 1996.

[14] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schaeffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger. The KUKA-DLR lightweight robot arm - a new reference platform for robotics research and manufacturing. In *Robotics (ISR), International Symposium on and German Conference on Robotics (ROBOTIK)*, Munich, Germany, June 2010.

[15] J. Bohg, M. Johnson-Roberson, M. Bjoïkman, and D. Kragic. Strategies for multi-modal scene exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, Oct. 2010.

[16] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *Annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, Los Angeles, CA, USA, Aug. 2001.

[17] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3), Apr. 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[18] H. Chen and B. Bhanu. 3D free-form object recognition in range images using local surface patches. *Pattern Recognition Letters*, 28(10):1252–1262, July 2007.

[19] Y.-W. Chen and C.-J. Lin. Combining SVMs with various feature selection strategies. In I. Guyon, M. Nikravesh, S. Gunn, and L. Zadeh, editors, *Feature Extraction*, volume 207 of *Studies in Fuzziness and Soft Computing*, pages 315–324. Springer, Berlin / Heidelberg, 2006.

[20] S. Chitta, J. Sturm, M. Piccoli, and W. Burgard. Tactile sensing for mobile manipulation. *IEEE Transactions on Robotics*, 27(3):558–568, May 2011.

[21] T. F. Cootes, C. J. Taylor, D. H. Cooper, J. Graham, and others. Active shape models-their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, Jan. 1995.

[22] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, CA, USA, June 2005.

[23] P. Danielsson and O. Seger. Generalized and separable sobel operators. In Freeman, H., editor, *Machine Vision for Three-Dimensional Scenes*, pages 347–379. Academic Press, San Diego, 1990.

[24] J. Dankert and H. Dankert. *Technische Mechanik*. Springer Vieweg, Wiesbaden, 2013.

[25] J. de Oliveira and R. Romero. Image skeletonization method applied to generation of topological maps. In *Latin American Robotics Symposium (LARS)*, Valparaiso, Chile, Oct. 2009.

[26] H. Deng and G. Runger. Feature selection via regularized trees. In *International Joint Conference on Neural Networks (IJCNN)*, Brisbane, Australia, June 2012.

[27] O. Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(3):485–508, Sept. 1988.

[28] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[29] M. Freundl. A visuo-haptic sensor for a screw driver tool on a robot arm. Student research project (Forschungspraxis), Institute for Media Technology, Technische Universität München, Munich, Germany, Mar. 2015.

[30] M. Fritz, M. Black, G. Bradski, S. Karayev, and T. Darrell. An additive latent feature model for transparent object recognition. In *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, Canada, Dec. 2009.

[31] S. Gabl and F. Walch. Visuo-haptic data fusion for accurate geometry reconstruction. Student research project (IDP), Institute for Media Technology, Technische Universität München, Munich, Germany, Sept. 2015.

[32] S. Garrido, L. Moreno, and D. Blanco. Exploration and mapping using the VFM motion planner. *IEEE Transactions on Instrumentation and Measurement*, 58(8):2880–2892, Aug. 2009.

[33] L. J. Gibson and M. F. Ashby. *Cellular Solids: Structure and Properties*. Cambridge solid state science series. Cambridge University Press, Cambridge, 2nd edition, 1999.

[34] P. Giguere and G. Dudek. Surface identification using simple contact dynamics for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 2009.

[35] D. Goger, N. Gorges, and H. Worn. Tactile sensing for an anthropomorphic robotic hand: Hardware and signal processing. In *IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 2009.

[36] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, Feb. 2007.

[37] D. Gross, W. Hauger, J. Schröder, and W. A. Wall. *Technische Mechanik 2*. Springer-Lehrbuch. Springer Vieweg, Berlin / Heidelberg, 2014.

[38] E. Guizzo. How rethink robotics built its new baxter robot worker. *IEEE Spectrum*, Sept. 2012.

[39] R. Haralick, H. Joo, C. Lee, X. Zhuang, V. Vaidya, and M. Kim. Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1426–1446, Nov. 1989.

[40] C. Harris and M. Stephens. A combined corner and edge detector. In *Fourth Alvey Vision Conference*, pages 147–151, Manchester, UK, Aug. 1988.

[41] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968.

[42] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, 2003.

[43] D. D. Hoffman. *Visual Intelligence: How We Create What We See*. Norton, New York, Feb. 2000.

[44] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: an efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, Feb. 2013.

[45] K. Hsiao, S. Chitta, M. Ciocarlie, and E. Jones. Contact-reactive grasping of objects with partial shape information. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, Oct. 2010.

[46] R. Huitl, G. Schroth, S. Hilsenbeck, F. Schweiger, and E. Steinbach. TUMindoor: An extensive image and point cloud dataset for visual indoor localization and mapping. In *IEEE International Conference on Image Processing (ICIP)*, Orlando, FL, USA, Sept. 2012.

[47] I. Ihrke, K. N. Kutulakos, H. P. A. Lensch, M. Magnor, and W. Heidrich. Transparent and specular object reconstruction. *Computer Graphics Forum*, 29(8):2400–2426, Nov. 2010.

[48] K. Kamiyama, K. Vlack, T. Mizota, H. Kajimoto, N. Kawakami, and S. Tachi. Vision-based sensor for real-time measuring of surface traction fields. *IEEE Computer Graphics and Applications*, 25(1):68–75, Jan. 2005.

[49] M. Kaneko, N. Nanayama, and T. Tsuji. Vision-based active sensor using a flexible beam. *IEEE/ASME Transactions on Mechatronics*, 6(1):7–16, Mar. 2001.

[50] Karto Robotics. KARTO open libraries 2.0, 2010. http://kartorobotics.com/products/. Retrieved Mar. 2015.

[51] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, Jan. 1988.

[52] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *IEEE and ACM International Workshop on Augmented Reality*, San Francisco, CA, USA, Oct. 1999.

[53] K. Khoshelham and S. O. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, Feb. 2012.

[54] A. Kimoto and Y. Matsue. A new multifunctional tactile sensor for detection of material hardness. *IEEE Transactions on Instrumentation and Measurement*, 60(4):1334–1339, Mar. 2011.

[55] U. Klank, D. Carton, and M. Beetz. Transparent object detection and reconstruction on a mobile platform. In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[56] U. Klank, M. Zia, and M. Beetz. 3D model selection from an internet database for robotic vision. In *IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 2009.

[57] E. Knoop and J. Rossiter. Dual-mode compliant optical tactile sensor. In *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[58] K. Konolige. A gradient method for realtime robot control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Takamatsu, Japan, Oct. 2000.

[59] K. Konolige, G. Grisetti, R. Kummerle, W. Burgard, B. Limketkai, and R. Vincent. Efficient sparse pose adjustment for 2D mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, Oct. 2010.

[60] Y. Lipman, R. M. Rustamov, and T. A. Funkhouser. Biharmonic distance. *ACM Transactions on Graphics (TOG)*, 29(3), July 2010.

[61] W.-Y. Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, Jan. 2011.

[62] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *ACM Computer Graphics (SIGGRAPH)*, Anaheim, CA, USA, July 1987.

[63] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004.

[64] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence (IJCAI)*, Vancouver, Canada, Aug. 1981.

[65] I. Lysenkov, V. Eruhimov, and G. Bradski. Recognition and pose estimation of rigid transparent objects with a kinect sensor. In *Robotics: Science and Systems*, Sydney, Australia, July 2012.

[66] Y. Ma. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer, New York, 2004.

[67] E. Malis. Improving vision-based control using efficient second-order minimization techniques. In *IEEE International Conference on Robotics and Automation (ICRA)*, New Orleans, LA, USA, Apr. 2004.

[68] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige. The office marathon: Robust navigation in an indoor office environment. In *IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, AK, USA, May 2010.

[69] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, May 2004.

[70] F. Mazzini, D. Kettler, J. Guerrero, and S. Dubowsky. Tactile robotic mapping of unknown surfaces, with application to oil wells. *IEEE Transactions on Instrumentation and Measurement*, 60(2):420–429, Feb. 2011.

[71] B. Merhy, P. Payeur, and E. Petriu. Application of segmented 2-d probabilistic occupancy maps for robot sensing and navigation. *IEEE Transactions on Instrumentation and Measurement*, 57(12):2827–2837, Dec. 2008.

[72] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, Oct. 2004.

[73] A. T. Miller and P. K. Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, Dec. 2004.

[74] S. Miller, J. v. d. Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel. A geometric approach to robotic laundry folding. *The International Journal of Robotics Research*, 31(2):249–267, Feb. 2012.

[75] P. Mittendorfer, E. Dean, and G. Cheng. 3D spatial self-organization of a modular artificial skin. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, USA, Sept. 2014.

[76] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136, Basel, Switzerland, Oct. 2011.

[77] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, New York, NY, USA, June 2006.

[78] OSR Foundation. TurtleBot 2, 2013. http://www.turtlebot.com/. Retrieved Mar. 2015.

[79] J. Pan, S. Chitta, and D. Manocha. FCL: A general purpose library for collision and proximity queries. In *IEEE International Conference on Robotics and Automation (ICRA)*, Saint Paul, MN, USA, May 2012.

[80] D. Pollard. *A User's Guide to Measure Theoretic Probability*. Cambridge University Press, Cambridge, 2001.

[81] V. Pratt. Direct least-squares fitting of algebraic surfaces. In *ACM Computer Graphics (SIGGRAPH)*, Anaheim, CA, USA, July 1987.

[82] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source robot operating system. In *Open Source Software (ICRA Workshop)*, Kobe, Japan, May 2009.

[83] T. Ruhr, J. Sturm, D. Pangercic, M. Beetz, and D. Cremers. A generalized framework for opening doors and drawers in kitchen environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, Saint Paul, MN, USA, May 2012.

[84] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Third International Conference on 3-D Digital Imaging and Modeling*, Quebec City, Canada, May 2001.

[85] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura. The intelligent ASIMO: system overview and integration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, Sept. 2002.

[86] J. M. Santos, D. Portugal, and R. P. Rocha. An evaluation of 2D SLAM techniques available in robot operating system. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Linköping, Sweden, Oct. 2013.

[87] D. G. Schneider, D. A. Seewald, and S. Heinrichs-Bartscher. "All-in-one" – video-based driver assistance systems. *ATZ worldwide eMagazine*, 113(3):26–29, Feb. 2011.

[88] F. Schroff, A. Criminisi, and A. Zisserman. Harvesting image databases from the web. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):754–766, Apr. 2011.

[89] J. Shi and C. Tomasi. Good features to track. Technical Report TR-93-1399, Cornell University, Nov. 1993.

[90] F. Y. Shih. *Image Processing and Mathematical Morphology: Fundamentals and Applications*. CRC Press, Boca Raton, Nov. 2009.

[91] H. Si. TetGen, a delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software (TOMS)*, 41(2), Feb. 2015.

[92] E. Sifakis and J. Barbic. FEM simulation of 3D deformable solids: a practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH Courses*, Los Angeles, CA, USA, Aug. 2012.

[93] F. S. Sin, D. Schroeder, and J. Barbič. Vega: Non-linear FEM deformable object simulator. *Computer Graphics Forum*, 32(1):36–48, Feb. 2013.

[94] R. Szabo. Topological navigation of simulated robots using occupancy grid. *International Journal of Advanced Robotic Systems*, 1(3):201–206, Sept. 2004.

[95] J. W. H. Tangelder and R. C. Veltkamp. A survey of content based 3D shape retrieval methods. *Multimedia Tools and Applications*, 39(3):441–471, Sept. 2008.

[96] S. Tsuji, A. Kimoto, and E. Takahashi. A multifunction tactile and proximity sensing method by optical and electrical simultaneous measurement. *IEEE Transactions on Instrumentation and Measurement*, 61(12):3312–3317, Dec. 2012.

[97] Universal Robots. Collaborative robot solutions, flexible robot arms, 2009. http://www.universal-robots.com/GB/Products.aspx. Retrieved Mar. 2015.

[98] Various Authors. Breakthrough factories. *MIT Technology Review*, Nov. 2014.

[99] Wikipedia. Young's modulus, 2015. https://en.wikipedia.org/wiki/Young's_modulus. Retrieved Mar. 2015.

[100] A. Witkin, D. Baraff, and A. Kass. Physically based modeling. In *ACM SIGGRAPH Courses*, Los Angeles, CA, USA, Aug. 2001.

[101] H. Yousef, M. Boukallel, and K. Althoefer. Tactile sensing for dexterous in-hand manipulation in robotics—a review. *Sensors and Actuators A: Physical*, 167(2):171–187, June 2011.

[102] J.-i. Yuji and K. Shida. A new multifunctional tactile sensing technique by selective data processing. *IEEE Transactions on Instrumentation and Measurement*, 49(5):1091–1094, Oct. 2000.

[103] Y. Zhuang, N. Jiang, H. Hu, and F. Yan. 3-d-laser-based scene measurement and place recognition for mobile robots in dynamic indoor environments. *IEEE Transactions on Instrumentation and Measurement*, 62(2):438–450, Feb. 2013.

[104] Z. Zivkovic and F. van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773–780, May 2006.