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 autonomous robots. In this paper, we consider the problem of planning the motion of a mobile manipulation robot in the presence of deformable objects. Our approach combines probabilistic roadmap planning with a physical deformation simulation system. Since the physical deformation simulation is computationally demanding, we use efficient Gaussian process regression to estimate the deformation cost for individual objects based on training examples. We generate the training data by employing a simulation system in a preprocessing step. Consequently, 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 a 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 assume a static world or environments that consist of rigid objects only. Recently, several researchers addressed the problem of planning for deformable robots [2, 3, 8] or the problem of dealing with deformable environments [19] or deformable objects such as cloth or towels [13]. Real world applications of planning in deformable environments include surgical simulations, where the interaction with (and potential injury of) organs should be minimized [8, 14].

A straightforward way of considering deformations of objects during planning is to generate collision-free trajectories while considering all deformable objects as free space. During path planning, 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. The problem with this method is that an appropriate physical simulation typically requires substantial computational resources, which makes such an approach unsuitable for realistic problems. In our previous work [5, 7], we considered the problem of 2D navigation among deformable objects in the context of reactive collision avoidance systems. Our previous method approximated the deformation cost function using a low-dimensional grid to allow for an efficient estimation of the expected deformation cost. Such an approach, however, is no longer feasible for manipulation robots due to their typically high-dimensional configuration spaces.

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

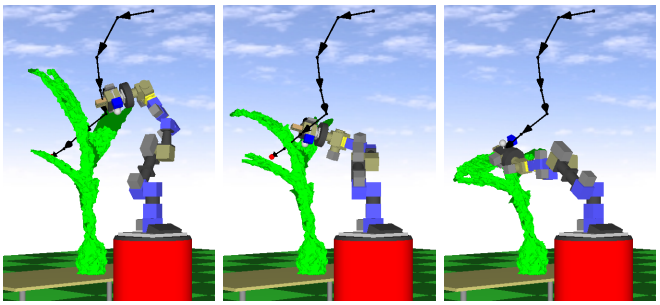

Fig. 1: Our mobile manipulation robot Zora moves along a trajectory and deforms the leaves of the plant to reach the goal.

In this paper, we present a novel approach that applies efficient Gaussian process regression to approximate the deformation cost functions of objects in the configuration space of the robot. This allows the robot to efficiently plan trajectories in the presence of deformable objects even for manipulation robots such as the one shown in Fig. 1. Throughout this paper, we assume that the robot can deform the objects but does not move them in the environment. To improve the efficiency of the learning process, we furthermore sample a restricted set of trajectories only. In different experiments, we demonstrate that our approach yields accurate estimates and, at the same time, allows for efficient planning of trajectories along which the robot interacts with deformable objects.

II. Related Work

Several path planning approaches for deformable robots in static environments have been presented [2, 3, 8]. These approaches have in common that a probabilistic roadmap is applied to plan motions and a deformation simulation is used to compute the expected deformations. Robots typically consist of primitive volumetric elements [2] and are modeled using mass-spring systems [2] or computationally efficient free-form deformations [3]. To achieve a physically more realistic simulation of deformations, Gayle *et al.* [8] add constraints for volume preservation. 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.* [19]. They employ a mass-spring system with additional physical constraints for volume-preservation to enforce a more realistic behavior of deformable objects and a path is found using rapidly exploring random trees. Planning for surgical tools in deformable environments was addressed e. g. in [1, 14]. While Alterovitz *et al.* [1] plan needle

placement in the 2D plane and account for deformations using an FEM simulator similar to ours, Maris *et al.* [14] plan paths for a surgical tool using a 3D simulation and optimizing the control points of a path with respect to constraints that consider the stiffness of objects and the penetration depth of the tool. Recently, Patil *et al.* [17] presented an extension to [1] that also considers the potential uncertainty during path execution in the planning process and chooses the path with the highest probability of success.

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 function for wheeled robots moving in a plane [5, 7] that can run online. In our new work, we extend our previous approach to the more complex problem of planning motions for manipulators that operate in 3D. In this setting, the possible trajectories that need to be considered are more complex and thus a more sophisticated method for estimating the deformation costs is needed. We present an efficient approximation based on Gaussian processes that allows the robot 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 [18]. In robotics, GPs have been used for terrain modeling [21], learning motion and observation models [12] and other problems [16, 20]. In some parts, the approach of Vasudevan *et al.* [21] 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 samples for 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 Presence 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 is to represent the collision-free configuration space of the robot by a set of samples that form the nodes of a graph. Edges in this graph model feasible trajectories between neighboring configurations. Such a roadmap can be precomputed given a model of the environment. To actually plan a trajectory for the robot, one connects the current robot configuration as well as the target configuration to the graph. Most motion planning systems assign costs to the edges that correspond to their distance in configuration or work space. 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 starting and goal point in the roadmap.

Since we are interested in considering deformable objects, we also allow for samples and edges that lead to collisions with these objects when generating the probabilistic roadmap. Accordingly, we need to consider the deformation costs when planning trajectories. 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 costs that are introduced when the robot deforms objects along its trajectory and the term $dist(i,j)$ corresponds to the distance between nodes in configuration space.

When constructing the roadmap, we uniformly sample nodes from the configuration space of the manipulator. For each node, we consider the M nearest neighbors and add an edge if there is a straight line path between both nodes that does not lead to collisions with *non-deformable* obstacles. In our implementation, M was set to 50. Our current implementation furthermore 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.

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 perform a physical simulation of the robot movement.

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 (FEM). In particular, we use DefCol Studio [10] as our simulation environment. It combines an FEM-based simulation of the deformations on volumetric meshes following the approaches described in [9, 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 [6]. Here, we assume isotropic elastic material and determine the Young modulus and the Poisson ratio of objects. These parameters describe the stiffness and the compressibility of the material, respectively. Both parameters cannot be observed directly and are therefore estimated by actively deforming a real object and by optimizing the deformation parameters in simulation until the observed and the simulated deformation match.

To define a measure for the deformation cost introduced by the robot, we use the inner potential energy of the object. It is computed in each simulation timestep based on the external forces (resulting from collisions with the robot) and

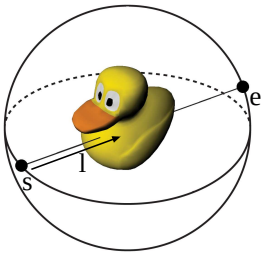

Fig. 2: Trajectory parametrization: starting point s and end point e on a virtual sphere around the deformable object together with the distance l from s towards the object.

corresponds to the deformation of the object. The deformation cost of a trajectory hence is the sum of the deformation costs of all objects that are in collision with the robot over all timesteps. For further details, we refer the reader to our previous work [5].

C. Limitations

The approach described so far can be used for planning the trajectory of a robot and its manipulator among 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 trajectory hypotheses needs to be evaluated for building the roadmap as well as 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 applications 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 among deformable objects online.

One assumption we make here is that objects can be deformed but cannot move, which is the case for instance when considering plants with deformable leaves or curtains that are fixed to the ceiling. Furthermore, our preprocessing simulations consider only interactions between the robot and an object, but interactions between different deformable objects are neglected.

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

A. Parametrization

The problem of estimating the deformation cost introduced by a robot can be efficiently approached by regression techniques. Let $y_{1:n}$ be the deformation cost values obtained from simulation 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 object 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 object. 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.

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 [18]. 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 due to our parametrization, which is based on four angles 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 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

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

[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.

with

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

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

Given a set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ of training data obtained from the 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

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

and the (noise-free) predictive variance

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

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 $\mathbf{X}$.

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

To find the best model for our prediction task, we need to determine the hyperparameters $\Theta = (\ell, \sigma_f, \sigma_n)$ that maximize the marginal log likelihood of the data given the model $p(y \mid X, \Theta)$. We use a standard gradient optimization procedure. More details on that can be found in [18].

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 from 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. 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 distance function given in Eq. (5) as

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

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 accurate predictions by setting $M = 50$. We experienced that the loss is negligible with respect to larger values of M, at least in all of our experiments. Determining the M closest neighbors to $\mathbf{x}^*$ can be computed efficiently by using a kd-tree that is built once 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 the robot's endeffector along the described trajectory to compute the deformation cost for the training samples of the GP. It does not consider the full configuration of the arm. This 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.

The planner, however, should be able to account for the fact that not only the end-effector, but also other parts of the manipulator might lead to object deformations along a trajectory. Therefore, when evaluating the deformation cost of an edge in the roadmap, we sample multiple points along the manipulator and examine the trajectories described by these points, e. g. the trajectory of the wrist and the elbow. We evaluate these trajectories using our GP prediction and consider the maximum of the individual costs

$$\hat{C}_{def} = \max_b \text{GP}(\mathbf{x}^*(b), \mathbf{X}'(\mathbf{x}^*(b)), \mathbf{y}'(\mathbf{x}^*(b))), \quad (9)$$

when determining the deformation cost of an edge in the roadmap. Here 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.

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. Our deformable object is a plush teddy bear for which we estimated the deformation parameters. To learn the deformation cost function of the teddy bear, we generated a set of samples by performing deformation simulations for different trajectory parameters. Since the simulation of sample trajectories is time-consuming, we restrict the manipulation movements to 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

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

Dataset	RMSE NN	RMSE GPStd	RMSE GPOpt	∅ time (ms) GPStd	∅ 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

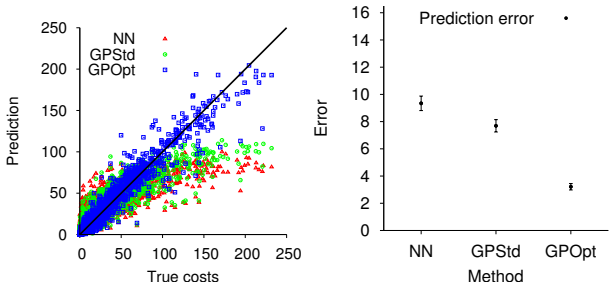

Fig. 3: Comparison of the prediction performance for nearest-neighbor estimation and GP-Regression (leave-one-out cross-validation on D12).

leave-one-out cross-validation for D1, D2, and D12 as well as cross-validation of D1 on D2 and vice versa. In all experiments, the considered set of trajectory examples was stored in a kd-tree. For predicting the cost of a trajectory, we selected the 50 nearest neighbors and constructed a GP from these 50 neighbors. We compare the prediction accuracy for a GP initialized with all hyperparameters set to 1 (GPStd) and a GP where the hyperparameters are optimized in each run (GPOpt). As a baseline, we furthermore compare it to the prediction obtained by averaging over the 50 nearest neighbors (NN). 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. The prediction accuracy of the GP with optimized hyperparameters is significantly better than the nearest neighbor prediction.

B. Performance

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

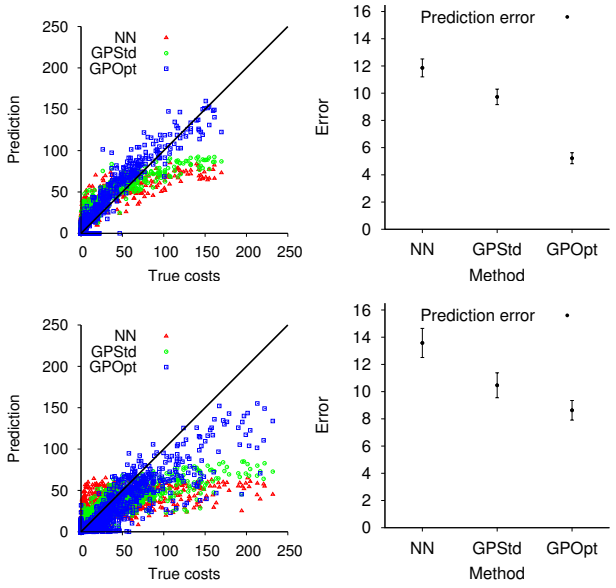

Fig. 4: Comparison of the prediction performance for nearest-neighbor estimation and GP-Regression (top row: Cross-validation D2 on D1, bottom row: Cross-validation D1 on D2).

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 was 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 ($M = 50$). 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 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 a deformable plush teddy bear on a shelf that

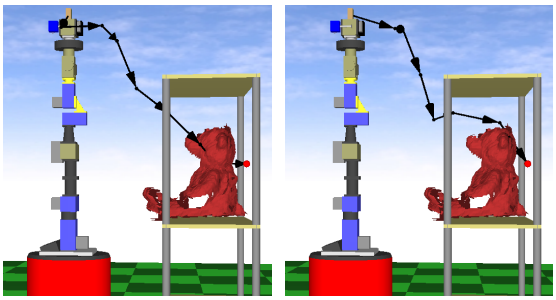

Fig. 5: Planning example. Left image: shortest path ($\alpha = 0$), right image: trade-off between path cost and deformation cost ($\alpha = 0.5$).

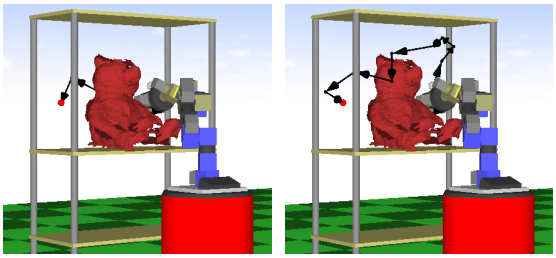

Fig. 6: Planning example. Left image: shortest path ($\alpha = 0$), right image: trade-off between path cost and deformation cost ($\alpha = 0.5$, only minimal deformations occur here).

is considered as a static obstacle. In both experiments, the robot had to move its arm from its current configuration to a goal. In the first example, the target was behind the teddy bear (Fig. 5) and in the second example it was on the other side of the bear (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). Videos of the robot executing these trajectories in a deformation simulation and more examples can be found on our website [4].

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.

Acknowledgments

The authors would like to thank Axel Rottmann for sharing his expertise on Gaussian process regression. 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.

References

[1] R. Alterovitz, K. Goldberg, J. Pouliot, and I. Hsu. Sensorless motion planning for medical needle insertion in deformable tissues. *IEEE Trans. on Information Technology in Biomedicine*, 13(2):217–225, 2009.

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

[3] O. Bayazit, J. Lien, and N. Amato. Probabilistic roadmap motion planning for deformable objects. In *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2002.

[4] B.Frank. Motion planning in environments with deformable objects. http://www.informatik.uni-freiburg.de/~bfrank/defplan.html.

[5] 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 and Automation (ICRA)*, 2008.

[6] 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.

[7] 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 and Automation (ICRA)*, 2009.

[8] 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.

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

[10] B. Heidelberger, M. Teschner, J. Spillmann, M. Mueller, M. Gissler, and M. Becker. DefColStudio 1.1.0. http://cg.informatik.uni-freiburg.de/software.htm, 2006.

[11] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. 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 and 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*, 2004.

[16] S. O'Callaghan, F. Ramos, and H. Durrant-Whyte. Contextual occupancy maps incorporating sensor and location uncertainty. In *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2010.

[17] S. Patil, J. van den Berg, and R. Alterovitz. Motion planning under uncertainty in highly deformable environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2011.

[18] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[19] S. Rodríguez, J. Lien, and N. Amato. Planning motion in completely deformable environments. In *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, 2006.

[20] C. Stachniss, C. Plagemann, and A. Lilienthal. Gas distribution modeling using sparse gaussian process mixtures. *Autonomous Robots*, 26:187–202, 2009.

[21] S. Vasudevan, F.T. Ramos, E.W. Nettleton, and H.F. Durrant-Whyte. Gaussian process modeling of large scale terrain. *J. Field Robotics*, 26(10):812–840, 2009.