Learning Cost Functions for Mobile Robot Navigation in Environments with Deformable Objects

Barbara Frank Markus Becker Cyrill Stachniss Wolfram Burgard Matthias Teschner

***Abstract*— The ability to reliably navigate through their environment is an important prerequisite for truly autonomous robots. In this paper, we consider the problem of path planning in environments with non-rigid obstacles such as curtains or plants. We present an approach that combines probabilistic roadmaps with a physical simulation of object deformations to determine a path that optimizes the trade-off between the deformation cost and the distance to be traveled. We describe how our approach utilizes Finite Element theory for calculating the deformation cost. As the high computational requirements of the corresponding simulation prevent this method from being applicable online, we present an approximative approach that uses a preprocessing step to determine a deformation cost function for each object. This cost function allows us to estimate the deformation costs of arbitrary paths through the objects and is used to evaluate the trajectories generated by the roadmap planner online. We present experiments which demonstrate that the resulting algorithm is highly accurate and at the same time allows to quickly calculate paths in environments with deformable objects.**

I. INTRODUCTION

Path planning is one of the fundamental problems in robotics, and the ability to plan collision-free paths is a precondition for numerous applications of autonomous robots. The path planning problem has traditionally received considerable attention in the past and has been well-studied. The majority of approaches, however, has focused on the problem of planning paths in static environments and with rigid obstacles [1, 2, 3]. In the real world, not all obstacles are rigid, and taking this knowledge into account can enable a robot to accomplish navigation tasks that otherwise cannot be carried out. For example, in our everyday life we deal with many deformable objects such as plants, curtains, or cloth and we typically utilize the information about the deformability of objects when planning a path. As a motivating example, consider the situation depicted in Figure 1, in which the robot needs to pass through a curtain to move from its current position to the goal location since no other path exists in the environment. In this particular situation, traditional approaches that do not take the deformability of objects into account, will fail since no collision-free path exists. In contrast, the approach presented in this paper is able to determine the deformation cost introduced by passing the curtain and to utilize this information during path planning. The key idea of our approach is to use a heuristic function to estimate the deformation cost, which allows the robot to perform the necessary calculations online.

All authors are with the Department of Computer Science, University of Freiburg, 79110 Freiburg, Germany.
{bfrank,mbecker,stachnis,burgard,teschner}@informatik.uni-freiburg.de

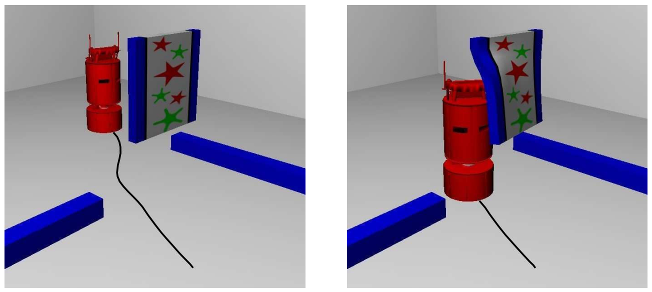


Fig. 1. Path planning in an environment with a deformable object. The robot (shown in red) deforms the curtain on its path to the goal.

One potential method of taking deformations of objects into account is by generating trajectories using a method such as probabilistic roadmaps and considering deformable objects as free space. When answering path queries, the planner has to simulate the deformation of the non-rigid objects resulting from the interaction with the robot. However, performing an appropriate simulation typically requires significant computational efforts which makes such an approach unsuitable for online trajectory planning. Therefore, we propose an approach to learn an approximative deformation cost function in a preprocessing step. The advantage of our method is that this function can be evaluated efficiently. In this way, our approach is able to reduce the time to solve a path query from several minutes to a few hundred milliseconds, as is demonstrated in the experimental section. The assumption made throughout this paper is that the robot can deform but cannot move objects in the environment. Additionally, we assume that there are no interactions between different deformable objects.

The contribution of this paper is an approach to mobile robot path planning that explicitly considers deformable objects in the environment. It employs the probabilistic roadmap method (PRM) [4] and learns a deformation cost function using a physical simulation engine that is based on Finite Element theory. Our approach trades off the travel cost with the deformation cost when answering path queries. Additionally, it can be executed online.

This paper is organized as follows: After discussing related work, we present our technique to compute the deformations of objects by means of physical simulation. We then describe how to plan a path in presence of deformable objects. Additionally, we describe our approximation of the deformation cost function. Finally, we present experiments that illustrate the advantages of our approach compared to previous methods.

II. RELATED WORK

The majority of approaches to mobile robot path planning assumes that the environment is static and that all objects are rigid [1, 4, 5]. Recently, several path planning techniques for deformable robots in static environments have been presented [6, 7, 8]. In the context of deformable objects, the underlying model has a substantial influence on the accuracy of the estimated deformations as well as on the performance of the planner. One typically distinguishes between geometrically and physically motivated approaches. Geometric approaches such as the free-form deformation (FFD) can be computed efficiently. For example, the FFD method of Sederberg and Parry [9] is based on trivariate Bernstein polynomials and allows for deformation by manipulating the control points.

To represent non-rigid objects and to calculate deformations, mass-spring systems have been frequently used. They are easy to implement and can be simulated efficiently. Whereas 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 [10]. Finite Element Methods (FEMs) reflect physical properties of the objects in a better way and allow for a more intuitive modeling since they require only a small number of parameters. The disadvantage of FEMs is the computational resources required to calculate deformations. In our current system we therefore use the computationally efficient co-rotational Finite Element approach of Mueller and Gross [11] and Hauth and Strasser [12] which avoids nonlinear computations and therefore can handle large deformations.

In the context of path planning, Kavraki *et al.* [8] developed the f-PRM-Framework that is able to plan paths for flexible robots of simple geometric shapes such as surface patches [13] or simple volumetric elements [14]. They apply a mass-spring model to compute deformations. The planner selects the deformation of the robot that minimizes its deformation energy. Similar to this technique, Gayle *et al.* [7] presented an approach to path planning for a deformable robot that is based on PRMs. To achieve a more realistic simulation of deformations they add constraints for volume preservation to the mass-spring model of the robot. Bayazit *et al.* [6] also studied planning for a deformable robot. Their algorithm proceeds in two steps. First, an approximate path is found in a probabilistic roadmap and in the next step, this path is refined by applying a free-form deformation to the robot and hence avoiding collisions with obstacles. This deformation method can be computed efficiently but is less accurate than physically motivated models. In contrast to our approach, these planners deform the robot rather than the obstacles to avoid collisions.

An approach to planning in completely deformable environments has been proposed by Rodríguez *et al.* [15]. They employ a mass-spring system with additional physical constraints for volume-preservation [16] to enforce a more realistic behavior of deformable objects. They use rapidly exploring random trees and apply virtual forces to expand the leafs of the tree until the goal state is reached. The obstacles in the environment are deformed through external forces resulting from collisions with the robot. The interesting aspect of this approach is that it is able to handle both robot deformations and deformations of obstacles in the environment.

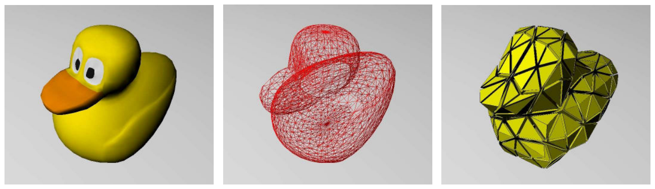


Fig. 2. Different levels of representation for a deformable object (left) in the simulation environment: fine surface mesh (middle) and coarse tetrahedral mesh (right).

All techniques mentioned above, however, require substantial computational resources and cannot be executed online in general. Our approach can efficiently answer path queries by estimating potential deformations of objects in a preprocessing step. This is achieved by approximating a deformation cost function which is then considered during the planning process. Furthermore, our deformable model is based on FEM, which allows for more accurate deformations.

III. SIMULATION OF DEFORMABLE OBJECTS

To consider non-rigid obstacles in the environment during planning, we need a model that allows for computing the deformations given an external force. In this section, we describe how we achieve a physically realistic simulation of object deformations. We will first introduce the co-rotational Finite Element model. Then we describe how to detect collisions between deformable objects and the robot and how to compute contact forces resulting from collisions. Finally, we introduce the cost resulting from a deformation.

Our simulation system proceeds as follows: in each time step, it computes deformations and unconstrained motions of objects, then it detects collisions, computes contact forces for colliding points, and finally corrects the unconstrained motion with appropriate repulsion forces.

A. Deformable Modeling

The obstacles in the workspace are 3D objects. The surface of objects is represented by a fine resolution triangle mesh. A tetrahedral mesh is used to represent the interior of these volumetric objects (see Figure 2). The actual deformations are computed on the coarse resolution tetrahedral mesh.

To compute the deformation of our tetrahedral objects we use the co-rotational Finite Element formulation proposed by Mueller and Gross [11] as well as by Hauth and Strasser [12]. The total potential energy of a single tetrahedral element e is given by

$$\Pi = U_e + WP, \tag{1}$$

with work potential WP determined by the external forces and inner energy U_e

$$U_e = \frac{1}{2}\int_e \sigma^T \epsilon \, dV. \tag{2}$$

As we assume only linear isotropic materials, we have a linear relation between the stress σ and the strain ϵ given by the generalized Hooke's law.

The idea of the Finite Element method is to discretize the object into a finite set of elements (in our case tetrahedrons) to compute the deformations based on Equation (1) on the nodes and to interpolate the deformation in the elements using the nodal values. To compute the strain ϵ from the nodal deformations in our model, we use the linear Cauchy strain tensor as it is computationally efficient. However, the Cauchy tensor is not rotationally invariant, leading to ghost forces which result in distortion for large rotational deformations. Therefore, we keep track of the rigid body motion for each element by extracting the rotation from the transformation matrix using polar decomposition. Applying the strain tensor in the rotated frame trivially leads to rotational invariance while we still have significantly less computational costs than by using a nonlinear strain tensor. We discuss the performance of our Finite Element approach compared to the versatile mass-spring approach used by Rodríguez *et al.* [15] in the experimental results section.

B. Collision Detection

For the realistic processing of interactions between the robot and the deformable objects, an efficient collision detection algorithm is required. We employ a spatial subdivision scheme in our simulation system, where the elements are stored in a hash grid [17].

The key idea of this approach is to implicitly discretize $\mathbb{R}^3$ by storing the elements and nodes in the hash grid. As space is usually filled sparsely and non-uniformly, this method consumes less memory than an explicit discretization. The hash key is computed from the coordinates of the corresponding grid cell. As a result, only the elements with the same hash key need to be checked for collisions.

To check for collisions, one computes the intersection between points and tetrahedra. This can be done efficiently using barycentric coordinates of the points with respect to the tetrahedra.

Methods commonly employed for rigid bodies, such as bounding box hierarchies [18], are less suited for deformable objects, where these spatial data structures cannot be precomputed [19].

C. Computation of Contact Forces

To handle collisions between the robot and the deformable objects, we employ the force-based collision handling scheme proposed by Spillmann *et al.* [20], which combines the advantages of penalty and constraint-based collision handling schemes. For a set of colliding points of a tetrahedral mesh, we compute a collision free state using a linearized relation between internal forces and displacements of all affected points. Contact forces can be computed analytically to obtain this collision-free state while conserving overall system energy.

Using this combination of FEM-based simulation and the collision handling described above, our system can simulate thousands of tetrahedra at interactive rates. An example implementation of the simulation system is available online [21].

D. Object Deformation Costs

The inner energy of an object, specified in Equation (2), provides a measure of the deformation costs of a tetrahedral object and thereby of the additional energy consumption of the robot. In case of an undeformed object, the inner energy is zero. Otherwise, the inner energy increases depending on the deformation of the object. For an object O with tetrahedral elements $\{e_i\}$, we define the total inner energy U_O induced by a robot r at position $\mathbf{p}$ approaching from direction θ by the sum over the inner energies of all elements e_i of the object $U_O(r, \mathbf{p}, \theta) := \sum_{e_i \in O} U_{e_i}(r, \mathbf{p}, \theta)$.

For any given position $\mathbf{p}$ and direction θ we determine the total deformation cost $C_{def}(\mathbf{p}, \theta) := \sum_O U_O(r, \mathbf{p}, \theta)$ by summing over all objects O in our workspace. The direction θ has to be taken into account, as deformable objects might deform differently when approached from different directions.

The total deformation cost on a path Γ of the robot r approaching from direction θ in our environment naturally results in the sum of the deformation costs of all objects that are deformed by the robot while moving on the path in discrete time steps t_i:

$$C_{def}(\Gamma, \theta) = \sum_{t=t_1}^{t_n} C_{def}(\mathbf{p}_r(t_i), \theta(t_i)), \tag{3}$$

where $\mathbf{p}_r(t_i)$ is the position of the robot on Γ at time t_i. $\theta(t_1)$ is given by θ, all other directions $\theta(t_i)$ are determined as the difference between $\mathbf{p}_r(t_i)$ and $\mathbf{p}_r(t_{i-1})$.

IV. PATH PLANNING WITH DEFORMABLE OBJECTS

In this chapter, we describe our path planning system and introduce the approximate deformation cost function that allows for a speedup of the path planning process in environments with deformable obstacles.

A. Overview of the Path Planning System

The general path planning problem is to find a sequence of valid robot configurations that lead from the starting to the goal configuration. Probabilistic roadmap planners achieve this by constructing a roadmap that represents the environment of the robot and by applying a graph search algorithm to find a path from the starting to the goal configuration for a given environment. The roadmap is constructed in a preprocessing step by sampling points in the configuration space of the robot. These points have to satisfy certain feasibility constraints. In general, valid configurations are required to be part of the free configuration space $\mathcal{C}_{free}$. In our situation, however, we modify this constraint to only require configurations to be in $\mathcal{C}_{free} \cup \mathcal{C}_{def}$, i.e., we also accept configurations that lead to collisions with deformable objects. In our current implementation, we use Hammersley-sampling [22] to generate configuration hypotheses in the space. This deterministic sampling method generates a sequence of points that are distributed with low discrepancy. After enough samples have been generated to cover the configuration space, a local planner connects neighboring samples, for which a valid path (one that does not lead to collisions with rigid obstacles) exists. This results in a roadmap that represents the environment of the robot and allows for planning of paths where objects are allowed to be deformed by the robot.

To answer a path query, we then insert the starting and the goal configuration into the roadmap and connect them to their neighbors. Finally, we apply $A^\star$ to find the best path from the starting to the goal location on the graph. Here, we search for the path with the best trade-off between travel costs and deformation cost. Therefore, we need a way of estimating the expected deformation cost arising on the edges of the roadmap.

The simulation system presented in the previous section can be used inside the planning algorithm to compute the deformation cost $C_{def}(i)$ of an edge i by simulating a robot moving along this edge deforming the object. The edges considered during the $A^\star$ planning are evaluated by trading deformation against travel cost. In our planning system, we assume the travel cost to be proportional to the length of the edge i. This results in the cost function

$$C(i) \quad := \quad \alpha\, C_{def}(i) + (1-\alpha)\, length(i), \tag{4}$$

where $\alpha \in (0,1)$ is a user-defined weighting coefficient. In order to obtain an admissible heuristic for $A^\star$, we use the Euclidean distance to the goal location weighted with $(1-\alpha)$

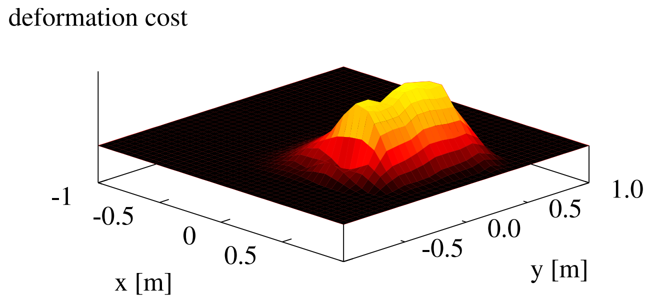
$$h(n) \quad = \quad (1-\alpha)\|g-n\|, \tag{5}$$

where g is the goal location and n the current node in the roadmap. Thus, we are able to find the path in the roadmap that optimizes the trade-off between travel cost and deformation cost for a given user-defined parameter α.

This technique leads to a working planning system that considers deformations of the objects in the environment when planning a trajectory for a mobile robot. The drawback of this technique, however, is its high computational requirements. Answering a path query typically takes several minutes even for small examples. Therefore, we developed an alternative approach that computes an approximation of the cost function in advance and thus facilitates online path planning.

B. Approximative Deformation Cost Function

The goal of the approximative cost function is to quickly provide an estimate of the deformation costs for all objects along an edge in the roadmap. Such a function can be used in the planning approach described above to determine C_{def}. The actual value of the deformation cost function of an object mainly depends on the trajectory of the robot relative to the object. Therefore, we precompute the deformation cost for a number of line segments through each object. A line is specified by a starting location (x, y) and the traveling direction θ as well as the traveled length l on the line segment. The traveled length is constrained to the maximum distance that the robot can travel while still deforming the object.



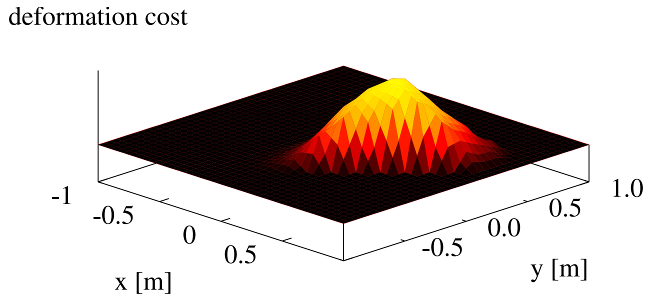


Fig. 3. Deformation costs for moving a robot along straight lines through a curtain. The lines are specified by starting points (x, y) and travel direction $\theta = 0°$ (top) resp. $\theta = 45°$ (bottom) relative to the center of mass of the obstacle.

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 $\hat{C}_{def}(x, y, \theta, l) \to \mathbb{R}$ which returns the deformation cost for edges of the roadmap.

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 [23] considering all neighboring line segments e^t in the histogram

$$\hat{C}_{def}(e) \quad = \quad \frac{\sum_t K\left(\frac{e-e^t}{h}\right)\hat{C}_{def}(e^t)}{\sum_t K\left(\frac{e-e^t}{h}\right)}, \tag{6}$$

where we use the multivariate Gaussian kernel

$$K(u) \quad = \quad \left(\frac{1}{\sqrt{2\pi}}\right)^3 \exp\left[-\frac{\|u\|^2}{2}\right]. \tag{7}$$

As distance metric between different line segments, we employ the Euclidean distance and normalize the orientation with respect to the positions.

To finally answer path queries, we apply the $A^\star$ algorithm on the roadmap. The cost of each edge in the graph is computed according to Equation (4) using the precomputed

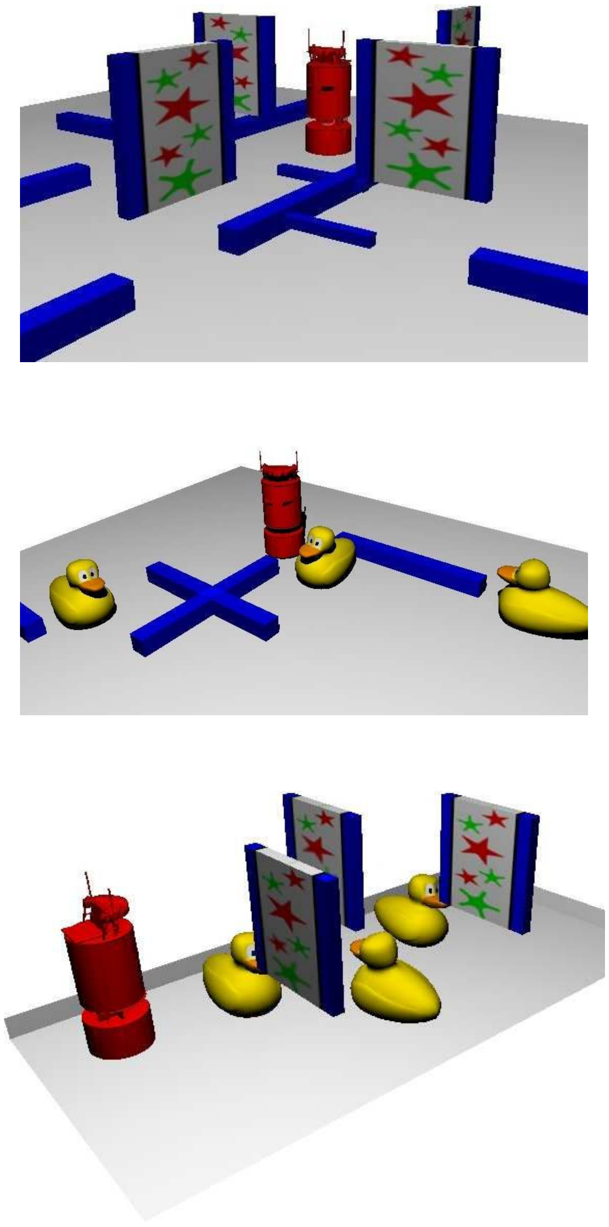


Fig. 4. Test environments: world 1 with curtains (top), world 2 with rubber ducks (middle), and world 3 with rubber ducks and curtains (bottom).

approximation $\hat{C}_{def}$ of C_{def}. Storing deformation costs in a preprocessing step dramatically increases the performance of our planner as can be seen in the experiments.

Although the precomputation is computationally intense, depending on the resolution of the approximative deformation cost function, it has to be done only once for each distinct object. The following section provides results on the runtime of the precomputation for different resolutions of the deformation cost grid. Figure 3 illustrates the deformation cost $\hat{C}_{def}$ of the curtain shown in Figure 1 along a series of straight lines.

V. EXPERIMENTS

We carried out different experiments to evaluate our path planning approach. In this section, we first compare the

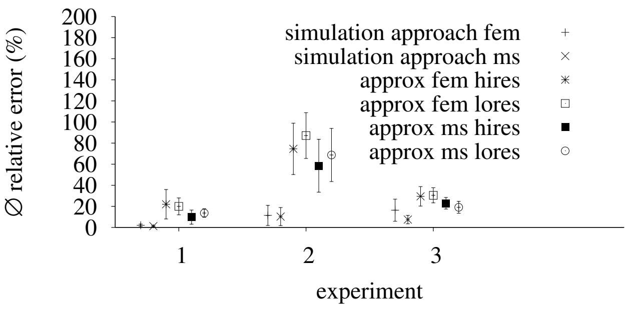


Fig. 5. Comparison of the error in the deformation cost for the simulation-based approach and our approximation.

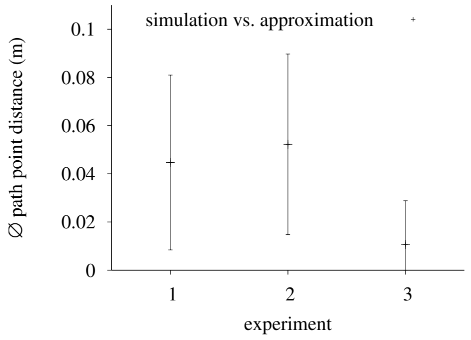


Fig. 6. Comparison of the simulation-based approach and our approximation: average deviation of the generated path points for the examples shown in Figure 7.

deformation cost obtained by the FEM-based simulation technique with our approximative solution that computes a deformation cost function for each object in a preprocessing step. We also analyze the accuracy of our cost function as well as the required execution time. Next, we investigate how the deformation cost weighting coefficient α influences the path search. Finally, we present examples for planned trajectories in environments with deformable objects. All experiments were carried out on a standard PC with 2.40 GHz Intel Core Duo Processor and an Nvidia GeForce 7950 GT graphics card. The Nvidia GPU is used by the simulation engine.

A. Cost Function and Runtime

In the first experiment, we compare the simulation approach to estimate the deformation cost with our approximative cost function. We chose curtains and rubberducks as deformable objects. The curtains are modeled to be easily deformable while the rubberducks have high deformation costs. Both approaches had to solve 25 path queries in the three test environments depicted in Figure 4. After planning, the best trajectory is sent to a path execution module that guides the robot along that trajectory. In our simulation, the execution of motion commands is affected by noise.

The experiments are carried out for different resolutions of the approximate cost function. Furthermore, we compare our approach to a simulation system using the versatile mass-spring model. We evaluated the error between predicted and measured deformation cost. The results are shown in Figure 5. As expected, the error of the simulation technique

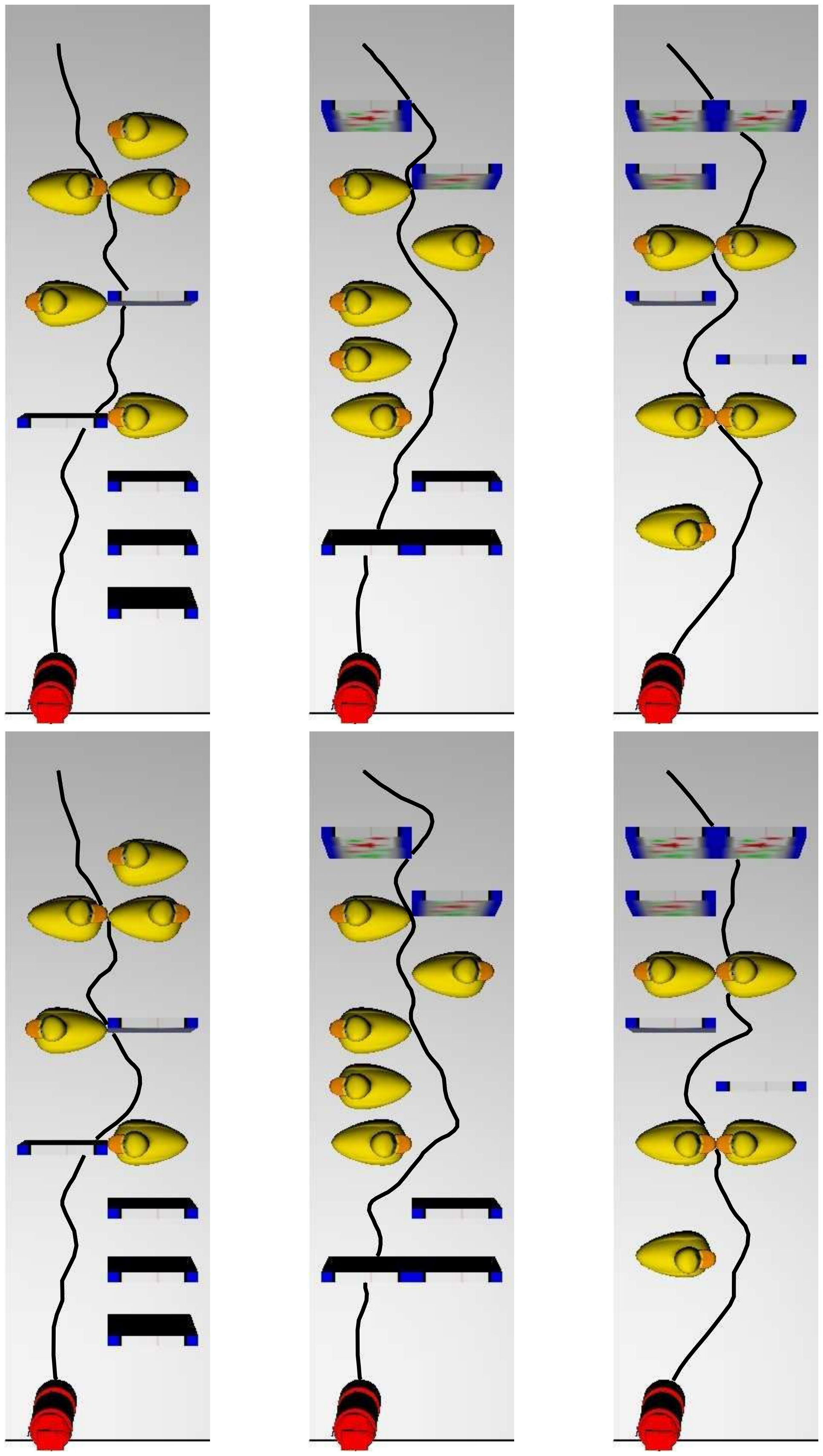
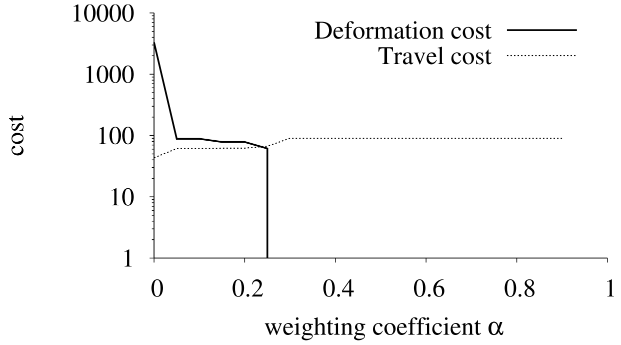


Fig. 7. Example trajectories generated by the simulation approach (top row) vs. our approximation approach (bottom row).



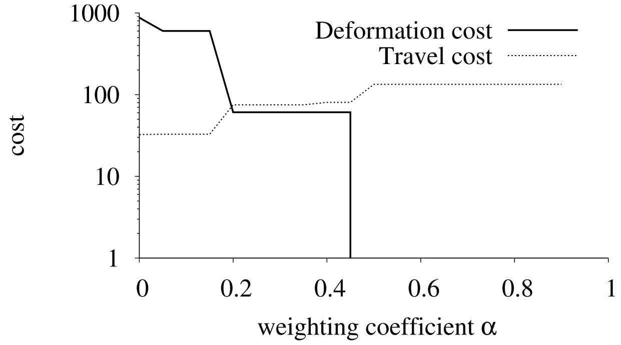


Fig. 8. Deformation and travel costs of executed trajectories shown in Figure 10 (left) and Figure 9 (right) depending on the weighting coefficient α.

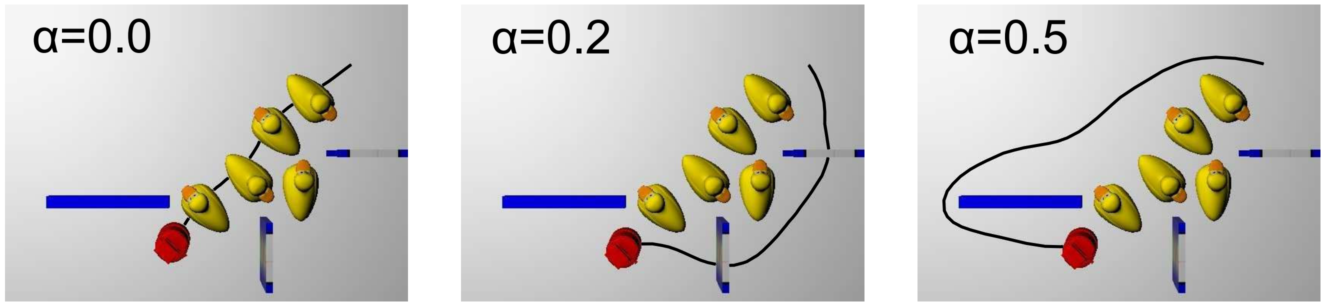


Fig. 9. Different trajectories to the goal point depending on the weighting coefficient α.

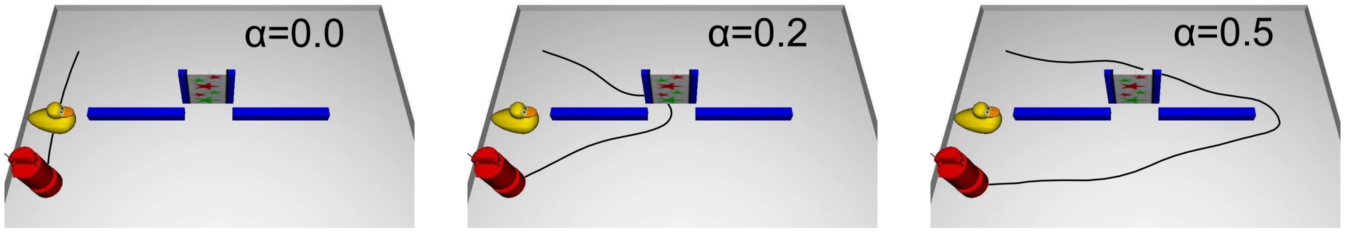


Fig. 10. Different trajectories to the goal point depending on the weighting coefficient α. In all our experiments, we set $\alpha = 0.2$.

is typically smaller compared to our approximative approach. This, however, comes at the expense of running time as illustrated in Table I. While our approach answers path queries even for complex environments in a few hundred milliseconds on average, the simulation approach spends generally about half an hour on one query. Thus, our approach is about four orders of magnitude faster than the simulation using FEM. We also compared our approach to a simulation system using the versatile mass-spring model. Although this simulation system can be evaluated faster, our approach still is about 2000 times faster. The runtime for the precomputation of the approximate cost function for different resolutions of the cost grid is summarized in Table II.

Additionally, we carried out an experiment in a randomized world, where we compare the computed paths for the simulation and the approximation approach. The generated trajectory points deviate on average by 0.09 m in an environment of 2.6×9 m as depicted in Figure 6 and the deformation costs of the trajectories deviate by $9.4 \pm 5.2\,\%$.

In most cases, the actual trajectories reported by the different planners do not deviate substantially. As the examples depicted in Figure 7 illustrate, the resulting trajectories are similar. This suggests that our approximative solution provides acceptable trajectories for planning in environments with deformable objects.

TABLE I
AVERAGE RUNTIME INCLUDING CONFIDENCE INTERVALS.

World	∅ Query (FEM)	∅ Query (mass-spring)	∅ Query (our approach)
1	36 m 41 s ± 347 s	12 m 45 s ± 112 s	0.4 s ± 0.04 s
2	30 m 33 s ± 512 s	8 m 10 s ± 73 s	0.2 s ± 0.02 s
3	29 m 43 s ± 130 s	7 m 27 s ± 36 s	0.3 s ± 0.04 s

TABLE II
RUNTIME FOR THE PRECOMPUTATION OF THE DEFORMATION COST FUNCTION FOR DIFFERENT RESOLUTIONS OF THE COST GRID.

Object	Mass-spring simulation		FEM simulation	
	coarse res. (200 lines)	fine res. (7056 lines)	coarse res. (200 lines)	fine res. (7056 lines)
curtain	17 m 48 s	10 h 37 m	1 h 11 m	41 h 16 m
rubberduck	18 m 21 s	10 h 56 m	1 h 16 m	44 h 44 m

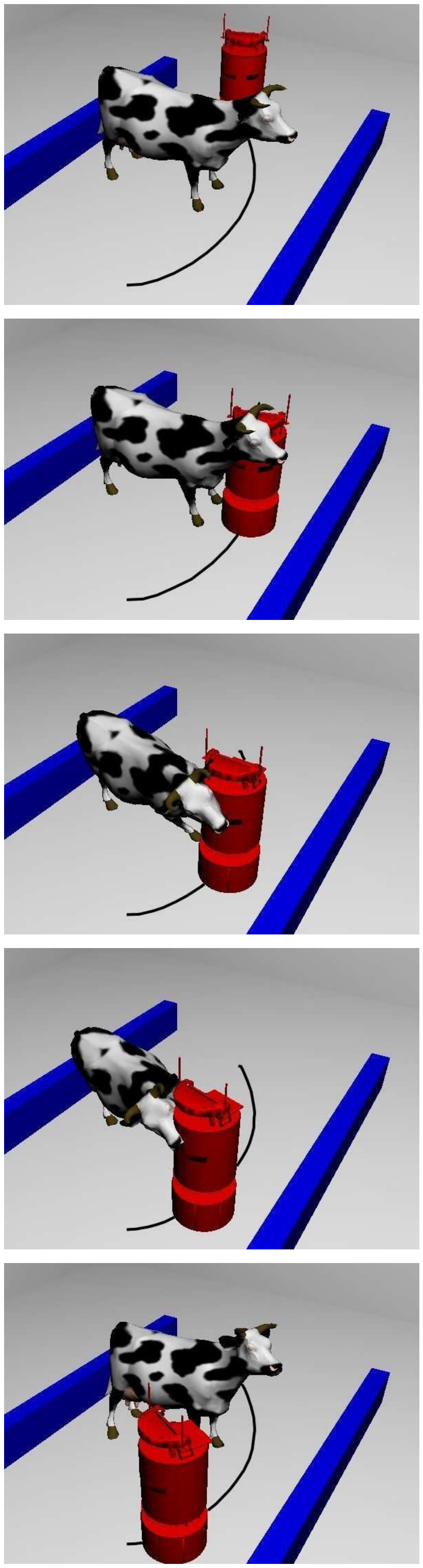


Fig. 11. Example trajectory guiding the robot through a deforming object.

B. Determination of the Weighting Coefficient

Equation (4) contains the weighting factor α that trades off the travel costs with deformation costs. To find good values for this factor, we carried out a series of planning experiments with varying values for α. Low values for α result in the fact that the robot traverses objects that are hard to deform in order to obtain a short trajectory. In contrast to this, high values for α will lead to a planning system that entirely avoids deformations if possible.

In all our experiments we set $\alpha = 0.2$. As a result, the robot selects trajectories through easily deformable objects such as curtains and tries to avoid objects that cause high deformation costs such as the rubber ducks. Figure 8 depicts the deformation costs and the travel costs for the environments shown in Figure 9 and Figure 10. These figures also depict different trajectories resulting from different values for α.

Finally, Figure 11 shows a sequence of snapshots taken during a planning experiment. They illustrate that a robot using our planning approach selects trajectories through deformable objects in case the deformation is not too expensive. More examples and animations are available at our website [24].

VI. CONCLUSIONS

In this paper, we presented an approach to path planning in environments with non-rigid objects. Our planner takes potential deformations of objects into account using a simulation engine that is based on the physically accurate Finite Element method. To improve the efficiency of the planner, our approach uses a learned, approximative deformation cost function that estimates the deformation cost of path segments relative to an obstacle. This avoids computationally expensive simulations during planning. The cost function is learned offline and is integrated into a probabilistic roadmap planner. It allows for computing a path between arbitrary starting and goal locations using $A^\star$ on the probabilistic roadmap, given our cost function that uses a trade off between the deformation costs and the travel costs. Due to the $A^\star$ search, the robot always finds the optimal path in the roadmap.

Our approach has been implemented and tested exhaustively in environments with deformable objects. Our approximative cost function leads to a speedup of about four orders of magnitude compared to the same approach without that approximation while providing similar trajectories.

Despite these encouraging results, further improvements can be done. The next goal is to acquire models of real obstacles with our robot and estimate their elasto-mechanical parameters, for example by using the method proposed in our previous work [25]. This will allow for applying our system to real world settings and accurately considering the properties of real deformable objects.

VII. ACKNOWLEDGMENTS

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

References

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

[2] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion*. MIT Press, 2005.

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

[4] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[5] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 2, pp. 224–241, 1992.

[6] O. Bayazit, J.-M. Lien, and N. Amato, "Probabilistic roadmap motion planning for deformable objects," in *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 2002, pp. 2126–2133.

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

[8] L. Kavraki, F. Lamiraux, and C. Holleman, "Towards planning for elastic objects," in *Robotics: The Algorithmic Perspective*. A.K. Peters, 1998, pp. 313–325, proc. of the Third Workshop on the Algorithmic Foundations of Robotics (WAFR).

[9] T. Sederberg and S. Parry, "Free-form deformation of solid geometric models," in *Proc. of the Conf. on Computer graphics and interactive techniques*, 1986, pp. 151–160.

[10] A. Nealen, M. Mueller, R. Keiser, E. Boxerman, and M. Carlson, "Physically Based Deformable Models in Computer Graphics," *Computer Graphics Forum*, vol. 25, no. 4, pp. 809–836, 2006.

[11] M. Mueller and M. Gross, "Interactive Virtual Materials," in *Graphics Interface*, 2004, pp. 239–246.

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

[13] C. Holleman, L. Kavraki, and J. Warren, "Planning paths for a flexible surface patch," in *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 1998, pp. 21–26.

[14] E. Anshelevich, S. Owens, F. Lamiraux, and L. Kavraki, "Deformable volumes in path planning applications," in *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 2000, pp. 2290–2295.

[15] S. Rodríguez, J.-M. Lien, and N. Amato, "Planning motion in completely deformable environments," in *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 2006, pp. 2466–2471.

[16] M. Teschner, B. Heidelberger, M. Mueller, and M. Gross, "A versatile and robust model for geometrically complex deformable solids," in *Proc. of Computer Graphics International*, 2004, pp. 312–319.

[17] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, and M. Gross, "Optimized spatial hashing for collision detection of deformable objects," in *Proc. Vision, Modeling, Visualization (VMV)*, 2003, pp. 47–54.

[18] C. Ericson, *Real-Time Collision Detection*. Morgan Kaufmann, 2005.

[19] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M. Cani, F. Faure, N. Magnenat-Thalmann, and W. Strasser, "Collision Detection for Deformable Objects," *Computer Graphics Forum*, vol. 24, no. 1, pp. 61–81, 2005.

[20] J. Spillmann, M. Becker, and M. Teschner, "Non-iterative computation of contact forces for deformable objects," *Journal of WSCG*, vol. 15, no. 1–3, pp. 33–40, 2007.

[21] B. Heidelberger, M. Teschner, J. Spillmann, M. Mueller, M. Gissler, and M. Becker, "DefColStudio – interactive deformable modeling framework," `http://cg.informatik.uni-freiburg.de/software.htm`.

[22] M. Branicky, S. LaValle, K. Olson, and L. Yang, "Quasi-randomized path planning," in *Proc. of the IEEE Int. Conf. on Robotics & Automation*, 2001, pp. 1481–1487.

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

[24] B. Frank, "Motion planning with deformable objects," 2008, `http://www.informatik.uni-freiburg.de/~bfrank/defplan`.

[25] M. Becker and M. Teschner, "Robust and efficient estimation of elasticity parameters using the linear finite element method," in *Proc. of Simulation and Visualization*, 2007, pp. 15–28.