Improving Path Execution in Deformable Environments Using Reactive Cost-space Control

Calder Phillips-Grafflin and Dmitry Berenson
Worcester Polytechnic Institute
{cnphillipsgraffl, dberenson}@wpi.edu

I. Introduction

We present a reactive controller designed to improve the execution of cost-space paths in the presence of sensor and modeling error that would otherwise result in significantly increased cost. In particular, we are concerned with the execution of paths planned in deformable environments, such as those produced using our cost-space planners [1]. In our previous work, executions of paths produced by these planners have suffered from significant unexpected object deformation.

Execution of planned paths in cost spaces is complicated by errors and inaccuracy in the robot's sensors, uncertainty in modeling the environment, error in the robot's controllers, and fundamental limitations of the motion planner. When executed, these paths can incur significantly higher cost than predicted by the planner's cost function. Modeling and sensor limitations are particularly pronounced with deformable environments, as the available models for deformable objects trade accuracy for reduced computational complexity [2], [3], [1]. To produce motion plans in acceptable time limits, existing planners use lower resolution or simplified models such as the voxel-based model introduced in our previous work [1]. Because these simplified models are inherently inaccurate, planners using them will fail to fully capture the behavior of the deformable objects.

Similar issues are faced with rigid environments when using multiple-resolution models [4], [5], however, in these cases an accurate high-resolution model can be used inside the planning process [5] or for post-processing such as [6], [7] using trajectory optimizers such as AICO [8], CHOMP [9], or STOMP [10]. Such an approach can compensate for planning with a reduced resolution to improve planning time. However, since an accurate model of a deformable environment may not be available (indeed, it may be infeasible to simulate the deformable behavior), it may not be practical (or possible) to optimize paths in deformable environments before execution.

Another challenge arises from the object deformations themselves. Since manipulation with deformable objects often involves actively changing the environment itself, we are often limited in the range of useful sensor information. For example, depth cameras may provide an accurate view of the undeformed obstacles ahead, but they cannot capture how the obstacles will deform when in contact. In contrast, touch or pressure sensors can provide a much better estimate of the deformation of the environment around the robot, but can only provide information on the current state. This is similar to the problem that occurs with operating in the presence of moving obstacles, where an accurate model of the entire environment is not available prior to execution.

In cases with rigid objects, this problem has been addressed in a variety of ways, such as [6], [7], [11], [12]; provided accurate information on the moving objects is available, the path can be deformed online to increase clearance. Alternatively, reactive control can be combined into the motion planning process a priori, such as [13], effectively anticipating changes during execution. In contrast, because of the aforementioned sensor limitations when using deformable objects, accurate information on the deformed environment may only be available at very short range or while in contact with obstacles, preventing any broader adaptation or optimization of the path. For robots operating in rigid environments with similar limits on sensor horizon, several planning and control schemes have been proposed [14], [15].

Our proposed reactive controller modifies a cost-space path on-the-fly during execution in an attempt to locally optimize the planned path to the real environment. While trajectory optimization techniques have already been applied to motion planning with deformable objects [2], our approach differs by performing the optimization incrementally online in a manner similar to path deformation [6], [7], [11], [12]. This approach combines the main strengths of motion planning, namely the completeness properties and global scope of the motion planner producing the initial path, with the reactive strengths of local control. Using the planner to provide a "guiding path" for local control addresses well-known problems of local control such as becoming stuck in local minima and failing to reach the goal. Complementing this, the local controller addresses one of the inherent limitations of most motion planning - the inability to react to unexpected changes or differences in the environment without expensive re-planning.

II. Reactive Controller

Our controller, initially proposed in [16], also referred to as the "Gradient Rejection Controller" (GRC), uses the local cost-space gradient to adapt the path during execution. This control strategy allows for local corrections to the path without compromising the global topology of the path, which could result in the controller becoming stuck in a dead end.

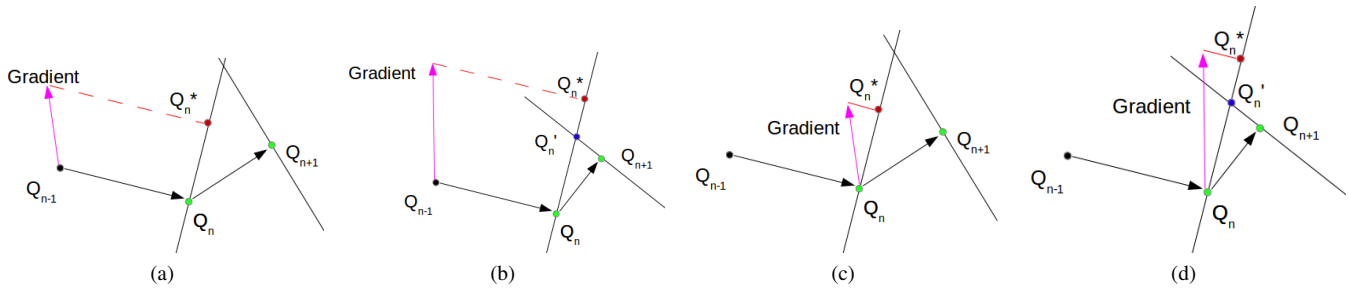


Fig. 1: Illustration of the Gradient Rejection Controller used with zero-step lookahead (a, b) and one-step lookahead (c, d). (a, c) Using the gradient, the controller computes a corrected version of Q_n, Q_n* that falls on the correction hyperplane. (b, d) Back-projection of constraints from successive states of the path: an initial corrected Q_n* violating the constraints for Q_{n+1} is brought back to the intersection of the correction hyperplanes at Q'_n.

Instead of a simple combination of the gradient and path which could result in undesirable behavior such as reversing or becoming stuck, the gradient is *rejected* onto the path so that only the components of the gradient orthogonal to the path are used to compute the correction. This combination is shown in Figures 1a and 1c, where the cost-space gradient is rejected onto the vector $\overrightarrow{Q_{n-1}Q_n}$ to produce the vector $\overrightarrow{Q_{n-1}Q_n*}$ that defines the corrected state Q_n*.

As a result of the rejection operation, all corrected Q_n* fall onto a hyperplane orthogonal to the vector $\overrightarrow{Q_nQ_{n-1}}$, which we refer to as the "correction hyperplane". To prevent corrected Q_n* from deviating too far from the original path, two constraints are applied to bound the GRC. The first bound limits the euclidean distance between Q_n and Q_n* to a user-defined scalar multiple α of the euclidean distance between Q_{n-1} and Q_n.

The second bound ensures that corrections cannot result in invalid future corrections or reversing. In certain pathological cases, corrections obeying the first constraint may effectively "skip" the next state of the path – Q_n* may fall ahead of the correction hyperplane for $Q_{n+1}*$, which would require either that the controller reverse to follow the path or skip Q_{n+1} entirely. To prevent this type of correction, potential values of Q_n* are compared against the correction hyperplane for Q_{n+1}. As shown in Figures 1b and 1d, Q_n* values that violate this constraint are restricted so that they fall on the intersection of the correction hyperplanes for Q_n and Q_{n+1}. This ensures that $Q_{n+1}*$ remains viable (albeit potentially a negligible distance from Q_n*) for any path.

III. Gradient

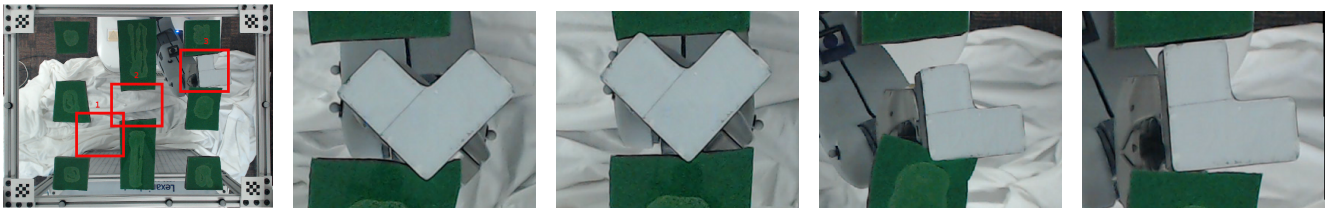
To compute the correction at each state, our controller requires a cost-space gradient. Ideally, this would be the actual gradient at the next state Q_n, since this gradient provides the best information about the neighborhood of Q_n. However, computing this gradient requires sensors with sufficient range, which may not be available. In many cases, such as a robot with touch sensors or whiskers, the sensors are only capable of providing information on the current state Q_{n-1}, which means we are only able to compute the gradient at Q_{n-1}. Thus, we can divide different controller modes based on the "lookahead" range offered by the onboard sensors. We use the term "zero-step lookahead" to refer to cases where we are only able to compute the gradient at Q_{n-1} as shown in Figure 1a, and the term "one-step lookahead" to describe cases where we have sufficient sensor information to compute the gradient at Q_n as shown in Figure 1c.

IV. Results

We have tested our controller using a test environment similar to that used in our previous work [1]. The test environment is constructed out of blocks of deformable plastic foam securely attached to a clear acrylic sheet to prevent the blocks from shifting or rotating during deformation. This deformable test environment, shown in Figure 2a, allows for three degrees-of-freedom (two translation, one rotation) while also allowing the compression of the environment to be tracked online via camera. This tracking provides us with a "ground truth" value of the actual cost incurred during the execution of a path. As with our previous work, paths for the L-shaped "robot" (shown in Figure 2) were executed using a PR2 robot. For planning purposes, cost is assessed using the cost function introduced in our previous work [1]; during execution, cost is assessed by measuring the change in area (pixels) of the deformable objects visible to the tracking system.

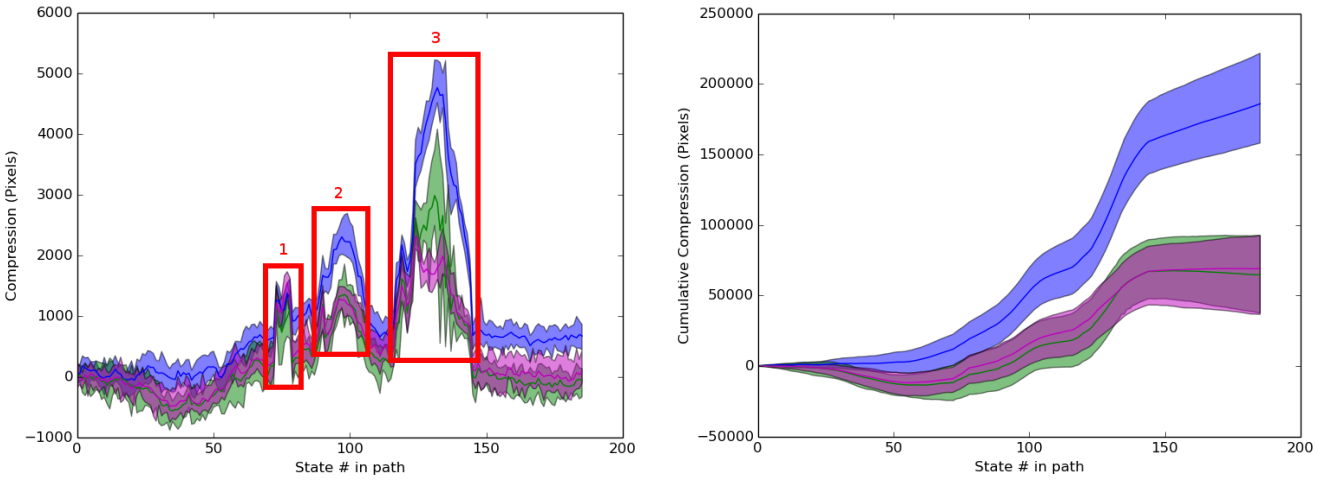
To simulate sensor error in the initial planning stage, the environment model used by the planer and the actual physical environment are distinctly different. To produce this difference, the passages through the environment were arbitrarily narrowed and offset, and the shapes of each foam block slightly altered. These differences were designed to result in significantly higher cost-as-executed for paths executed with open-loop control without changing the topology of the environment (thus ensuring that the planned paths remain feasible). Notably, these simulated errors create three distinct high-cost passages in the environment, as shown in Figure 2a. Each successive passage differs more from the environment model, and as a result, results in successively higher cost-as-executed.

Using an initial path produced using the A*-derived planner introduced in [1], the path was executed with three separate methods: First, using open-loop control to measure the "baseline" cost-as-executed of the path. Second, using



(a) (b) (c) (d) (e)

Fig. 2: (a) Test environment and L-shaped robot with high-cost passages marked. Execution in high-cost passages: (b) passage 2 with zero-step lookahead, (c) passage 2 with one-step lookahead, (d) passage 3 with zero-step lookahead, and (e) passage 3 with one-step lookahead. Note the high penetration in (b) and (d) caused by oscillation during execution.



(a) (b)

Fig. 3: Performance of our controller executing a path in the test environment, showing (a) the incremental cost at each state with the three high-cost passages highlighted, and (b) the cumulative cost of execution, averaged over 10 executions with each control mode. Open-loop execution with no added control is shown in blue, execution with zero-step lookahead in green, and execution with one-step lookahead in magenta. Shading indicates the range between minimum and maximum values for the 10 executions.

GRC with zero-step lookahead. Third, using GRC with one-step lookahead. The path was executed ten times with each method, for a total of 30 executions. For both zero- and one-step lookahead, cost-space gradients were computed using a signed distance field generated from the tracking camera. The incremental and cumulative costs measured during the execution of each mode are shown in Figure 3.

As shown in Figure 3b, using our reactive controller significantly reduces cost-as-executed using both zero- and one-step lookaheads. This reduction in cost-as-executed is particularly clear in Figure 3a, which shows the cost reductions for each of the three high-cost passages. This indicates that our controller is suitable for use both in cases with limited-range sensor data is (one-step lookahead) and in cases with only contact sensor data (zero-step lookahead).

While the cumulative cost-as-executed suggests that zero- and one-step lookahead result in similar performance, in practice there are significant differences in the quality of the execution. As illustrated in Figures 2b and 2d, using zero-step lookahead can result in large oscillations (similar to those encountered before [16]) that cause higher-than expected penetration and risk damage to the environment. In contrast, one-step lookahead largely eliminates these oscillations, as seen in Figures 2c and 2e. The effects of these oscillations are particularly pronounced in the third passage, and this can be seen in Figure 3a, where the one-step lookahead results in significantly reduced cost.

The significant oscillations encountered while using zero-step lookahead are the result of states where the current gradient provides insufficient or incorrect information on the neighborhood of the next state. Two specific cases of these oscillations are particularly notable: the first occurs when moving closely beside an object, the second occurs when moving down a narrow passage. When moving closely beside an object, the gradient at state Q_n provides sufficient information to push the robot away from the object. However, once the robot has been corrected away from the object at state Q_{n+1}, a gradient is no longer available, and the robot returns to the original path at state Q_{n+2}. This cycle continues until the path diverges from the obstacle. When moving through certain narrow passages, following the gradient at state Q_n leads to an opposing gradient at the next state Q_{n+1}, resulting in oscillation between the two sides of the passage. In both cases, the oscillations occur when the gradient at a *corrected* state becomes inconsistent with the gradient at the *uncorrected* next state.

Several approaches exist to reduce or eliminate these oscillations. Ideally, sufficient sensor information is available

to use one-step lookahead, which eliminates the problem by directly using the gradient at the uncorrected next state. However, if this information is not available (for example, pressure sensors), then past gradients (and the corrections applied to the path) may be used to damp the oscillations in a similar manner to the damping terms used in other controllers. In practice, however, using this damping may reduce oscillations at the expense of increased overall cost, since the corrections will no longer reflect the true gradient at the next state as accurately.

V. Discussion

In future work we intend to augment controller behavior by extending the lookahead window and using a combination of path optimization and re-planning. Combining a larger lookahead window with path optimization techniques such as [8], [9], [10] allows the controller to optimize the path for the entire window, rather than a single step at a time. In addition to better path optimization, larger lookahead windows allow early detection of unexpected high-cost regions which may then be completely avoided (if possible) by re-planning the path. Complementing further development of the controller, we are also developing a specialized test platform that will allow us to test our controller on tasks in a full three-dimensional environment.

References

[1] C. Phillips-Grafflin and D. Berenson, "A Representation Of Deformable Objects For Motion Planning With No Physical Simulation," in *ICRA*, 2014.

[2] B. Maris, D. Botturi, and P. Fiorini, "Trajectory planning with task constraints in densely filled environments," in *IROS*, Oct. 2010.

[3] F. Faure, B. Gilles, G. Bousquet, and D. K. Pai, "Sparse meshless models of complex deformable solids," *ACM Transactions on Graphics*, pp. 73–73, 2011.

[4] S. M. Lavalle, *Planning Algorithms*. Cambridge University Press, 2006.

[5] M. Pivtoraiko and A. Kelly, "Graduated Fidelity Motion Planning," in *International Symposium on Combinatorial Search*, 2011.

[6] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *ICRA*, 1993.

[7] O. Brock and O. Khatib, "Elastic Strips: A Framework for Motion Generation in Human Environments," 2002.

[8] M. Toussant, "Robot Trajecotry Optimization using Approximate Inference," in *International Conference on Machine Learning*, 2009.

[9] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient Optimization Techniques for Efficient Motion Planning," in *ICRA*, 2009.

[10] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic Trajectory Optimization for Motion Planning," in *ICRA*, 2011.

[11] F. Lamiraux, D. Bonnafous, and O. Lefebvre, "Reactive path deformation for nonholonomic mobile robots," 2004.

[12] V. Delsart and T. Fraichard, "Navigating dynamic environments using trajectory deformation," in *IROS*, 2008.

[13] Y. Yang and O. Brock, "Elastic roadmaps – motion generation for autonomous mobile manipulation," 2010.

[14] T. Schouwenaars, J. How, and E. Feron, "Receding horizon path planning with implicit safety guarantees," in *American Control Conference*, 2004.

[15] F. Andert and F. Adolf, "Online world modeling and path planning for an unmanned helicopter," 2009.

[16] C. Phillips-Grafflin and D. Berenson, "Path Planning and Execution For Deformable Objects Using a Voxel-Based Representation," in *European Workshop on Deformable Object Manipulation*, 2014.