Efficient Motion Planning for Manipulation Robots in Environments with Deformable Objects

Barbara Frank Cyrill Stachniss Nichola Abdo Wolfram Burgard

Abstract— **The ability to plan their own motions and to reliably execute them is an important precondition for most autonomous robots. In this paper, we consider the problem of planning the motion of a mobile manipulation robot in the context of deformable objects in the environment. Our approach combines probabilistic roadmap planning with a deformation simulation system. Since appropriate physical deformation simulation is computationally demanding, we use an efficient variant of Gaussian Process regression to estimate the deformation cost for individual objects based on training examples. We generate the training data as a preprocessing step offline using the physical deformation simulation system so that no simulations are needed during runtime. We implemented and tested our approach on a mobile manipulation robot. Our experiments show that the robot is able to accurately predict and thus consider the deformation cost its manipulator introduces to the environment during motion planning. Simultaneously, the computation time is substantially reduced compared to the system that employs physical simulations online.**

I. Introduction

The ability to plan its own motion is an important capability of a truly autonomous robot. There is a large body of literature on path and motion planning for mobile robots, most of them assuming a static world or environments that consist of rigid objects only. Recently, several researchers addressed the problem of dealing with deformable objects or even a deformable robot [9, 1, 2, 6, 18]. An increasing number of robots has to deal with deformable objects such as plants, pillows, cloth, or towels [13]. Other real world applications of planning in deformable environments are surgical simulations, where the injury of organs should be minimized.

In our previous work [3], we considered the problem of 2D navigation among deformable objects and similar aspects in the context of reactive collision avoidance systems [5]. In this work, we extend our planning framework towards manipulation planning in the presence of deformable objects. This transition from 2D path planning imposes different challenges for the modeling of the interactions between the robot and objects in the environments, especially the prediction of the deformation costs.

The straightforward way of considering deformations of objects during planning is to generate collision-free trajectories while considering all deformable objects as free space. When planning a path, the planner has to simulate the deformation of the objects resulting from the interaction with the robot and its manipulator and consider these additional costs online. The problem with this approach is that an appropriate physical simulation typically requires significant computational resources which makes such an approach unsuitable for realistic problems. In this paper, we present a novel approach that applies an efficient Gaussian Process regression approach to approximate the deformation cost function of objects in configuration space. This allows a robot such as the one shown in Fig. 1 including its arm to plan trajectories in the presence of deformable objects. An assumption that is made throughout this paper is that the robot can deform but cannot move objects in the environment. In addition to that, we restrict the set of possible trajectories for deforming objects—details are provided in Section IV.

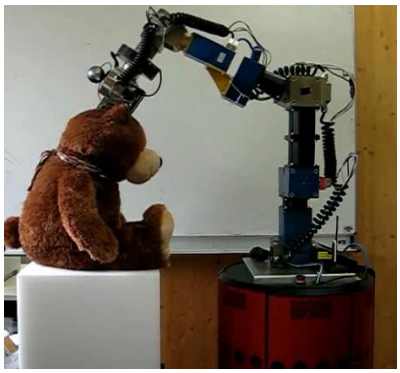

Fig. 1: Our mobile manipulation robot Zora.

II. Related Work

Recently, several path planning approaches for deformable robots in static environments have been presented [1, 2, 6, 9]. These approaches have in common that a probabilistic roadmap is used to plan motions and a deformation simulation is used to compute the expected deformations. The considered deformation models used among the different approach vary. Often, robots are assumed to be surface patches [10, 9] or consist of basic volumetric elements [1] and are modeled using spring-mass systems. To achieve a physically more realistic simulation of deformations, Gayle *et al.* [6] add constraints for volume preservation. Due to the required deformation simulations, the path planning process is computationally demanding. Bayazit *et al.* [2] apply a free-form deformation to the robot in order to avoid collisions with obstacles. This deformation method can be computed more efficiently but is less accurate than physically motivated approaches. In contrast to our approach, these planners deform

All authors are with the Department of Computer Science, University of Freiburg, Germany.

This work has partly been supported by the DFG under SFB/TR-8, by the European Commission under FP7-248258-First-MM, and by Microsoft Research, Redmond.

the robot rather than the obstacles to avoid collisions. In our approach, collisions with deformable objects are allowed but introduce additional costs. An approach to planning in completely deformable environments has been proposed by Rodríguez *et al.* [18]. They employ a spring-mass system with additional physical constraints for volume-preservation to enforce a more realistic behavior of deformable objects. Instead of probabilistic roadmaps, they use rapidly exploring random trees and apply virtual forces to expand the leaves 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. Other approaches such as [14] plan paths for a surgical tools. In this work, the organs are modeled as deformable objects and the aim is to minimize their deformation as well as penetration. This is done by optimizing the control points of a path with respect to constraints that consider the stiffness of objects and the penetration depth of the surgical tool. The tool, however, is constrained to a rod, that always has to pass through a fixed point (the insertion position), and the degrees of freedom are limited to four.

A drawback of the approaches discussed above is that they need to compute the deformation simulations during runtime. This is computationally demanding when planning the motions of real robots. In our previous work, we presented an approximation of the deformation cost functions for wheeled robots moving in a plane [3, 5] that can run online. In our new work, we extend our previous approach to the more complex problem of planning motions for manipulators with many degrees of freedom that operate in 3D world. In this setting, the possible trajectories that need to be considered are more complex and thus more sophisticated for estimating the deformation costs given a set of training examples is needed. We present an efficient approximation based on Gaussian Processes that allow to carry out motion planning tasks on the fly. Computationally demanding preprocessing steps are only needed per object type that is considered during planning. These preprocessing operations are independent of the shape of the environment itself.

In the context of robot learning tasks, Gaussian processes (GPs) are becoming increasingly popular. A good introduction into GPs can be found in [17]. In robotics, GPs have been used for terrain modeling [20], for occupancy mapping [16], for estimating gas distributions [19], learning motion and observation models [12] and several other problems. In some parts, the approach of Vasudevan *et al.* [20] is similar to our method. To model large outdoor terrain structures, they perform a nearest neighbor query on measured elevation data and consider only inputs in the local neighborhood of the query point. This is done efficiently using a KD-tree. We apply the same trick to reduce the number of training points used in the GP to the subset of the most relevant ones for solving the regression problem at hand.

III. Our Approach to Motion Planning in the Context of Deformable Objects

A. Planning using Probabilistic Roadmaps

To plan trajectories for our manipulation robot, we use the probabilistic roadmap framework [11]. The key idea of methods belonging to this class of planning algorithms is to represent a set of collision-free configurations of the robot that are considered during planning by sampled configurations. These configurations form the nodes in a graph, which is often called roadmap. In addition to the sampling, edges between nearby nodes are constructed. These edges model possible trajectories for the robot to move from one configuration to another. To plan a real trajectory of a robot given such a roadmap, one typically connects the current configuration of the robot as well as the target configuration with the graph. Most motion planning systems assign costs to the edges that correspond to their distance in configuration or works space or to the time needed to move the robot from one configuration to another. Then, this graph allows for applying graph search techniques such as $A^\star$ or Dijkstra's algorithm to search for the optimal path between a given start and goal point in the roadmap.

In the typical motion planning framework, samples in the roadmap represent collision-free configurations and trajectories between samples, i.e., the edges, are also checked for collision-free executability. Since we are interested in considering deformable objects, we need to allow for samples that lead to collisions with deformable object. Thus, when generating the probabilistic roadmap, samples that lead to collisions with deformable objects are accepted and not rejected.

To build up a motion planning system that considers deformable objects in the environment, the costs that are assigned to the edges of the roadmap need to consider the deformation costs. Our system uses a weighted sum between the distance of the nodes in configuration space and the deformation costs. For an edge between the nodes i and j, its cost is given by

$$C(i,j) \quad := \quad \alpha\, C_{def}(i,j) + (1-\alpha)\, dist(i,j), \qquad (1)$$

where $\alpha \in [0,1]$ is a user-defined weighting coefficient. The term $C_{def}(i,j)$ represents the deformation costs that are introduced by deforming objects in the environment. In case the robot does not interact with any object, this term is zero. The term $dist(i,j)$ corresponds to the distance between both nodes in configuration space. As a result, the robot prefers shorter trajectories over longer ones.

Our current implementation applies $A^\star$ to find the optimal path in the roadmap given Eq. (1). To obtain an admissible heuristic for $A^\star$, i.e., a heuristic that underestimates the real costs, we use the distance to the goal configuration weighted with $(1-\alpha)$. 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.

The key difficulty when considering deformable objects in real world planning tasks is to obtain the cost of deformations,

i.e., estimating the term $C_{def}(i, j)$, in an efficient way. One possible way to determine this quantity is to use a physical simulation engine.

B. Determining Deformation Costs via Physical Simulation

To determine the object deformations introduced by the robot and the associated costs, we employ a physical simulation engine that is based on finite element methods. In particular, we use DefCol Studio [8] as our simulation environment. It combines FEM-based simulation of the deformations on volumetric meshes following the approaches described in [7, 15], with an efficient collision handling scheme.

In our previous work, we presented an approach for building such volumetric meshes consisting of tetrahedrons from sensor data and estimating the deformation parameters for real objects [4]. The parameters, which cannot be observed directly, are estimated by actively deforming a real object and simultaneously optimizing the deformation parameters in simulation until the real shape and the simulated ones match. Here, we use the parameters estimated with our previous method [4].

C. Limitations

The approach described so far can be used for planning the trajectory of a robot and its manipulator amongst deformable objects. The key problem, however, is the computational requirements. Although the deformation simulation can be executed online for a scene, a large number of hypotheses needs to be evaluated for building up the roadmap or for planning online using $A^\star$. Additionally, small changes in the world require to recompute the costs for the edges of the roadmap—this makes real world application basically impossible. To overcome this limitation, the next section presents an efficient way to accurately estimate the deformation costs for individual objects using Gaussian Process regression. Our approach uses the simulation system to generate the training inputs and estimates the deformation costs for new trajectories or in a modified environment based on the training data that are generated beforehand. The combination of the planning system and the regression technique allows for efficient planning amongst deformable objects online.

IV. Efficient Estimation of the Deformation Cost using Gaussian Process Regression

A. Parametrization

The problem of estimating the deformation cost introduced by a robot given a set of training samples can be efficiently approached by regression techniques. Let $y_{1:n}$ be the deformation cost values obtained from n simulations where the virtual robot executed n different trajectories $\mathbf{x}_{1:n}$. Then, the goal is to learn a predictive model $p(y_* \mid \mathbf{x}_*, \mathbf{x}_{1:n}, y_{1:n})$ for estimating the deformation cost y_* given a (new) query trajectory $\mathbf{x}_*$.

In theory, all possible trajectories through a deformable objects can be executed. To bound the complexity of the regression problem, we consider only straight line motions through the object here. This is an assumption but not a really strong one since the trajectories generated by most roadmap planners are often piecewise linear motions. The motions considered to estimate the deformation cost are parametrized by five parameters: a starting point s and end point e on a virtual sphere around the robot. The points s and e are each described by an azimuth ϕ and an elevation angle θ, together with a distance l from the starting point that describes the length of the motion. Fig. 2 illustrates this parametrization. Thus, $\mathbf{x}_i$ is a five-dimensional vector in our case with $\mathbf{x}_i = [\theta_i^s, \phi_i^s, \theta_i^e, \phi_i^e, l_i]^T$ where the superscript s refers to the starting point and e to the end point.

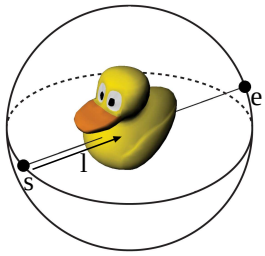

Fig. 2: Trajectory parametrization: starting point s and end point e on a virtual sphere around the deformable object together with the distance l from s towards the object.

B. Regression for Estimating Deformation Costs

We approach the problem of estimating the deformation costs by means of nonparametric regression using the Gaussian Process (GP) model [17]. In this Bayesian approach to non-linear regression, one places a prior on the space of functions using the following definition: A Gaussian process is a collection of random variables, any of which have a joint Gaussian distribution. More formally, if we assume that $\{(\mathbf{x}_i, f_i)\}_{i=1}^n$ with $f_i = f(\mathbf{x}_i)$ are samples from a Gaussian process and define $\mathbf{f} = (f_1, \ldots, f_n)^\top$, we have

$$\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}) \ , \quad \boldsymbol{\mu} \in \mathbb{R}^n, \mathbf{K} \in \mathbb{R}^{n \times n} \ . \tag{2}$$

For simplicity, we set $\boldsymbol{\mu} = \mathbf{0}$[1]. The interesting part of the GP model is the covariance matrix $\mathbf{K}$. It is specified by $[\mathbf{K}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ using a covariance function k. Intuitively, the covariance function specifies how similar two function values $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ are. The standard choice for k is the squared exponential covariance function

$$k_{SE}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2}\frac{|\mathbf{x}_i - \mathbf{x}_j|^2}{\ell^2}\right) , \tag{3}$$

where the so-called length-scale parameter ℓ defines the global smoothness of the function f and σ_f^2 denotes the amplitude (or signal variance) parameter. These parameters, along with the global noise variance σ_n^2 that is assumed for the noise component, are known as the hyperparameters of the process.

The standard squared exponential covariance function given in Eq. (3) is clearly suboptimal for our problem. The reason for that is our parametrization, which is based on four angles

[1]The expectation is a linear operator and for any deterministic mean function $m(\mathbf{x})$, the Gaussian process over $f'(\mathbf{x}) := f(\mathbf{x}) - m(\mathbf{x})$ has zero mean.

and one Euclidean distance. Considering these dimensions alike does not allow us to model the "similarity" between trajectories well. Therefore, we define a variant of the squared exponential covariance function that considers that the these angles are used to describe two points on a sphere. Thus, we consider the distance between the starting points and the end points lying on the sphere from the two inputs $\mathbf{x}_i$ and $\mathbf{x}_j$ plus the difference in the length of the trajectory. This results in

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2}\frac{d^2(\mathbf{x}_i, \mathbf{x}_j)}{\ell^2}\right), \tag{4}$$

with

$$\begin{aligned} d(\mathbf{x}_i, \mathbf{x}_j) &= ||l_i - l_j|| + ||\text{p2e}(\theta_i^s, \phi_i^s) - \text{p2e}(\theta_j^s, \phi_j^s)|| + \\ &\quad ||\text{p2e}(\theta_i^e, \phi_i^e) - \text{p2e}(\theta_j^e, \phi_j^e)|| \end{aligned} \tag{5}$$

and where $\text{p2e}(\cdot)$ is the mapping of the spherical coordinates to points on the sphere expressed in $\mathbb{R}^3$.

Given a set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ of training data obtained from the physical simulation engine, we aim at predicting the target value y_* for a new trajectory specified by $\mathbf{x}_*$. Let $\mathbf{X} = [\mathbf{x}_1; \ldots; \mathbf{x}_n]^\top$ be the matrix of the inputs and $\mathbf{X}_*$ be defined analogously for multiple test data points. In the GP model, any finite set of samples is jointly Gaussian distributed. To make predictions at $\mathbf{X}_*$, we obtain the predictive mean

$$\mathbb{E}[f(\mathbf{X}_*)] = k(\mathbf{X}_*, \mathbf{X}) \left[k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right]^{-1} \mathbf{y} \tag{6}$$

and the (noise-free) predictive variance

$$\begin{aligned} \mathbb{V}[f(\mathbf{X}_*)] &= k(\mathbf{X}_*, \mathbf{X}_*) - k(\mathbf{X}_*, \mathbf{X}) \\ &\quad \left[k(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right]^{-1} k(\mathbf{X}, \mathbf{X}_*), \end{aligned} \tag{7}$$

where $\mathbf{I}$ is the identity matrix and $k(\mathbf{X}, \mathbf{X})$ refers to the covariance matrix built by evaluating the covariance function $k(\cdot, \cdot)$ for all pairs of all row vectors $(\mathbf{x}_i, \mathbf{x}_j)$ of $\mathbf{X}$.

In sum, Eq. (6) provides the predictive means for the deformation cost when carrying out a movement along $\mathbf{x}^*$ and Eq. (7) provides the corresponding predictive variance.

C. Efficient Regression by Problem Decomposition

With the GP model explained above, we can make predictions for a set of trajectories deforming an object given training data obtained form the physical simulation. The key problem in practice, however, is that a substantial set of training data is required to obtain accurate predictions of the deformation cost. For the objects we experimented with, around 3000 training trajectories are needed for obtain good predictions. The GP framework, however, has a runtime that is cubic in the number of training examples so that the approach gets rather inefficient for more than 1000 training examples.

Therefore, we decompose the overall regression problem into a number of local ones. For a query trajectory $\mathbf{x}^*$, we determine its M closest neighbors from the training data under our covariance function given in Eq. (4) and Eq. (5) as

$$\mathbf{X}'(\mathbf{x}^*) = [\mathbf{x}'_1; \ldots \mathbf{x}'_M] = \underset{[\mathbf{x}'_1; \ldots; \mathbf{x}'_M]}{\operatorname{argmin}} \sum_{k=1}^{M} d(\mathbf{x}'_k, \mathbf{x}^*). \tag{8}$$

The M closest neighbors $\mathbf{X}'$ to the query trajectory $\mathbf{x}^*$ are the training data points that have the highest influence on the prediction of y^* in the GP framework. Considering only $\mathbf{X}'$ instead of $\mathbf{X}$ in the GP is equivalent to assuming that $k(\mathbf{x}^*, \mathbf{x}_i) = 0$ for all $\mathbf{x}_i$ that are not part of $\mathbf{X}'$. In our current implementation, we are able to get appropriate prediction by setting $M = 50$. We experienced that the loss is negligible with respect to larger values of M, at least in all in our experiments. Determining the M closest neighbors to $\mathbf{x}^*$ can be computed efficiently by a KD-tree that is once built from the training data. Thus, queries can be obtained in logarithmic time in the number of training examples and the GP prediction does not depend on the size of the training set anymore but only on M.

D. Considering the Full Kinematic Chain for Estimating the Deformation Cost

The deformation simulation system considers the movement of a rigid sphere with a diameter that is equal to that of the robot's manipulator along the described trajectory to compute the deformation cost. It does not consider the full configuration of the arm. This is done intentionally (the simulation supports for that) and it is clearly an approximation but it allows us to parametrize the regression problem with a low-dimensional input. Otherwise, the full configuration of the robot would need to be considered in the GP framework. With higher-dimensional inputs, a much larger number of training examples would be needed. To take into account the fact that not only the end-effector but also other body parts may deform an object, we sample multiple points along the kinematic chain of the robot. Then, we perform the estimation of the deformation cost for all sampled points along the kinematic chain and consider the maximum of the individual costs

$$C_{def} = \max_b \text{GP}(\mathbf{x}^*(b), \mathbf{X}'(\mathbf{x}^*(b)), \mathbf{y}'(\mathbf{x}^*(b))), \tag{9}$$

where b refers to the individual body parts and $\mathbf{x}^*(b)$ to the motion that the body parts carry out given the kinematic structure of the robot. Considering the maximum in Eq. (9) instead of, for example, the sum, typically generates more accurate predictions since the largest deformation forces are typically generated by one body part only.

In theory, there may be situations in which this assumption is not valid, for example when a robot with two manipulators would squeeze an object—such situations, however, are rarely observed in most practical settings.

V. Experimental Evaluation

A. Prediction of Deformation Costs

In this section, we evaluate our GP-based regression technique for predicting the deformation costs of robot trajectories. To show the effectiveness of the GP-based technique, we furthermore compare it to a standard nearest-neighbor prediction, which uses the average of the M nearest neighbors as an estimate.

Our deformable object is a plush teddy bear for which we estimated the deformation parameters. To learn the

TABLE I: Performance comparison for GP-based regression and nearest-neighbor approximation.

Dataset	RMSE NN	RMSE GPStandard	RMSE GPOpt	∅ time (ms) GPStandard	∅ time (ms) GPOpt
	leave-one-out				
D1	24.3	18.4	9.2	26.3	48.2
D2	19.5	27.0	5.8	19.3	42.9
D12	18.0	15.2	7.5	46.9	69.7
	cross-validation				
D1 on D2	26.9	22.5	17.8	19.4	42.1
D2 on D1	17.3	14.6	9.4	25.0	46.5

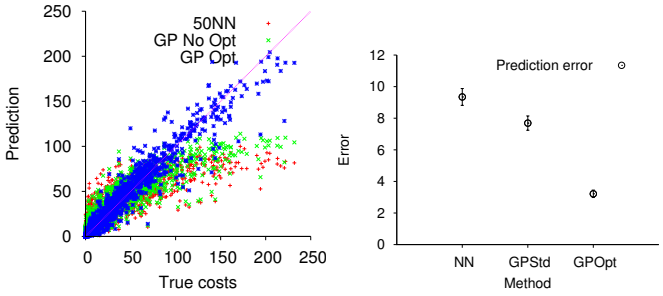

Fig. 3: Comparison of the prediction performance for Nearest-Neighbor estimation and GP-Regression (Leave-One-Out cross-validation on D12).

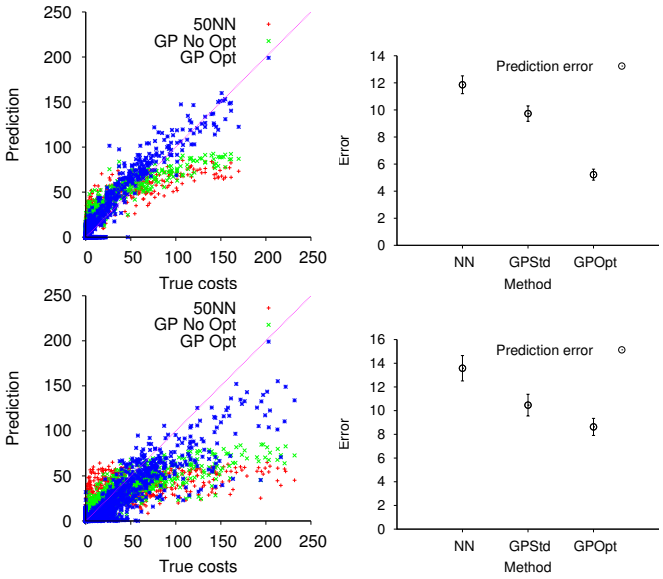

Fig. 4: Comparison of the prediction performance for Nearest-Neighbor estimation and GP-Regression(top row: Cross-validation D2 on D1, bottom row: Cross-validation D1 on D2).

deformation cost function of the teddy bear, we generated a set of (trajectory, deformationcost) samples by performing deformation simulations for the trajectory parameters. Since the computation of sample trajectories is time-consuming, we restrict the manipulation movements to those movements in the plane at different z-levels. Note that this can easily be generalized to arbitrary trajectories in 3D.

We consider 3 different data sets, which are D1 with 1,800 trajectory samples at z = 0, 20, and 40 cm, D2 with 1,400 trajectory samples at z = 10 and 30 cm, and D12 which is the combination of D1 and D2 with 3,200 trajectory samples. To evaluate the accuracy of the deformation cost prediction, we performed two different experiments, namely leave-one-out cross-validation for D1, D2, and D12 as well as cross-validation of D1 on D2 and vice versa. We compare the prediction results for the 50 nearest neighbor prediction (NN), the prediction of a GP with standard hyperparameters (GPStandard), and the prediction of a GP with optimized hyperparameters (GPOpt). The results for the different data sets are summarized in Tab. I.

Whereas a visual comparison for the leave-one-out validation is shown in Fig. 3, the results for the cross-validation are depicted in Fig. 4.

B. Performance

In this section, we analyze the computational overhead introduced by our estimation of the deformation cost compared to a standard roadmap planner.

1) Generation of path examples: The computation of the example trajectories is done offline in a preprocessing step and is quite time-consuming. The simulation of one example trajectory takes on average 40 s for a trajectory length of approximately 1 m. To obtain the 3,200 path examples we used in our evaluation, the total simulation time was around 129,417 s (approx. 36 h).

2) Roadmap Computation: The roadmap for a given static and non-deformable environment is computed in a preprocessing step. The main computational load comes from collision checks that need to be performed in order to determine edges that can be connected by collision-free paths. This is independent of our deformation cost estimation and takes for our test scenario with 1,000 sample configurations and 7,306 collision-free edges around 40 min. This could further be improved by using a more sophisticated collision checking algorithm. Evaluating the deformation cost for the 7,306 edges additionally takes 105 s (note that only the edges intersecting the bounding sphere of the deformable object need to be further analyzed using our GP based regression. This were 801 edges in this example).

3) Answering Path Queries: To answer path queries, starting and goal configurations need to be added to the roadmap. This means that the planner attempts to connect these configurations to the M nearest neighbors in the roadmap. The time-consuming factor here is again the collision-checking. We evaluated 12 path queries. Connecting them to the roadmap took on average 3.5 s. The necessary evaluation of the deformation costs of the collision-free edges additionally requires 1.8 s on average.

4) Comparison to a Roadmap Planner with Integrated Deformation Simulation: Instead of precomputing sample trajectories and estimating the deformation costs of edges in the roadmap using regression, it would be possible to perform the simulation of the edges directly when constructing the roadmap. Considering the example above, evaluating 801 edges requires an additional 267 min when constructing the roadmap. Furthermore, when answering path queries, we need to connect the start and the goal by adding new edges for which simulations need to be performed online. This requires

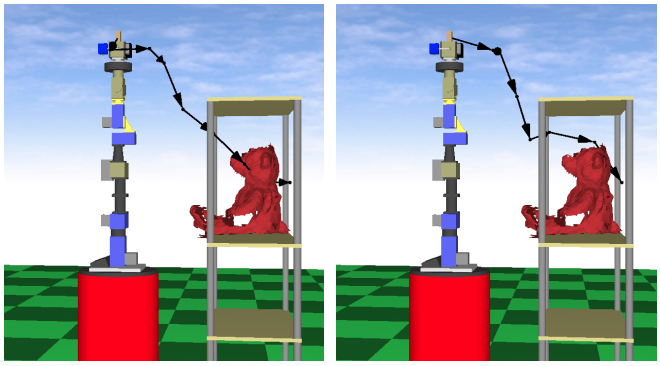

Fig. 5: Planning example. Left image: shortest path, right image: trade-off between path cost and deformation cost.

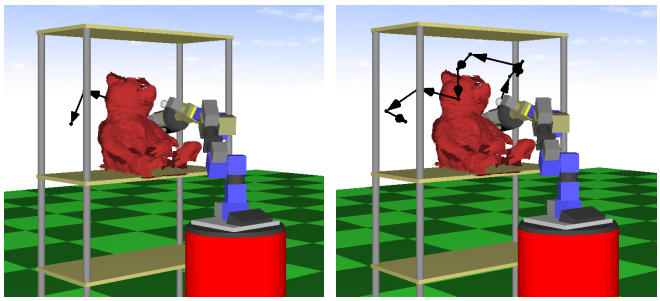

Fig. 6: Planning example. Left image: shortest path, right image: trade-off between path cost and deformation cost (only minimal deformations occur here).

another 10 min per path query. In contrast to that, our GP-based approach adds an overhead of approximately 1.8 s, thus requiring two orders of magnitude less computation time.

C. Example Trajectories

Finally, we carried out two planning experiments that are designed to illustrate the generated trajectories of our planner. We placed the teddy bear, which is deformable, in a shelf that is considered as a static obstacle. In both experiments, the robot had to move its arm from the current location to a goal. Once, the target is behind the teddy bear (Fig. 5) and once on the other side (Fig. 6). In both cases, the planner that does not consider the deformation costs would lead to significant deformation (left images) whereas our approach results in less deformation and still short paths (right images).

VI. Conclusion

In this paper, we presented a novel approach for efficiently planning the motion of a manipulation robot in environments that contain deformable objects. Our planner is based on probabilistic roadmaps and considers deformation costs that are computed from a physical simulation engine. To overcome the high computational demands of an appropriate physical deformation simulation, our approach employs an efficient variant of Gaussian Process regression to estimate the deformation cost for individual objects based on training examples. To limit the complexity of the regression problem, we train the Gaussian Process only based on the most relevant training data given a specific query trajectory. The training data is generated offline in a preprocessing step using the physical deformation simulation system so that no simulations are needed during runtime. Our experimental evaluation shows that our approach enables the robot to accurately estimate the expected deformation cost that its manipulator introduces to the objects in the scene along its path. It furthermore shows that our method substantially reduces the computation time compared to an approach that completely relies on the simulation engine during planning.

References

[1] E. Anshelevich, S. Owens, F. Lamiraux, and L.E. Kavraki. Deformable volumes in path planning applications. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pages 2290–2295, 2000.

[2] O.B. Bayazit, J.-M. Lien, and N.M. Amato. Probabilistic roadmap motion planning for deformable objects. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pages 2126–2133, 2002.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] M. Hauth and W. Strasser. Corotational Simulation of Deformable Solids. In *Int. Conf. on Computer Graphics, Visualization, and Computer Vision (WSCG)*, pages 137–145, 2004.

[8] 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`.

[9] C. Holleman, L.E. Kavraki, and J. Warren. Planning paths for a flexible surface patch. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, pages 21–26, 1998.

[10] L.E. Kavraki, F. Lamiraux, and C. Holleman. Towards planning for elastic objects. In *Proc. of the Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 313–325, 1998.

[11] 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.

[12] J. Ko and D. Fox. GP-BayesFilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 2009.

[13] J. Maitin-Shepard, J. Lei M. Cusumano-Towner, and P. Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[14] B. Maris, D. Botturi, and P. Fiorini. Trajectory planning with task constraints in densely filled environments. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

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

[16] S. O'Callaghan, F.T. Ramos, and H.F. Durrant-Whyte. Contextual occupancy maps incorporating sensor and location uncertainty. In *Proc. of the Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[17] C. E. Rasmussen and C. K.I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[18] 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.

[19] C. Stachniss, C. Plagemann, and A.J. Lilienthal. Gas distribution modeling using sparse gaussian process mixtures. *Autonomous Robots*, 26:187ff, 2009.

[20] S. Vasudevan, F.T. Ramos, E.W. Nettleton, and H.F. Durrant-Whyte. Gaussian process modeling of large scale terrain. *Journal of Field Robotics*, 26(10), 2009.