Constraint-Based Motion Planning of Deformable Robots*

Russell Gayle **Ming C. Lin** **Dinesh Manocha**
Department of Computer Science
University of North Carolina at Chapel Hill
{rgayle,lin,dm}@cs.unc.edu

***Abstract*— We present a novel algorithm for motion planning of a deformable robot in a static environment. Given the initial and final configuration of the robot, our algorithm computes an approximate path using the probabilistic roadmap method. We use "constraint-based planning" to simulate robot deformation and make appropriate path adjustments and corrections to compute a collision-free path. Our algorithm takes into account geometric constraints like non-penetration and physical constraints like volume preservation. We highlight the performance of our planner on different scenarios of varying complexity.**

***Index Terms*— motion planning, deformable models, physical constraints, probabilistic roadmap algorithms**

I. Introduction

Motion planning is a classical problem in robotics. The basic problem is defined as follows: Given a robot and an environment with a set of obstacles, find a collision-free path from an initial configuration to a final configuration. This problem has been well studied for more than three decades. However, most of the existing work has focused on robots that are either rigid bodies or articulated models. There is relatively little research on motion planning of a deformable robot. Flexible robots are becoming increasingly important in industrial and medical applications. For instance, motion planning can be used for cable placement in large factory plants and buildings, wire routing for nano-, micro-, to mega-scale electronic and mechanical structures, surgical procedure planning and simulation, etc.

Some existing planning algorithms compute a path by exploring the space of all valid configurations of a robot. However, when the robot is flexible or can easily change its shape, the dimensionality of its configuration space can become extremely large. Even a simple discretized deformable object can have hundreds of degrees of freedom. As a result, the algorithms for rigid robots are not directly applicable to deformable robots.

Main Results: One of our major goals is to investigate and develop physically-based motion planning methods for general deformable robots. More specifically, we address the following modified motion planning problem: Given a robot which deforms in a physically realistic manner and an environment with many static obstacles, find a path from an initial configuration to a final configuration. In addition, we would like to design a planner to be as efficient as possible.

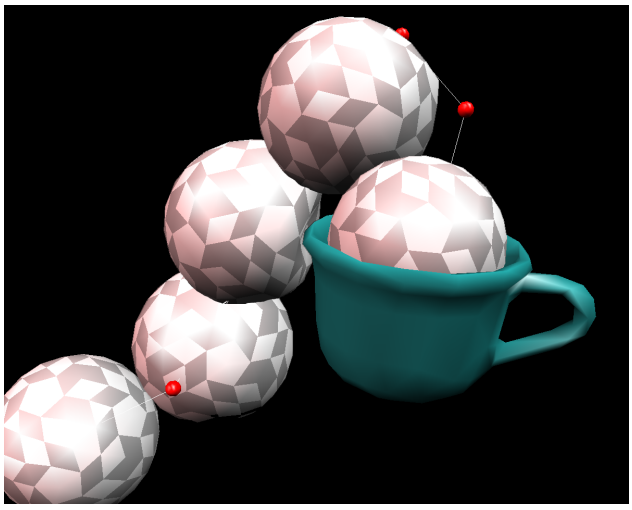

Fig. 1. *The Ball in Cup Scene: The goal is to plan a path to move the ball into the cup. Since the cup's rim has about the same diameter as the ball, a deformation on the ball simplifies the path planning. The image, generated by our planner, shows the path of the deforming ball as it moves toward the center of the cup.*

Our approach is built upon *constraint-based planning*. We use the probabilistic roadmap (PRM) planner for a point robot to generate an initial trajectory and incorporate physical constraints from model deformation to control the motion of the robot. The initial trajectory may have collisions with the obstacle. We use constraint-based dynamics to simulate robot deformation and make appropriate path corrections to compute a collision-free path.

It is possible to substitute any other roadmap generation algorithm to replace PRM in our framework. Currently, we use non-penetration and volume preservation constraints for the deformable robot, so that the robot will deform as it comes in close contact with the obstacles in a physically plausible manner. In order to achieve interactive performance, we relax the strict global volume preservation constraint by implementing a local method that sets a threshold on the amount of deformation (e.g. elongation or compression) and on internal pressure fluctuation. We demonstrate the performance of our planner on different scenarios of varying complexity.

* This project is supported in part by ARO, AMSO, DARPA, NSF, ONR/VIRTE, Intel Corporatin, and a DOE Fellowship.

Organization: The rest of the paper is organized as follows. Section 2 discusses related work. We give an overview of our approach in Section 3. Section 4 presents the planner in more detail and Section 5 describes the algorithm for modeling robot deformation. In section 6, we discuss various implementation issues and highlight some experimental results.

II. Related Work

In this section, we give a brief overview of prior work in randomized motion planning algorithms and deformable models.

A. Randomized Motion Planning

The motion planning problem has been well studied in robotics and computational geometry for more than three decades [1]. The complexity of exact or complete algorithms is exponential in the number of degrees of freedom of the robot. As a result, most practical planners are based on spatial subdivision of workspace or configuration space into cells [2], potential field methods [3] or randomized sampling [4]. In particular, the probabilistic roadmap planners (PRMs) have been successful in solving many difficult instances of the motion planning problem. A PRM planner generates samples at random in configuration space, attempting to connect each sample by a simple C-space path to one of the sample already found. Over time, the graph thus produced tends to represent the connectivity of the C-space reasonably well, and a query can be rapidly performed by linking the search points to the graph and then searching the graph.

Many algorithms have been proposed to improve the performance of PRM planners in terms of developing better sampling strategies and handling narrow passages [5], [6], [7], [8]. Most of these approaches are limited to rigid or articulated robots. Over the last few years, some algorithms for motion planning of deformable models have been proposed. Holleman et al. [9] and Lamiraux et al. [10] presented a probabilistic planner capable of finding paths for a flexible surface patch by modeling the patch as a low degree Bèzier patch. Anshelevich et al. [11] presented a path planning algorithm for simple volumes such as pipes and cables by using a mass-spring representation. Bayazit et al. [12] described a two-stage approach that initially computes an approximate path and refines the path by applying free-form deformation to the robot.

B. Deformable Models

Deformable models have been widely used in computer graphics, computer vision, medical imaging, physically-based modeling and related areas [13], [14]. They have been used for shape editing and analysis, computer animation, object reconstruction, image segmentation, training systems and simulation. Some of the simplest deformable models do not take into account physical properties and are based on functional deformation [15] and free-form deformation [16].

The simplest physically-based deformable models are based on a mass-spring system, where an object is modeled as a collection of point masses connected by springs in a lattice-like structure. The spring forces may be linear or non-linear and mass-spring models are widely used in computer animation. In practice, mass-spring systems are easy to construct and can be simulated at interactive rates on current commodity hardware. However, the choice of spring constants used in the simulation can become a major issue. Moreover, the underlying discrete model can become a poor approximation of the actual physics. More accurate physical models treat deformable objects as a continuum, i.e. solid objects with mass and energy distributed throughout and the continuum models are derived from equations of continuum mechanics. One of the most commonly used continuous model is finite element methods (FEM). The object is decomposed into elements joined at discrete node points and a function that solves the equilibrium equation is computed for each element. The computational requirements of FEM can be high and it can be difficult to use them for real-time applications. Other continuum models include *Snakes* [17] and discretized deformation energy models [13].

III. Overview

In this section, we introduce the notation used in the rest of the paper, and give an overview of our constraint-based planning framework.

A. Notation and Definitions

The deformable robot, R is represented as a time dependent set of masses, m_i, connected by edges, e_j, $\forall i, 1 \leq i \leq N$ and $\forall j, 1 \leq j \leq M$, where N represents the number of nodes and M is the number of edges describing the robot R. Each mass has an associated state vector $s_i = (x_i, v_i)$, representing the current position and velocity of each mass. The node points $X(t) = [x_1(t), x_2(t), \ldots, x_N(t)]$ collectively represents the robot's shape and position at time t.

Using this representation as a discrete form of FEM, we can represent the state of the robot as $S(t) = [s_1(t), s_2(t), \ldots, s_N(t)]$ at time t. We assume that the connectivity of the robot does not change during the simulation, so each e_i will remain the same throughout the planning process. Each edge has an associated stress value and threshold, $stress_i$ and δ_i, respectively. We use these values to implement volume preservation for the deformable robot.

We also define a deformation energy function $E(X)$ of the robot. $E(X)$ simulates the potential energy of elastic solids, and is a measure of the amount of deformation. The robot can be interactively deformed as contacts occur with the obstacles in the environment. This deformation may

change the volume $V(X)$, as well as the energy $E(X)$. To simulate physically plausible deformation, we need to find a new configuration of X, that preserves the total volume of the deformed robot, while minimizing the the total energy of the system.

We assume that the set of obstacles in the environment are rigid and they are represented as $O = \{o_1, o_2, \ldots\}$ in the workspace, W. Our roadmap G consists of a set of milestones, $N = \{n_1, ..., n_l\}$, and links, $L = \{l_1, ..., l_k\}$, which form a graph-like structure (i.e. $G = \{N, L\}$). A path, P along this roadmap is a sequence of valid milestones that are connected by valid links.

Problem Formulation: Given these definitions, we restate our problem as: Find a sequential set of robot configurations $X(t_1), .., X(t_f)$ such that no $X(t_i)$ intersects any obstacle in O and $X(t_i)$ satisfies the non-penetration and volume preservation constraints, where $X(t_1)$ is the initial configuration of the robot, and $X(t_f)$ is the final configuration.

B. Constrained-Based Planning

Garber and Lin proposed to reformulate the motion planning problem as simulating a constrained dynamical system, called "constraint-based motion planning" [18]. This formulation is due to the close resemblance between motion planning and the boundary value problem, where the boundary conditions can be mapped to the initial and goal configurations of the robot as a dynamical system. The planning algorithm is used to compute the intermediate states that link the two and satisfy the constraints imposed on the dynamical system.

The key differences between this approach and existing geometric techniques is that motion planning is no longer treated as a purely geometric problem. With this framework, we can automatically incorporate the mechanical and physical properties of the robots and obstacles, in addition to their geometric description. This framework is based on constrained dynamics [19] used for physically-based modeling, where constraints are enforced by virtual forces imposed on the system. By combining these constraint forces into a simulation framework, the constrained dynamical system can guide the robot to follow these constraints, such as volume preservation constraints, at each time step, which can lead to a path toward the final configuration. An overview of the framework and the computation of constraint forces is shown in Fig. 2. We refer the readers to [18] for more detail.

C. Extension to Deformable Models

In the earlier work on using constraint-based motion planning for articulated robots in a dynamic environment, each robot is treated as a rigid body, or a collection of rigid bodies, and is moved subject to different types of constraint forces [18]. In this paper, we introduce constraints designed to guide the *deformable* robot through the environment

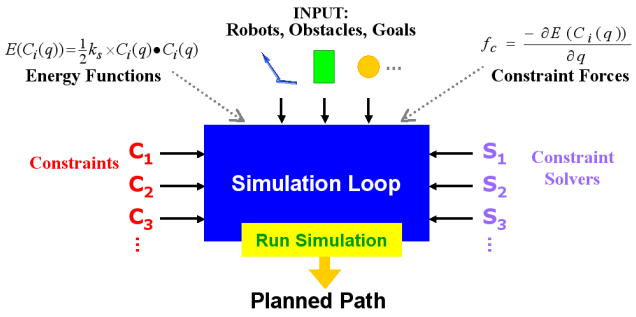

Fig. 2. *Constraint-Based Motion Planning Framework: C_i are constraints, S_i are different types of constraint solvers. Each virtual force f_c is introduced to the simulation system through a energy function $E(C_i(q))$ associated with the constraint $C_i(q)$, where q represents a configuration.*

to the desired goal configuration. Using global analysis from any roadmap algorithm (e.g. PRM) to compute a possible path for a point robot, we define both geometric and physical constraints that move the deformable robot to avoid both static and moving obstacles, and also follow an estimated path to the goal. This approach transforms a motion planning problem into one of defining suitable physical constraints and then simulating the deformable body dynamics of the scene with each constraint acting as a virtual force on the robot.

Guiding Path: Our framework allows the use of any estimated path computed by either quick global analysis of the environment, such random sampling [4], [8], [20] or user guided input [21].

Constraints: To achieve the desired results without robustness problems, we further classify the constraints into soft and hard constraints and treat them differently. *Hard Constraints* are those that absolutely must be satisfied at every time step of the simulation, such as non-penetration and volume preservation constraints. A wide range of constraint solvers exists to find the solutions for hard constraints [18]. *Soft Constraints* serve as guides to encourage or influence the objects in the scene to behave in certain ways, such as moving along a certain path. They are simulated using penalty forces. The estimated path will be used as a soft constraint to guide the planning of the deformable robot or a rigid robot moving through a deformable environment. Our proposed algorithm will compute the new path by taking into consideration the interaction of the flexible robot with the obstacles in the workspace.

Proximity Queries: We can also take advantage of methods proposed in [22], [23] that uses graphics hardware to quickly perform proximity queries on the workspace or to provide dynamically updated discretized distance fields for obstacle avoidance involving deformable models.

Discretization of Continuum: As for modeling the deformation of the robot or the environment, there exists many possible approaches. For example, these may include *physically-based* free-form deformation (FFD), mass-spring systems, boundary element methods (BEM), and

finite element methods (FEM). For coding simplicity and runtime performance, we chose an implementation based on the spring-mass system to validate our basic approach. However, we can easily incorporate any discretization technique with this algorithmic framework.

IV. Algorithm

Our algorithm consists of two phases: (a) off-line roadmap generation and (b) a runtime path query phase based constrained dynamic simulation. The planner follows the basic steps below:

1) **Roadmap Generation:** Create a roadmap, $G = \{M, L\}$ of the environment with a point robot
2) **Path Estimation:** Find an initial path P in G from the initial to goal configuration.
3) **Path Query:** While the robot is not in its final configuration
   a) Take a step in the simulation, and set $t = t + h$
   b) If constraints are not satisfied as a result of this simulation step

      Perform a roadmap pruning step and correct the robot's motion

A. Roadmap Generation

As described earlier, the goal of this phase is to create the set M of milestones and set L of links. For the sake of simplicity and efficiency, we treat the robot as a point robot. This strategy for simplifying the problem makes the generation of each milestone simple and quick.

Like a standard PRM, we can set milestones to be random samples of the configuration space. For each m_i, we first determine if it is a valid milestone by performing a very quick check to determine if it is inside any obstacle. Then, we can find either the nearest k milestones or all milestones that are within some search radius of m_i and place them into a set, N, of neighboring milestones. For each milestone m_j in N, we determine if there exists a straight-line path from m_i to m_j that does not intersect any obstacle in O. If such a path exists, we add a link, l_k, from m_i to m_j to L. We also introduce extra milestones for the initial and final configurations, perform link tests on these, and add them to the roadmap. By repeating these steps for some user-defined number of milestones, we can quickly generate a roadmap for the point robot.

It is likely that the initial roadmap generated could cause the resting state of the robot to be intersecting with the environment. If the robot must reach one of thees milestones during the simulation, it is possible that the robot will have to deform to reach the goal configuration subject to the physical constraint. In cases when it cannot, the edges linking to this milestone will be pruned away and the planner can look for another path.

One potential drawback to this method is that the resulting path may not be a minimal energy path. This can easily be accomplished by computing the minimal distance from an obstacle to the link and weighting the links accordingly. Since the largest deformations are most likely to occur near an obstacle, by staying farther away from obstacles, we can reduce the energy required to perform all deformations along a path. The main benefit of computing the roadmap in this manner is speed and simplicity.

B. Path Query Phase

The online path query relies heavily on the constraint-based dynamics simulation of the deformable robot. We start by first generating an initial path P directly from the roadmap, G, by using Dijkstra's shortest path algorithm. Once an estimated path is computed, we begin our simulation loop with the robot at its initial configuration.

Each object is represented by some state which is typically a tuple of attributes like position or velocity. For rigid objects, the state can also include angular orientation and angular velocity. By using the state representations, we solve motion equations using various numerical methods to determine the next step. Typically, these motion equations either include or can be augmented by adding various forces. The forces can represent constraints, collision impulses, or other external forces like gravity. By incrementally taking simulation steps and satisfying proper geometric, physical and mechanical constraints, we can control and influence the motion of the object as the simulation steps forward toward the goal configuration.

The robot itself has some built in constraints, including a hard non-penetration constraint with obstacles and a hard volume preservation constraint which, however, is loosely approximated in our current implementation (see the next section). It also maintains a soft path-following constraint and energy minimization constraint that essentially tries to keep the robot at its equilibrium state.

Given the set of constraints which are treated as virtual forces in the system, each simulation step is then solved using numerical methods to advance toward the goal configuration. The following fragment shows a single simulation step or a path query step for our planner.

For each simulation or path query step:

1) Accumulate all external forces, F_e (e.g. gravity).
2) Deform the robot R (as described in the next section)
3) Compute constraint forces, F_c.
4) Given all forces, solve the Ordinary Different Equations describing the dynamical system
5) If constraints are not satisfied,
   a) Set the last valid milestone as the next destination
   b) Remove the current edge in the roadmap
   c) Find a new path from the last valid milestone to the goal
   d) Compute new constraint forces and solve the ODE, using last state of the robot R and F_e
6) Set the next robot state to be the current ODE solution and set $t = t + h$

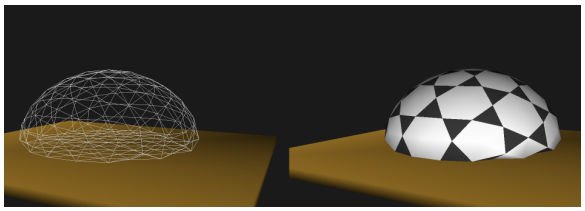

Fig. 3. *A simple deformable robot in resting contact with a plane. The left is the wireframed (mass-spring) lattice of the image on the right.*

This fragment is repeated until we reach the final configuration. This simulation framework generates robot motions that look physically plausible at a relatively fast rate. The processing of the deformation is an important step and also one of the most computationally expensive steps. We will describe it next.

V. Robot Deformation

We represent the deformable robot as a mass-spring system. In principle, a mass-spring system is a set of point-masses that are connected in a lattice by an underlying spring structure (see Fig. 2). As in our deformable robot state that is associated with each mass, m_i, is the state $s_i = (x_i, v_i)$. By aggregating the x_i's into a vector X, we get a $3N$-dimensional system whose motion can be described by the following second-order differential equation:

$$M\ddot{x} + C\dot{x} + Kx = F,$$

where M and C are diagonal matrices, and K is a banded matrix. The ith diagonal element of M is simply the value of m_i and similarly the ith diagonal element on C is a dampening constant for the mass m_i, which is usually assigned by the user. K is banded since it must represent spring forces which are functions of distance between two masses. F is a $3N$-dimensional vector which includes the external forces acting upon each mass.

A. Deformation Energy Function

The elastic deformation energy measures the amount of deformation. The deformation is essentially local stretches in various directions. If the motion is simply a rigid transformation, meaning that it preserves the distances between all particles (no stretches), the energy must be zero. The local deformation is governed by the deformation gradient. The right Cauchy-Green tensor $C = F^T F$ measures the length of an elementary vector after deformation, and is insensitive to rigid body transformations.

Let $E(X)$ be the energy density function of an elastic solid undergoing deformation. The total energy is obtained by integrating $E(X)$ over the entire volume of the solid. E can be expressed as a function of the right Cauchy-Green tensor C for elastic materials unless zero E is allowed for a non-rigid transformation (spurious zero-energy mode).

In fact, the simplest law uses a quadratic function of the right Cauchy-Green tensor C [LeTal94]. Since F and C are a linear and quadratic functions of X respectively, E is at least a quartic function of X. We have chosen the energy function of a spring network that connects the neighboring nodes. The energy function can be written as:

$$E_s(X) = \sum_j \frac{k}{2}(||d_j|| - L_j)^2$$

where j is the index of a spring and L_j is the natural length of the spring and d_j is the distance between two nodes x_i and x_k connected by the spring.

Basically we would like to find X by solving

$$min \;\; E(x) \text{ subject to } \nabla V(X) \leq \epsilon.$$

Here we relax the hard volume preservation constraint by allowing the change in volume to be less than a given tolerance ϵ. This problem can be solved by using a global constrained minimization technique. Our current implementation uses a local method that checks whether the internal pressure fluctuation is bounded and the deformation at each edge e_i does not exceed certain pre-defined tolerance (i.e. $stress_i < \delta_i$) to achieve the same effects.

B. Volume Preservation

One limitation of the mass-spring system is the difficulty to represent objects with sharp edges. A possible solution would be to add extra angular springs. In order to achieve a similar effect and satisfy the volume preservation constraint, we rely on the ideal gas law. By definition, the force due to pressure on a surface, $\vec{F_p}$, has the magnitude:

$$F_p = PA$$

where P is the total pressure inside the object, A is the surface area and $\vec{F_p}$ has the same vector direction as the face normal $\hat{n}$ pointing away from the surface. To quickly approximate the pressure inside the object, we use the ideal gas law:

$$PV = nR_gT$$

where V is the volume of the robot, n is the number of moles of gas, R_g is the universal gas constant and T is the temperature of the gas. For any given situation, we can set nR_gT to be a user-defined constant since it should not vary within the simulation. The remaining unknown in this equation is the volume of the robot.

C. Volume Computation

To compute the volume, we break the space into triangular prisms. Each prism is formed as the swept volume from the triangle to its projection onto the $X - Y$ plane. To find the volume of the prism, we multiply the average height by the area of the projected triangle. Or:

$$V = \sum A_{base} h,$$

where

$$A_{base} = \frac{1}{2}((p_2 - p_1) \times (p_3 - p_1))_z$$

and

$$h = \frac{(p_1 + p_2 + p_3)}{3}.$$

It is important to note that PV is a constant. Thus, if P does not vary much, then V would not change much either.

D. Deformation Step

Once we solve for P, we can compute F_p for a triangle. By computing this quantity for each mass, we obtain a simple simulation of pressure in a soft object. The following code fragment shows the loop involved in handling the deformations:

1) Perform collision detection
2) Handle any collisions to enforce non-penetration constraint
3) Accumulate the spring forces, F_s
4) Compute the volume, V of the object
5) Set $P = nR_gT/V$
6) For each face f on the object's geometry
   a) Set $F_p = PA$
   b) For each vertex v of f

      Find the pressure forces on v by adding F_p divided by the number of faces incidental to v.

Since a collision can result in penetration, it is important to check for contacts during the simulation. For robustness of the implementation, we consider that the robot is colliding with an obstacle, if the robot is within tolerance to an obstacle, or when the robot has actually intersected the obstacle.

To accelerate the collision detection, we impose a uniform grid around the workspace. The size of grid cell is set to be the size of the longest edge of the deformable object in its minimal energy state. With this setting, in most cases we only have to check eight grid cells per step. Then, we can perform a standard collision check between the robot and the obstacles in these cells. Collision response is then performed based on if the collision results from edge-edge intersection or point-face intersection.

Taking discrete time steps can occasionally lead to situations where the object penetrates or intersects one of the obstacles, if it is moving fast enough. We can handle this situation by performing backtracking in time and then applying the collision resolution step.

One of the features of this simulation-based approach is that it maintains several of the hard constraints automatically. In particular, the non-penetration constraint is performed by collision detection and contact resolution.

VI. Implementation and Results

We have implemented and tested this algorithm on a laptop with a 1.5 GHz Pentium-M processor, 512 MB of main memory, and a 64 MB ATI Radeon 9000 Mobility graphics card. To test the effectiveness of our algorithm,

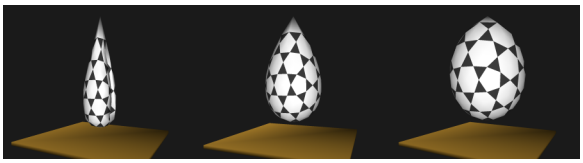

Fig. 4. *Soft-body robots with varying internal pressures. The leftmost robot has a relatively low pressure, while the rightmost has a fairly high pressure. The middle image has a pressure value in between the other two.*

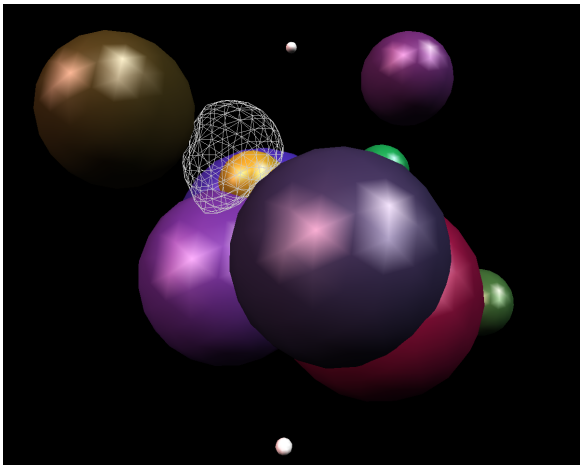

Fig. 5. *The results of a simple test environment. The robot is shown in wireframe, and is deforming between some of the larger spheres. Each of the larger spheres are stationary, rigid obstacles.*

we applied our planner to various scenarios that evaluate different aspects of the planner. They are as follows:

- **Ball In Cup** - The simplest of the scenarios, this environment has a spherical, deformable robot which must find its way into a cup. The cup's rim has been adjusted so that the ball will barely be able to fit in, and will thus have to deform to do so. (See Fig. 1)
- **Many Spheres** - This is primarily a test on how varying degrees of geometric complexity affects the performance of the planner. The environment is composed of a collection of spheres of varying positions and radii. A spherical, deformable sphere must move through these spheres to reach a goal in the opposite corner of the space. (See Fig. 5)
- **Walls with Holes** - Based on one of the Parasol Motion Planning benchmarks [25], this scene has a sequence of four walls, each of which has a small hole. In order to reach its goal, the robot must make its way through each hole. We modified the model to have a wide, flexible cylindrical robot that either cannot, or barely can, fit through the holes. It is wide enough that finding a collision free path would be very difficult, if not impossible, if the robot was rigid. The goal of this scenario was to fit a deformable robot through a small space. (See Fig. 6)
- **Tunnel** - This is also based on a Parasol Motion Planning benchmark [25]. The scene contains a small

Scenario	Obstacle (tris)	Robot (tris)	Path Est. time (sec)	Total Sim. time (sec)	Average Step time (sec)
Ball In Cup	500	320	1	41.5	0.015
Sphere World	3200	320	1	333.16	0.077
Holes In Walls	216	720	48	605.958	0.037
Tunnel	72	720	575	833.24	0.068

TABLE I

This table highlights the performance of our planner running on a laptop with 1.5GHz Pentium-4 processor. We highlight the geometric complexity of the environment in term of the number of triangles. We also report the high-level path generation time and simulation time. The last column reports the average time taken per simulation step.

tunnel with two bends in it. Like in the Walls benchmark, we augment the model by having a robot that is both long and wide. These constraints would make it impossible for a rigid robot to find a collision-free path. This scenario tested the planner's ability to force the robot to bend and deform around sharp corners. (See Fig. 7)

Results from each of these tests can be seen in their associated figures and are also summarized in Table I. The first two columns of the table describe the geometric complexity of the scene, for both the robot and the obstacles. The next columns give performance results of the planner by highlighting the time spent in various stages. One column gives the time the planner spent in high-level path generation, and the next gives the total time the planner spent in simulation steps. Lastly, we give an average step time for each simulation step.

Taking differences of systems into account, the average step time reported here is comparable to the time to process purely geometric deformations (based on FFD) used in other planners for deformable objects [12]. However, our technique also takes into account the physical constraints.

As can be seen from the table, the bulk of the time is in the simulation steps. These steps include the force calculation, collision resolution, establishing constraints, and solving the resulting system. Our current implementation is not optimized and the performance of each of these stages can be further improved.

VII. Analysis and Future Work

In this paper, we have presented a new framework for motion planning of deformable robots. We use a constraint-based planning algorithm and impose non-penetration and volume preserving constraints along with energy minimization. Our approach is applicable to all robots that can be represented as closed objects. Our framework can use any roadmap generation algorithm, as well as any discretized deformation model. We have applied our algorithm to four different scenarios and the initial results are promising.

Our approach has some limitations. It currently does not have the ability to include differential constraints on the robot's motion. Its motion is determined entirely by the soft-constraint which moves it towards the goal and the response to other external forces and collisions. Moreover, our robot is restricted to a closed shape. Our planner uses local techniques to compute a trajectory and cannot guarantee a physically plausible motion and deformation for all cases.

There are many avenues for future work. In order to incorporate differential constraints, we would like to use RRT algorithm [20]. Also, with the recent advances in graphics hardware, graphics processing units (GPU) have extra features which could be very useful for accelerating the performance our planner. Moreover, GPU based intersection or visibility computations can be used to accelerate link queries and collision detection and the planner can be used in more complex environments [26].

Currently the coefficients used in our planner are estimated empirically. We would like to obtain better coefficients through physical measurements. We would like to further compare the results of our planner with real scenarios and evaluate its effectiveness. Finally, we would like to extend the planner to environments, where the obstacles are non-rigid or moving, as well as better modeling of friction, sliding and rolling contacts between deformable surfaces.

Acknowledgment

We would like to thank the anonymous reviewers for their helpful comments in improving the manuscript.

References

[1] J. Latombe, "Motion planning: A journey of robots, molecules, digital actors, and other artifacts," *International Journal of Robotics Research*, pp. 1119–1128, 1999.

[2] R. A. Brooks and T. Lozano-Pérez, "A subdivision algorithm in configuration space for findpath with rotation," *IEEE Trans. Syst*, vol. SMC-15, pp. 224–233, 1985.

[3] O. Khatib, "Real-time obstable avoidance for manipulators and mobile robots," *IJRR*, vol. 5, no. 1, pp. 90–98, 1986.

[4] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, pp. 12(4):566–580, 1996.

[5] D. Hsu, L. Kavraki, J. Latombe, R. Motwani, and S. Sorkin, "On finding narrow passages with probabilistic roadmap planners," *Proc. of 3rd Workshop on Algorithmic Foundations of Robotics*, pp. 25–32, 1998.

[6] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo, "Obprm: An obstacle-based prm for 3d workspaces," *Proceedings of WAFR98*, pp. 197–204, 1998.

[7] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "Motion planning for a rigid body using random networks on the medial axis of the free space," *Proc. of the 15th Annual ACM Symposium on Computational Geometry (SoCG'99)*, pp. 173–180, 1999.

[8] M. Foskey, M. Garber, M. Lin, and D. Manocha, "A voronoi-based hybrid planner," *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2001.

[9] C. Holleman, L. Kavraki, and J. Warren, "Planning paths for a flexible surface patch," *IEEE Int. Conf. Robot. Autom. (ICRA)*, 1998.

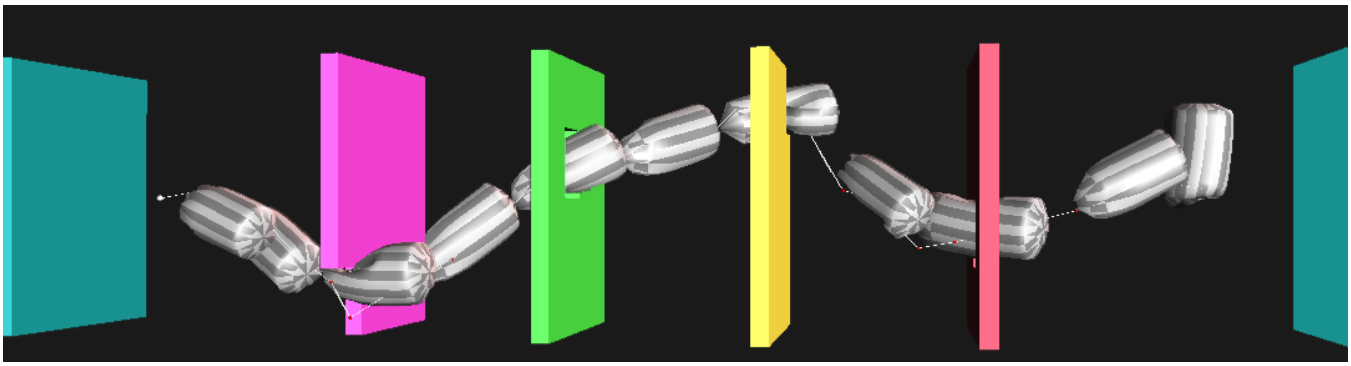

Fig. 6. *The Wall with Holes test. This scenario shows a sequence of four walls with holes in them. To reach the goal, the robot travels through each hole. This figure highlights the various states of the deformable robot while travelling along the displayed path.*

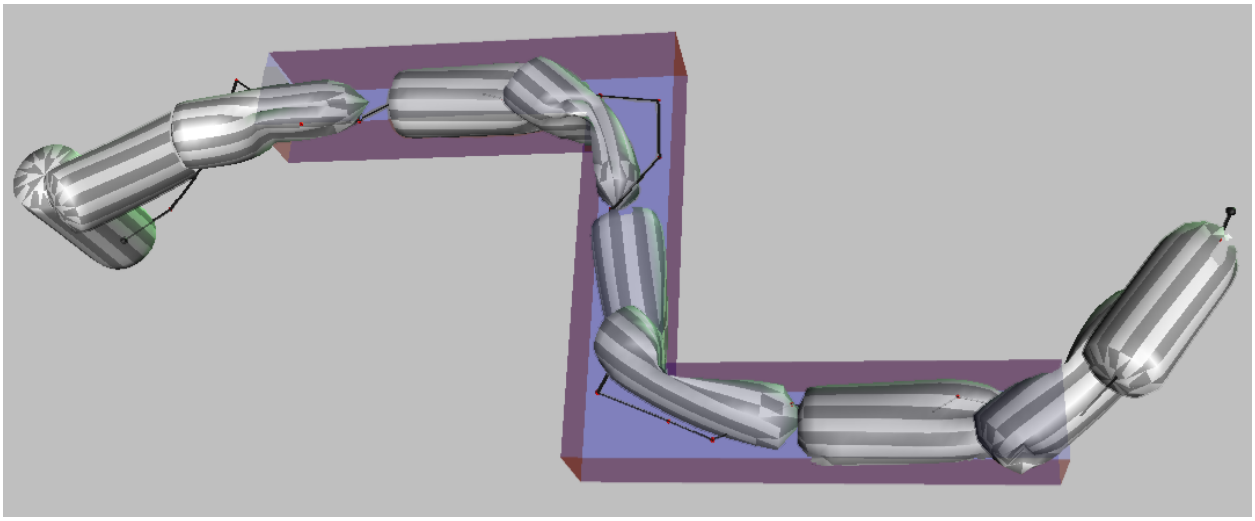

Fig. 7. *The Tunnel Environment. This environment is a simple tunnel which the robot must travel through in order to reach the goal. In this image, the striped cylinders represent the deformable robot at various states in its path through the tunnel. Note that a collision-free would not be possible if this robot was not deformable*

[10] F. Lamiraux and L. Kavraki, "Path planning for elastic objects under manipulation constraints," *International Journal of Robotics Research*, vol. 20, no. 3, pp. 188–208, 2001.

[11] E. Anshelevich, S. Owens, F. Lamiraux, and L. Kavraki, "Deformable volumes in path planning applications," *IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 2290–2295, 2000.

[12] O. B. Bayazit, H. Lien, and N. Amato, "Probabilistic roadmap motion planning for deformable objects," *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2002.

[13] S. F. Gibson and B. Mirtich, "A survey of deformable modeling in computer graphics," Mitsubishi Electric Research Laboratory, Tech. Rep. Technical Report, 1997.

[14] A. Singh, D. Goldgof, and D. Terzopoulos, Eds., *Deformable Models in Medical Image Analysis*. IEEE Press, 1998.

[15] A. H. Barr, "Global and local deformations of solid primitives," in *Computer Graphics (SIGGRAPH '84 Proceedings)*, vol. 18, July 1984, pp. 21–30.

[16] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," in *Computer Graphics (SIGGRAPH '86 Proceedings)*, D. C. Evans and R. J. Athay, Eds., vol. 20, no. 4, Aug. 1986, pp. 151–160.

[17] M. Kass, A. Witkin, and D. Terzopoulus, "Snakes: active contour models," *International J. of Computer Vision*, vol. 1, no. 4, pp. 321–332, 1987.

[18] M. Garber and M. Lin, "Constraint-based motion planning using voronoi diagrams," *Proc. Fifth International Workshop on Algorithmic Foundations of Robotics*, 2002.

[19] A. Witkin and D. Baraff, *Physically Based Modeling: Principles and Practice*. ACM Press, 1997, course Notes of ACM SIGGRAPH.

[20] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," *Robotics: The Algorithmic Perspective (Proc. of the 4th Int'l Workshop on the Algorithmic Foundations of Robotics*, 2000.

[21] O. B. Bayazit, G. Song, and N. Amato, "Enhancing randomized motion planners: Exploring with haptic hints," *Proceedings of ICRA*, pp. 529–536, 2000.

[22] K. E. Hoff, III, T. Culver, J. Keyser, M. Lin, and D. Manocha, "Fast computation of generalized Voronoi diagrams using graphics hardware," in *Computer Graphics Annual Conference Series (SIGGRAPH '99)*, 1999, pp. 277–286.

[23] N. Govindaraju, S. Redon, M. Lin, and D. Manocha, "CULLIDE: Interactive collision detection between complex models in large environments using graphics hardware," *Proc. of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pp. 25–32, 2003.

[24] M. Matyka and M. Ollila, "A pressure model for soft body simulation," *Proc. of SIGRAD*, pp. 325–333, 2003.

[25] P. R. Group, "Motion planning benchmarks," 2003, http://parasol.tamu.edu/groups/amatogroup/benchmarks/mp.

[26] R. Gayle, P. Segars, M. Lin, and D. Manocha, "Path planning for deformable robots in complex environments," University of North Carolina-Chapel Hill, Tech. Rep., 2005.