Learning Deformable Object Models for Mobile Robot Navigation using Depth Cameras and a Manipulation Robot

Barbara Frank Ruediger Schmedding Cyrill Stachniss Matthias Teschner Wolfram Burgard
University of Freiburg, Department of Computer Science, 79110 Freiburg, Germany

***Abstract*— In this paper, we present our recently developed robotic system that can navigate in environments with deformable objects. To achieve this, we propose techniques to learn models of deformable objects by physical interaction between the robot and the objects. We determine the model parameters by establishing a relation between the applied forces and the corresponding surface deformations as observed with a depth camera. After modeling the objects in a scene, the robot can perform its navigation tasks more efficiently by considering the cost of deformations during path planning. As we demonstrate in real-world experiments, our system is able to estimate appropriate physical parameters that can be used to predict future deformations and exploits this information during path planning.**

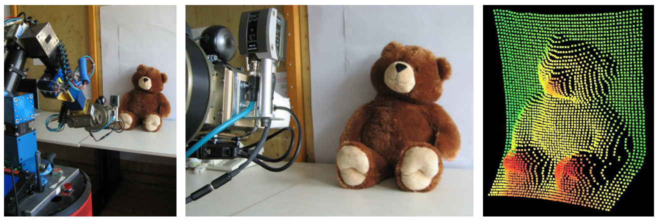
Fig. 1. Reconstructing a model of a deformable object: manipulation robot (left) equipped with a depth camera and force-feedback sensor (middle) and the 3D perception of the robot (right).

I. Introduction

Perceiving the surroundings and modeling the environment is an important competence of intelligent mobile robots since such models are relevant for efficiently solving other high-level tasks. For example, generating a collision-free path through the environment in an efficient way requires path planning which builds on top of a model of the environment. In this paper, we present a complete robotic system that is able to perceive the environment and model the deformable objects in the scene. The system estimates the deformation properties of objects, and finally is able to plan a trajectory through the environment, taking potential object deformations into account.

Dealing with deformable objects in the environment is relevant for building robust robotic systems, especially when operating in domestic environments. Even though the majority of path planning approaches focuses on planning in static environments and with rigid obstacles [16, 17], not all obstacles are rigid in reality. Considering that an object such as a curtain is deformable can enable a robot to accomplish navigation tasks that otherwise cannot be carried out. In domestic environments — a key target domain for service robots — a robot must deal with many deformable objects such as plants, curtains, or cloth. Ignoring the deformation properties will clearly limit the tasks a service robot can carry out.

To consider deformable objects in the path planning process, such objects need to be handled in the simulation system that underlies the planner. The realistic simulation of object deformations is still an active area of research. There exists a variety of relevant applications in computer graphics, robotics [23], virtual reality, games, movies, and medical simulation [18, 6, 21]. Most planning techniques as well as most applications considering deformations assume that the elasticity parameters of objects in the scene are given. These parameters are required for the accurate simulation of the deformation and the computation of the deformation costs. In practice, the underlying parameters for the appropriate simulation of deformations are typically adjusted manually. Thereby, the parameters are usually modified until the result looks visually plausible. This might be applicable for computer games or movies, but does not necessarily lead to a physically realistic computation of the involved forces. These forces, however, need to be known accurately for navigation in the presence of deformable objects. For example, whenever robots interact with real-world objects, only limited forces should be applied to them. This is of utmost importance in medical applications or in domestic settings, i.e. whenever robots have to manipulate plants or clothes. Especially in these domains robots need exact knowledge about the parameters of the deformation process.

In this paper, we consider the problem of estimating the elasticity parameters of objects and subsequently use them in the path planning process. Generating such realistic models of deformable objects not only involves observing and reconstructing the three-dimensional surface of an object. Physical interaction with the object under consideration is required to learn about its behavior when exposed to external forces. Therefore, we equipped our robot with a force sensor at the end of the manipulator and with a depth camera. This setup allows the robot to interact with objects and to measure the forces exerted on them while at the same time observing the deformations (see Figure 1). Based on the observed deformations and forces, our approach seeks to determine the elasticity parameters of the object. This is done by simulating the object deformation under the applied forces. An error minimization approach is applied to iteratively adapt the deformation parameters so that the difference between the real object under deformation and the simulation is minimized. As we will demonstrate in the experimental section of this paper, our approach is able to find elasticity parameters that enable our robot to accurately predict the deformation of real-world objects.

II. Related Work

Most approaches to mobile robot path planning assume that the environment is static and that all objects are rigid [16, 14]. In the last years, however, path planning techniques for deformable robots in static environments have been presented [11, 13]. All these approaches operate in simulated environments. Our planning system, first introduced in [8], applies FEMs to compute object deformations. In contrast to [8], we realize in this paper a planning system on a real physical mobile robot and not only in simulation. This requires a series of adaptations and new techniques for successfully planning paths in environments with deformable objects.

Deformable modeling and parameter estimation are active areas of research. To represent non-rigid objects and to simulate deformations, mass-spring systems have been frequently used as they are easy to implement and can be simulated efficiently [7]. While such models are able to handle large deformations, their major drawback is the tedious modeling as there is no intuitive relation between spring constants and physical material properties in general [20]. Finite element methods (FEMs) reflect physical properties of the objects in a more natural way [2]. The disadvantage of FEMs lies in the computational resources required to calculate deformations. A computationally more efficient approach, which we also use in our current system, is the co-rotational finite element approach [12, 19] that avoids nonlinear computations.

There exist some approaches to determine the physical parameters of models. Bianchi *et al.* [4] learn the stiffness constants of mass-spring models by using a genetic algorithm and comparing it to a FEM reference model. Another approach that estimates the stiffness properties of mass-spring models was proposed by Burion *et al.* [5]. They use a particle filter to obtain a posterior distribution over the stiffness parameters and evaluate the particles by comparing simulated and observed deformations. In contrast to our work, they do not compare the deformed surfaces but the measured forces in the single nodes of the object. Furthermore, they did only work on simulated data.

One approach that deals with real objects was presented by Lang *et al.* [15]. They describe a deformable model as a discrete boundary value problem and estimate Greens' functions from measured forces and displacements. They formulate the estimation of the deformation matrix as a linear estimation problem. In contrast to Lang *et al.*, we use a different approach for modeling deformability, namely the finite element method. In our setup, the mobile manipulation robot furthermore carries its force sensor and camera on-board and thus is the basis for fully autonomous exploration, whereas Lang *et al.* use a fixed manipulator in combination with an accurate measuring device and place the objects on a turntable.

This paper summarizes and combines the work carried out in [8, 10, 9]. In contrast to most of the previous approaches, our method has been realized on a real mobile manipulation robot and deals with real data. Furthermore, the resulting models can directly be used for simulations, which have been shown to be relevant to robot navigation in environments containing deformable objects.

III. Overview

Our approach to navigation among deformable objects is based on a physical simulation of object deformations (see Section IV) and consists of three main steps:

- data acquisition with a manipulator and a depth camera (Section V),
- parameter estimation via simulation and error minimization (Section VI), and
- path planning using the learned models (Section VII).

In the data acquisition process, our robot first constructs a 3D model of the object under consideration in an undeformed state. In a next step, the robot interacts with the object and measures the forces it exerts on the object. Additionally, it observes the surface of the deformed object. This allows us to estimate a relationship between the displacement of the surface points, the applied forces and the physical elasticity parameters.

After estimating the elasticity parameters, the Young modulus and the Poisson ratio in our case, we can use these models in an $A^{\star}$-based path planning approach. The planner seeks to find the trajectory that optimizes the trade-off between travel– and deformation cost. To answer path queries efficiently, the robot caches the cost of potential trajectories that lead to objects deformations and generalizes them using regression. This in turn allows us to avoid time-consuming deformation simulations during robot navigation.

IV. Deformation Simulation

The key idea of our elasticity parameter estimation approach is to adapt the parameters of a realistic simulation system until the simulated deformations approximate the ones measured on the real object. This section briefly describes our simulation environment that is based on finite element methods and is used to deform virtual objects.

A. Modeling Objects using Tetrahedral Meshes

To simulate the deformations of the object, our system requires a volumetric model of the object. We obtain that model by first registering multiple depth images. The resulting point cloud is then transformed into a triangular surface mesh and can be used to determine the volumetric tetrahedral mesh. The actual internal forces are then computed on the volumetric mesh based on force-displacement relations. To establish this tetrahedral mesh, we employ the meshing approach by Spillmann *et al.* [25].

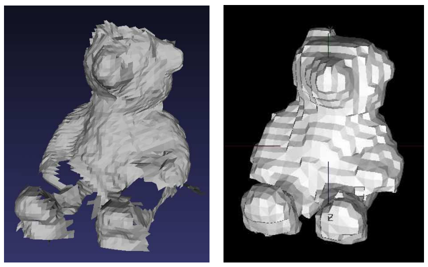

Fig. 2. Reconstruction of a geometric model: registered surface mesh consisting of four depth images (left) and the resulting volumetric model used in simulation (right).

This approach is particularly suited for our application, as it can handle unorientable, non-manifold, and even incomplete data. In this approach, one first computes a signed distance field where voxels having a negative sign represent the volume of the object. In a second step, one divides the spatial domain by a uniform axis-aligned grid. We discard all cells of this grid that do not contain any voxel with negative sign. The remaining cells are an approximation of the object's volume, whose quality is given by the grid resolution. We divide these cells into five tetrahedrons each. In a post-processing step, we smooth the tetrahedrons to align with the given surface mesh. This reconstruction step is illustrated in Figure 2.

In our simulator, we perform all deformation computations based on the tetrahedral mesh. The coupling of the surface mesh to the tetrahedral mesh guarantees that the surface mesh is also deformed. This allows us to compare it to the scanned surface mesh of the real-world object.

B. Finite Elements Deformation Model

To simulate the dynamic behavior of an object and the reaction to external forces, we have to compute the internal forces that act inside the object depending on the current deformation. These forces are computed using a force-displacement-relation that is derived from the underlying deformation model.

The basic idea of the finite element method is to divide the object into smaller elements and to establish the force-displacement relations on these small elements. In our case, these elements are the tetrahedrons mentioned above. This allows us to assume constant stress over an element, which results in a linear force-displacement-relation. Putting all these relations together, one can establish the so-called stiffness matrix $\mathbf{K} = \mathbf{K}(E, \nu)$ that depends on the Young modulus E and the Poisson ratio ν. For n being the number of vertices of an object, the dimension of $\mathbf{K}$ is $3n \times 3n$. The global force-displacement relation then becomes

$$\mathbf{f} = \mathbf{K}\mathbf{q}, \tag{1}$$

where $\mathbf{f} \in \mathbb{R}^{3n}$ is the internal force induced by the displacement $\mathbf{q} \in \mathbb{R}^{3n}$ of the vertices of the tetrahedral mesh.

In sum, the stiffness matrix for given parameters allows us to compute the displacement of the object's vertices given an external force.

When restricting ourselves to forces that do neither cause translations nor rotations, we are able to obtain the inverse relationship $\mathbf{q} = \mathbf{K}^{-1}\mathbf{f}$ even if $\mathbf{K}$ is not invertible in general. The solution can be computed, since only the Eigenvalues modeling the displacement are zero and all others are non-zero.

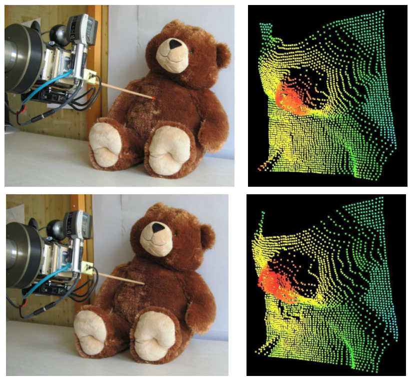

Fig. 3. Data acquisition: Deforming a plush teddy bear and the corresponding observation of the robot.

V. Data Acquisition

Our system for acquiring real data consists of a mobile platform equipped with a 7-DoF manipulator including a force-torque sensor. To perceive the environment, we use a PMD[vision]-O3 time-of-flight camera, which is attached to the gripper of the manipulator. This setup allows us to obtain surface meshes of objects from different view points and therefore to reconstruct complete models (as shown in Figure 2). Furthermore, the robot is able to deform objects and measure the corresponding forces in a flexible way (illustrated in Figure 3).

To actually interact with the object, the robot uses its manipulator to apply a force to the object. In our current implementation, the robot approaches the object, and step by step, increases the force, until a maximum force of 20 N is applied or the end-effector moved for more than 10 cm. In this way, we obtain a set of force measurements $z_t^f \in \mathbb{R}^3$ in combination with corresponding surface meshes $z_t^s \in \mathbb{R}^{3n}$ for every point in time t. We assume here, that z_t^s contains only the object, i. e. the parts of the robotic body have already been removed. In addition to that, we estimate the contact point p_t of the manipulator on the surface. This information is required for the parameter estimation process described below.

VI. Parameter Estimation

The deformation model introduced in Section IV requires to specify two parameters: The Young modulus and the Poisson ratio. The Young modulus E describes the stiffness of an isotropic elastic material. It is defined as the ratio of the uni-axial stress over the uni-axial strain in the range of stress in which Hooke's Law holds. In contrast to that, the Poisson ratio ν is the ratio of the contraction or transverse strain (perpendicular to the applied load) to the extension or axial strain (in the direction of the applied load).

In this section, we explain our approach to estimate both parameters based on observations of the robot. The key idea of our approach is to apply a gradient-descent based error minimization approach to minimize the difference between the real deformation and the simulated one given the elasticity parameters.

A. Error Function

To apply gradient descent, we need to define an appropriate error function, which in our case should reflect the difference between the measured and the simulated surface, since the surface can be observed by the robot. To compute this difference, we first align the surfaces with a registration procedure and then measure the remaining difference.

The task of registration algorithms is to align multiple overlapping scans of the same object, i.e., to compute a translation and a rotation that align the surfaces correctly. In our approach, we apply the ICP-algorithm by Besl and McKay [3], with some extensions similar to the ideas given by Pulli [22] and Rusinkiewicz [24]. Since the correspondences are not known in general, the ICP algorithm determines some correspondences, computes a transformation that aligns the scans for these correspondences, and then determines new correspondences to compute a new relative position. Typically, this procedure converges to a minimum and yields an accurate alignment if a proper initial configuration is chosen. In our system, we can easily derive a good initial alignment from the position of the manipulator to which the camera is attached.

After applying ICP, we can define the error function between a model M and the measured surface z^s as

$$Err(E,\nu) \quad = \quad \text{dist}(\text{simulate}(E,\nu,M,z_t^f,p_t),z_t^s), \quad (2)$$

with

$$\text{dist}(M_{def},z^s) \quad = \quad \sum_{i\in z^s} \min_{j\in M_{def}} ||i-j||^2, \quad (3)$$

where i and j refer to the points from the observed and the simulated surface, respectively.

B. Gradient Descent for Parameter Estimation

After defining the error function above, we can apply gradient descent to seek for a Young modulus E and Poisson ratio ν that minimize the error. Algorithm 1 summarizes the main routine. The variable M refers to the undeformed object model which is generated from observations of the undeformed object. This model is the basis for all simulations. Line 3 of the algorithm requires to compute the partial derivative of the error function. Since the error function involves the simulation approach explained above, the derivatives cannot be computed in closed form. Thus, we approximate this term numerically.

Algorithm 1 Iterative parameter estimation

```
Require: Object model $M$, observations $z_t^f$, $z_t^s$, contact point $p_t$,
 1: Initialize $(E_0,\nu_0)$, i=1
 2: loop
 3:    $(E_i,\nu_i)^T = (E_{i-1},\nu_{i-1})^T - \lambda\nabla Err(E_{i-1},\nu_{i-1})$
 4:    $M_{def}$ = simulate($E_i,\nu_i,M,z_t^f,p_t$)
 5:    $err$ = dist($M_{def},z_t^s$)
 6:    if $err < \epsilon$ then
 7:       return $(E_i,\nu_i)$
 8:    end if
 9:    i++
10: end loop
```

VII. Robot Trajectory Planning Considering Object Deformations

In our planning system, we apply a standard randomized roadmap planning approach [14]. The key challenge here is to efficiently determine the deformation cost. Cost queries need to be frequently answered during planning and thus carrying out thousands of simulations is too costly for online applications.

A. Deformation Cost Functions

To allow for the efficient generation of trajectories for a mobile robot in environments with deformable objects, we build upon our recent work [8]. The key idea is to learn cost functions for the individual deformable objects, which are parametrized by different trajectories leading to deformations in a preprocessing step.

The goal of the approximate cost function is to quickly provide an estimate of the deformation costs for all objects along an edge in the roadmap. The actual value of the deformation cost function mainly depends on the trajectory of the robot relative to the object and the object itself. For each object, we therefore precompute the deformation cost for a number of linear path segments through the object. A path segment is specified by a starting location (x,y) and the traveling direction θ as well as the traveled distance l on the path segment. The traveled distance is constrained to the maximum distance that the robot can move while still deforming the object.

In a preprocessing step, we carry out the simulations for a uniform resolution of starting points and directions and store the deformation costs for a fine length resolution in a histogram. This leads to the approximate deformation cost function $\hat{C}_{def}(x,y,\theta,l) \rightarrow \mathbb{R}$ which returns the deformation cost for edges of the roadmap.

B. Processing path queries

We compute the deformation cost $\hat{C}_{def}(x,y,\theta,l)$ of an arbitrary edge e in the roadmap by first determining the starting position (x,y), direction θ, and length l relative to the deformable object. We then apply a kernel smoother [1] considering all neighboring line segments e^t in the histogram

$$C_{def}(e) \quad = \quad \left(\sum_t K\left(\frac{e-e^t}{h}\right)C_{def}(e^t)\right)\left(\sum_t K\left(\frac{e-e^t}{h}\right)\right)^{-1} (4)$$

where $K(u)$ is the multivariate Gaussian kernel with identity as covariance.

To finally answer path queries online, we apply the $A^\star$ algorithm on the roadmap and use the cost function

$$C(path) = \alpha \, C_{def}(path) + (1 - \alpha) \, C_{travel}(path), \qquad (5)$$

where $\alpha \in [0, 1]$ is a user-defined weighting coefficient that determines the trade-off between deformation– and path costs.

Although the precomputation is computationally intense, it has to be done only once for each distinct object. Furthermore, a cost function for an object can even be transferred between environments.

Given our current implementation, the robot is able to answer path queries in typical indoor environments in less than 1 second – in contrast to several hours that would be needed if the deformation simulations were carried out at run-time. For further details, we refer the reader to our previous work [8]. It should be noted that our approach makes the assumption that there are no interactions between the different deformable objects and that they are fixed in the environment, such as curtains or (rather heavy) plants.

VIII. Experimental Evaluation

Our approach has been implemented and evaluated in several experiments using a real robot and simulated data.

A. Parameter Estimation

In the first set of experiments, we estimated the elasticity parameters of two different deformable objects, namely a plush teddy bear and an inflatable ball. Additionally, we considered two different sensors: a PMD-[vision]-O3 time-of-flight sensor and a Bumblebee stereo camera.

For both sensors, the procedure is mainly identical. The manipulator is used to deform the object and to record the applied forces, the proximity sensor is used to scan the object. The resulting 3D point clouds are then aligned with the model of the object using the ICP algorithm. Based on the error function, the simulation and error minimization is carried out to find the correct elasticity parameters.

First, we deformed our plush teddy bear as shown in Figure 3 and the surface meshes were obtained with the time-of-flight sensor. With the recorded forces and corresponding surface meshes, we used our approach to estimate the elasticity parameters of the object.

Second, we also used a Bumblebee stereo camera. We deformed an inflatable ball with a diameter of approx. 40 cm and used the 3D point clouds from the stereo camera. Since the Bumblebee stereo camera has a much larger field of view than the PMD sensor, we do not require a 3D model of the object from multiple scans but can perceive the object once before the deformation to obtain a sufficient model.

Figure 4 shows three images. The left one shows a 3D point cloud observation of a ball that can be deformed. The image in the middle and right depict an error mask that illustrate the differences between the simulated deformations and the observed deformations. Here, red pixels correspond to areas of high error and dark pixels indicate a low error. The middle image depicts the error mask after our approach converged and the mask indicates a low error over the whole object. Thus, the estimated parameters lead to simulated deformations that are similar to the real ones. In contrast to that, the right image shows the error mask for suboptimal parameters. Here, large errors, especially in the area around the physical interaction (arrow) can be observed.

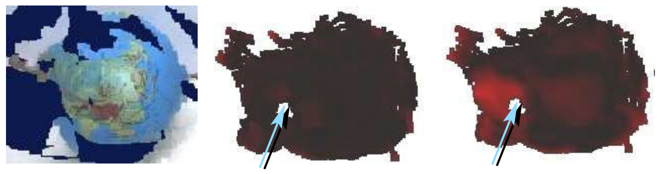

Fig. 4. Left: Color 3D point cloud of the surface of a ball. Middle: Error mask showing the differences between the simulated deformations and the observed deformations of a good parameter estimate (our approach). Red pixels indicate high errors and dark pixels indicate low errors. Right: Error mask for suboptimal parameters where the observed deformation is not in line with the prediction one (red area). The hole in the model (arrow) results from occlusions due to the physical interactions of the manipulator with the object.

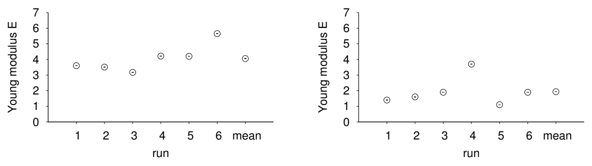

Fig. 5. Learning the deformation parameters of the inflatable ball (left) and the plush teddy bear (right). Shown are the results for different independent runs.

When analyzing the shape of the error function for the real world data, it turned out that the influence of the Young modulus on the error function is substantially larger than the one of the Poisson ratio. Therefore, we plot only the Young modulus in the subsequent evaluations (although both parameters are optimized).

Furthermore, we repeated the experiment by applying different forces to the object to evaluate the robustness of the parameter estimation. We deformed the ball with six substantially different forces and obtained six different surface scans using the Bumblebee camera. The resulting estimate for the young modulus is shown in Figure 5 (left). We can see that the estimation converges to similar values for the young modulus. We did the same with the PMD camera deforming the teddy as shown in the right image.

1) Lessons learned – Time-of-flight vs. Stereo vision: The Bumblebee stereo camera has different advantages over the PMD time-of-flight sensor: a bigger field of view, higher resolution, color information. However, to determine the depth from stereo vision, textured material is required and the teddy is comparably difficult to scan. In contrast to that, the time-of-flight camera directly provides depth information for each pixel. The depth measurements, however, are corrupted by different sources of noise, e.g. the color of objects, the background light, the distance to the camera, and even the

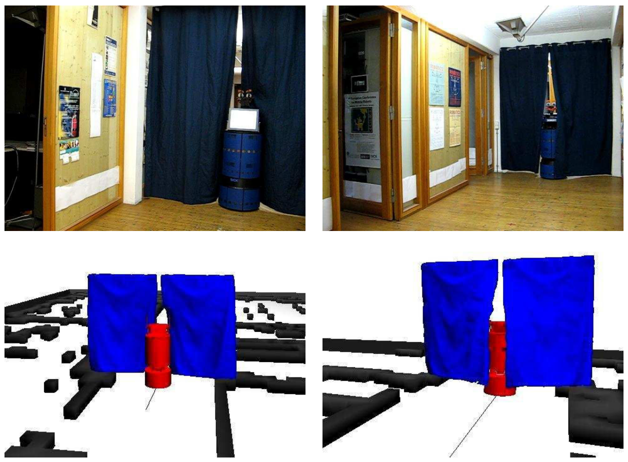

Fig. 6. The mobile robot Albert moving through a curtain in simulation as well as in the real world.

temperature. Thus, accurately mapping the surface of the ball is not trivial given that the texture resulted in different noise levels. Only for single-colored objects, such as the teddy bear, we found the results of the PMD camera acceptable.

Thus, every depth camera setup has its own advantages and disadvantages and the surface of the object under consideration has a substantial influence on the decision which sensor to prefer.

B. Path Planning

For illustrating our planner that considers deformable objects, we mounted a set of curtains in the corridor of our office environment. This setup in simulation as well as in reality is shown in Figure 6. We note here, that even for different starting and goal locations, the planner chooses the path that guides the robot in the middle between both curtains since the resulting deformation cost are in this case smaller than if the robot would travel through one curtain.

IX. Conclusions

In this paper, we presented our real robotic system, that is able to acquire models of deformable objects and to estimate their elasticity parameters. These parameters are relevant for robots that need to predict the deformations of objects in their environment depending on the forces applied to the objects. Our approach uses a mobile robot that is equipped with a manipulator, a force sensor, and a depth camera. It applies a deformation force to an object and records the resulting force-displacement relation with the force sensor and the camera. Based on a gradient descent-based error minimization approach carried out within a realistic finite element-based simulation system, the robot can determine the elasticity parameters that best explain the observed deformations. As we showed in our experiments, we are able to estimate the parameters of real objects in a robust manner.

Acknowledgment

This work has partly been supported by the German Research Foundation (DFG) under contract number SFB/TR-8.

References

[1] E. Alpaydin. *Introduction to Machine Learning*. MIT Press, 2004.

[2] K.-J. Bathe. *Finite Element Procedures*. Prentice Hall, 2 edition, 1995.

[3] P.J. Besl and N.D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992.

[4] G. Bianchi, B. Solenthaler, G. Szkely, and M. Harders. Simultaneous topology and stiffness identification for mass-spring models based on fem reference deformations. In *Med. Image Computing and Computer-Assisted Intervention*, volume 2, pages 293–301, 2004.

[5] S. Burion, F. Conti, A. Petrovskaya, C. Baur, and O. Khatib. Identifying physical properties of deformable objects by using particle filters. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, 2008.

[6] D. Chen and D. Zeltzer. Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. In *Proceedings of ACM SIGGRAPH*, 1992.

[7] F. Conti, O. Khatib, and C. Baur. Interactive rendering of deformable objects based on a filling sphere modelling approach. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, 2003.

[8] B. Frank, M. Becker, C. Stachniss, M. Teschner, and W. Burgard. Efficient path planning for mobile robots in environments with deformable objects. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, 2008.

[9] B. Frank, R. Schmedding, C. Stachniss, M. Teschner, and W. Burgard. Learning the elasticity parameters of deformable objects with a manipulation robot. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010. Currently under review.

[10] B. Frank, C. Stachniss, R. Schmedding, M. Teschner, and W. Burgard. Real-world robot navigation amongst deformable obstacles. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, 2009.

[11] R. Gayle, P. Segars, M.C. Lin, and D. Manocha. Path planning for deformable robots in complex environments. In *Proc. of Robotics: Science and Systems (RSS)*, pages 225–232, 2005.

[12] M. Hauth and W. Strasser. Corotational Simulation of Deformable Solids. In *WSCG*, pages 137–145, 2004.

[13] L.E. Kavraki, F. Lamiraux, and C. Holleman. Towards planning for elastic objects. In *Robotics: The Algorithmic Perspective*, pages 313–325. A.K. Peters, 1998. Proc. of the Third Workshop on the Algorithmic Foundations of Robotics (WAFR).

[14] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[15] J. Lang, D. K. Pai, and R. J. Woodham. Robotic acquisition of deformable models. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, 2002.

[16] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Pub., 1991.

[17] S.M. LaValle. *Planning Algorithms*. Cambridge Univ. Press, 2006.

[18] M.C. Metzger, M. Gissler, M. Asal, and M. Teschner. Simultaneous cutting of coupled tetrahedral and triangulated meshes and its application in orbital reconstruction. *International Journal of Computer Assisted Radiology and Surgery*, 4(5):409–416, 2009.

[19] M. Mueller and M. Gross. Interactive Virtual Materials. In *Graphics Interface*, pages 239–246, 2004.

[20] A. Nealen, M. Mueller, R. Keiser, E. Boxerman, and M. Carlson. Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.

[21] G. Picinbono, H. Delingette, and N. Ayache. Non-linear and anisotropic elastic soft tissue models for medical simulation. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, 2001.

[22] K. Pulli. Multiview registration for large data sets. In *Proceedings of the International Conference on 3D Digital Imaging and Modeling (3DIM)*, pages 160–168, 1999.

[23] S. Rodríguez, J.-M. Lien, and N.M. Amato. Planning motion in completely deformable environments. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pages 2466–2471, 2006.

[24] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings of the International Conference on 3D Digital Imaging and Modeling (3DIM)*, pages 145–152, 2001.

[25] J. Spillmann, M. Wagner, and M. Teschner. Robust tetrahedral meshing of triangle soups. In *Proc. Vision, Modeling, Visualization (VMV)*, pages 9–16, 2006.