# Constraint-based Motion Synthesis for Deformable Models

William Moss Ming C. Lin
Dinesh Manocha
University of North Carolina at Chapel Hill
{wmoss,lin,dm}@cs.unc.edu
http://gamma.cs.unc.edu/

## Abstract

*We present a fast goal-directed motion synthesis technique that integrates sample-based planning methods with constraint-based dynamics simulation using a finite element formulation to generate collision-free paths for deformable models. Our method allows the user to quickly specify various constraints, including a desired trajectory as a sparse sequence of waypoints, and it automatically computes a physically plausible path that satisfies geometric and physical constraints. We demonstrate the performance of our algorithm by computing animated realistic motion of deformable characters and simulation of a medical procedure.*

**Keywords:** goal-directed motion, finite element, physical based modeling

## 1 Introduction

Physics-based simulations are increasingly used to create complex visual effects, including smoke, water, cloth, hair and articulated and deformable bodies for animation, gaming, training, education, design and prototyping, medical simulation, and many other applications. One of the challenges in motion synthesis of complex dynamic systems is control of the simulation results, as users would often like to generate certain types of motion and outcomes to exaggerate an effect, convey an emotion, or test a hypothesis. Although some recent success has been seen in control of rigid body dynamics [1], simulation of multi-body systems [2], complex particle systems [3], and fluid animation [4], by contrast relatively little progress has been made in goal-directed motion synthesis of deformable agents.

### *1.1 Our Approach*

We present an efficient method for generating physically-plausible, goal-directed motion for deformable models in complex, non-rigid environments. Our approach allows the animator to quickly specify the desired trajectory as a *sparse* sequence of waypoints through a scene. Given the set of waypoints, a deformable model and the environment our algorithm automatically computes a plausible path for the model through the environment, subject to various geometric and physical constraints. The geometric constraints may correspond to non-penetration, collision avoidance with nearby obstacles, and following desired trajectories to perform certain tasks (e.g. reaching a place or avoiding predators); whereas physical constraints may include maximum extent of allowable deformation, volume preservation, and energy minimization. The motion of the deformable model is simulated as it traverses the desired trajectories using a variant of sampling-based planning techniques in *contact space* [5], interacting with the environment and obstacles, and deforming as needed.

The main results of this work include:

- Synthesis of a finite element method (FEM) with constraint-based dynamics formulation [6] to achieve realistic physical behavior and plausible deformations of the deformable model, thus enabling physically plausible and stable simulation of large deformations;
- Introduction of a physics-based simulation framework with novel motion planning techniques based on new contact-space sampling to allow animators to specify trajectory constraints and incorporate

locomotion patterns (e.g. wiggling body motion);

- Formulation of appropriate physical and geometric constraints for deformable solids to properly model key characteristics of flexible bodies moving among either rigid or deformable obstacles.

We have implemented and applied this novel framework to both animation of deformable characters navigating through complex scenes with narrow passages and medical visualization of an automated catheter insertion procedure at nearly interactive rates. To the best of our knowledge, our approach offers the first real-time constraint-based, goal-directed motion synthesis technique for deformable agents using FEM.

### *1.2 Outline*

The rest of this paper is organized as follows. We briefly discuss related work in section 2. Section 3 presents our framework for automatic motion generation for deformable models with given constraints. We describe our approach to modeling the dynamics of deformable bodies with path planning constraints in section 4. Finally, we highlight the performance of our algorithm on multiple complex scenarios in section 5.

## 2 Related Work

There has been a significant amount of previous work on motion synthesis and control of rigid and articulated characters, and more recently on control of fluid motion. In contrast, our focus in this paper is on deformable models. In this section, we briefly review some of the previous work in these areas.

### *2.1 Deformable Models*

Modeling deformable bodies has been extensively studied in recent years. We refer the readers to [7] and [8] for a more comprehensive survey on modeling of non-rigid solids. Our work builds upon the seminal work on constraint-based dynamics [6] extending it by using an FEM simulation, instead of using only point masses and springs, to incorporate more realistic physical constraints and simulate complex interactions among heterogeneous models.

### *2.2 Motion Planning for Flexible Robots*

Several techniques have been proposed in robot motion planning to compute collision-free paths for deformable robots. These include sample-based planners for flexible surface patches that formulate the surface as a low degree Bèzier patch and use an approximate energy function to model deformation of the part [9, 10]. Such computations can be further accelerated by generating samples near the medial axis of the workspace [11]. Bayazit et al. [12] describe a two-stage approach that initially computes an approximate path and then refines the path by applying geometric-based free-form deformation to the robot. Path planning algorithms based on a mass-spring representation have been proposed for simple volumes such as pipes and cables [13] and complex environments [14, 15]. Path planning in fully deformable environments using an RRT is suggested by Rodriguez, et al. [16]. This method differs from other standard RRTs in that the roadmap is generated in force space, not C-space, however, this method does not run in real time and provides no bounds on the amount of deformation incurred by a robot while moving to the goal.

Augmenting sample-based techniques in robot motion planning, our work presents the first constraint-based motion synthesis method using FEM to automatically generate large-scale path planning through complex and possibly non-rigid environments with narrow passages.

## 3 Overview

The objective of this work is to provide the animator with sufficient control to create an animation that meets his or her artistic intentions while minimizing the efforts required to synthesize physically realistic motion subject to various constraints for a high degree-of-freedom deformable character. We allow the animator to indicate a desired trajectory for the non-rigid agent by specifying a *sparse sequence* of waypoints through the scene, enabling the animator varying levels of control depending on the scene requirements. This process can be as simple as giving the deformable body an initial location and a final position, or setting very specific waypoints to closely guide the control of the model.

Our approach consists of two essential stages:

1. Estimated Path Generation (offline)
2. Motion Synthesis by Constrained Dynamics (online)

An estimated path is precomputed during Stage 1 to guide the motion of the deformable

model. It is then refined using our constraint-based dynamics formulation to generate physically plausible motion for the deformable body on the fly. Fig. 3 provides a full overview of our system.

### 3.1 Path Generation

The animator first specifies the desired trajectory using a sparse set of waypoints and various constraints.

**Sampling:** The waypoints are connected by randomly sampling the space within the scene. Our algorithm checks whether the samples are collision-free and connect them to form a graph or a roadmap [17]. If the animator prefers a path that maximizes the clearance between the agent and the nearby obstacles, then the algorithm samples along the medial axis of the workspace to bias the sampling. On the other hand, if the animator prefers the computed trajectory to lie in a certain region, say on the ground plane or in close proximity to certain objects, then sampling in the *contact space* can be used [5].

**Constraints:** An animator can specify either *soft* constraints, that can be loosely satisfied using "constraint forces" (see next section), or *hard* constraints, that must be explicitly enforced. In our system, the hard constraints include non-penetration and deformation constraints; both are strictly satisfied at every simulation step. Our soft constraints include a user-defined smoothness of the path, such that no two segments of the path form an angle larger than a specified bound. A user can also specify a functional deviation from the path to automatically generate certain locomotion patterns or motion style. Although this cannot exactly simulate many motion patterns, it provides the animator a quick and effective way to approximate many standard motion patterns found in animation. For example, in one of the benchmarks described in section 5.1, we specify a sinusoidal deviation from the path to simulate the slithering motion of a snake. We found this increased the realism of the simulated motion dramatically without the need for an explicit dynamics controller.

### 3.2 Motion Synthesis

Once an estimated path is computed, the deformable model must move along this path to reach its final position. Using the constraint-based dynamics framework [6], we transform geometric constraints, such as non-penetration, path following and goal seeking, into constraint forces imposed on the deformable body. We then simulate the dynamics of the deformable agent using a finite element method to automatically generate physically plausible motion.

In order to control the motion of the deformable agent, the animator specifies whether each node will follow the path individually, or whether all the nodes will move along the path uniformly. If the animator selects uniform motion, he/she must specify a group of control nodes, $\mathbf{N}_c = \{n_{c1}, n_{c2}, \ldots, n_{cn}\}$, and optionally a goal node, $n_g$. If not specified, the goal node defaults to the node closest to the center of mass of $\mathbf{N}_c$. During simulation, all the nodes in $\mathbf{N}_c$ feel an identical force calculated to move the body such that $n_g$ will pass through the specified waypoints.

The animator can also specify a distance threshold, $\epsilon_d$, that defines how closely the model must follow the desired path. If the model is outside this threshold, it will be subject to a constraint force directing it towards the path, whereas if it is on the path the model will feel a goal-directed force along the path.

While traversing the path, the model is also constrained not to penetrate any obstacle in the scene, as such a situation would not only violate physical principles, but be visually very unappealing. When interacting with rigid objects in the scene, the non-rigid model deforms to ensure non-penetration, and when the object in the scene is also deformable, both objects deform according to their material properties. In the next section, we describe how we simulate the motion of deformable bodies subject to constraints.

## 4 Modeling of Constrained Deformable Bodies

In this section, we present our algorithm to simulate the motion of deformable bodies subject to geometric and physical constraints. During each time step we compute the forces acting on each body and solve a linear system using an FEM. The hard constraints, including non-penetration and deformation constraints, are then explicitly enforced.

### 4.1 Modeling Material Properties

As the deformable model moves along the estimated path, the dynamic simulation of the deformable system representing the agent is governed by its own material properties. The deformations of each model (or a subset of the vertices in a model) are controlled by the specification of two material parameters, Young's modulus, $E$, and Poisson's ratio, $\nu$. The elastic force a body will exert when deformed is determined

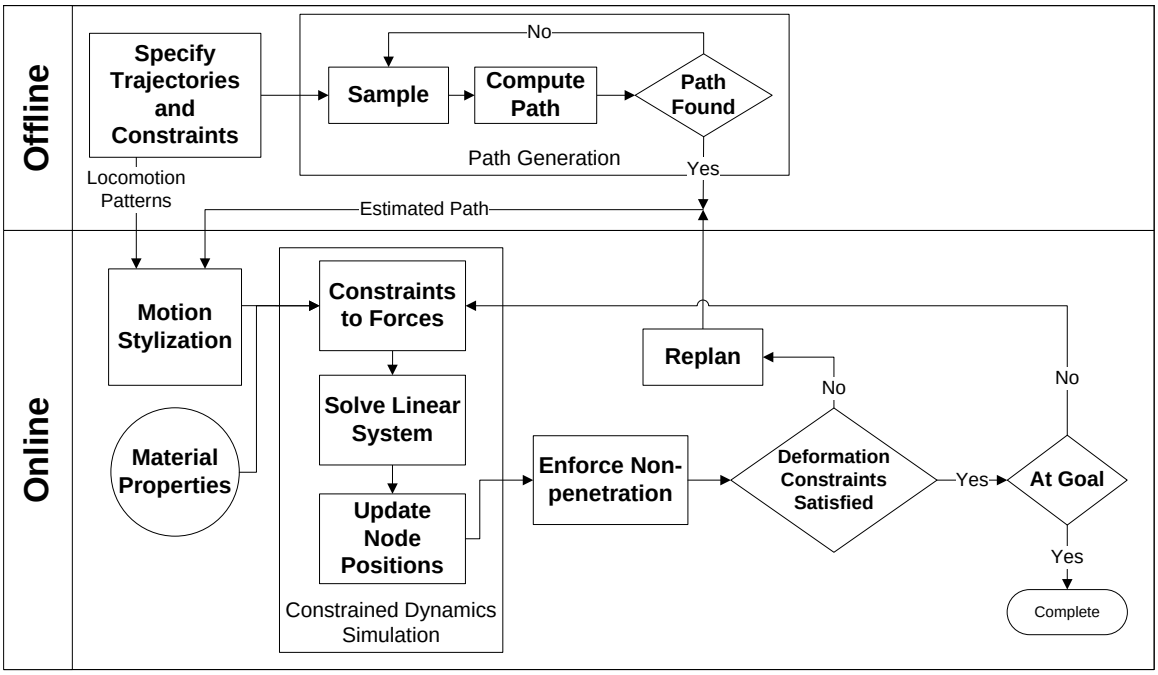



Figure 1: ***System Overview: Each stage of the algorithm is shown. The preprocessing tasks are listed in the upper half while the tasks that are completed at every time step are listed below the line.***

by the Young's modulus as follows,

$$F = \frac{EA_0\Delta L}{L_0}$$

where $A_0$ is the original cross-sectional area and $\Delta L$ and $L_0$ are the change and original length of the body, respectively. Young's moduli vary from around $0.1$ Gigapascals for rubber to around 200 Gigapascals for steel [18]. The change in volume due to a deformation is defined by Poisson's ratio,

$$\frac{\Delta V}{V_0} = (1 - 2\nu)\frac{\Delta L}{L_0}$$

where $\Delta V$ and $V_0$ are the change and original volume of the body, respectively, and $\Delta L$ and $L_0$ are defined as in the Young's modulus. Poisson's ratio is only defined between $-1$ and $0.5$, and it can be seen from the formula that as $\nu$ approaches $0.5$, the change in volume will approach zero for any change in length.

These two properties are used to construct the stiffness matrix that is used in the finite element method. Each property can be set per element in the mesh, allowing for varying properties throughout a model.

### *4.2 Constraint Forces*

*Soft* geometric constraints are converted into forces imposed on the dynamical system of the deformable body. One of them is a path-following force which can be represented by:

$$f_i^{Constraint} = w^{curvature}\frac{(x_i^{proj} - x_i)}{||x_i^{proj} - x_i||}$$

where $w^{curvature}$ is a weight that depends on the curvature of the path segment constraining node $i$, $x_i$ is the position of node $i$ and $x_i^{proj}$ is the projection of $x_i$ on path segment.

The other constraint is the goal-seeking force that can be simply modeled by:

$$f_i^{Attraction} = k(L_i - L_{rest})$$

where $k$ is the spring constant, $L_i$ is the distance between the milestone $q_i$ and node $i$, and $L_{rest}$ is the intended distance between the node and the waypoint, $q_i$. When this force becomes less than a user defined threshold, we conclude that the model reached the waypoint $q_i$.

### *4.3 Deformations*

We simulate deformations using a standard FEM with stiffness warping [19] using linear, tetrahedral elements. We present only a brief overview of this approach and refer the reader to [19] for a complete discussion. The motion equation that describes the dynamic system for a deformable

body is typically described by:

$$M\ddot{\mathbf{x}} + C\dot{\mathbf{x}} + K(\mathbf{x} - \mathbf{x}_0) = \mathbf{f}_{ext} \qquad (1)$$

where $\mathbf{x}_0$, $\mathbf{x}$, $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}} \in \mathbb{R}^{3n}$ are vectors containing the initial position, current position, velocity and acceleration of all the vertices in the body, $M \in \mathbb{R}^{3n\times 3n}$ is a mass matrix, $C \in \mathbb{R}^{3n\times 3n}$ is a damping matrix which is dependent on a damping constant and the mass of each vertex, $K \in \mathbb{R}^{3n\times 3n}$ is the stiffness matrix, and $\mathbf{f}_{ext} \in \mathbb{R}^{3n}$ is a vector representing the external forces being exerted on each vertex in the body. The $K$ matrix represents the coupling of the vertices in the body and therefore the $K(\mathbf{x} - \mathbf{x}_0)$ term in the equation of motion generates the forces that attempt to minimize the internal energy due to the deformations of the body. The stiffness matrix, $K$, is an assembly of the stiffness matrices for each element, $K_e = VB_e^T E_e B_e$ where $V$ is the volume of element, $B_e$ is a matrix representing the shape of the tetrahedral element and $E_e$ is the isotropic elasticity matrix, constructed from the specified material properties, discussed in section 4.1. The elasticity matrix, $E_e$, is nothing more than the extension of Hooke's law to three dimensions, $\sigma = E_e\epsilon$, relating the strain, $\epsilon$, on a tetrahedron to the stress, $\sigma$, generated by that strain.

Once we have constructed the matrices listed above, we must accumulate the external forces, $\mathbf{f}_{ext}$, for each node. Once the forces have been summed and applied to each node, we solve the system using the implicit formulation from [19], by constructing the following linear system for $\dot{\mathbf{x}}^{i+1}$ and solving it using the conjugate gradient method:

$$(M+\Delta tC+\Delta t^2 K)\dot{\mathbf{x}}^{i+1} = M\dot{\mathbf{x}}^i - \Delta t(K\mathbf{x}^i + \mathbf{f}_0 - \mathbf{f}_{ext})$$

After computing the values for $\dot{\mathbf{x}}^{i+1}$, we can update the positions of the nodes in the mesh using $\mathbf{x}^{i+1} = \mathbf{x}^i + \Delta t\dot{\mathbf{x}}^{i+1}$. Once the positions of the nodes are updated, we resolve the collisions and enforce the deformation constraints, discussed in the two sections to follow.

### 4.4 Contact Handling

The contact determination is performed using a bounding volume hierarchy of axis-aligned bounding boxes [20]. The hierarchy is initially generated using a top-down rebuilding algorithm whose complexity is $O(nlogn)$ for $n$ triangles. During each step of the simulation, this hierarchy is updated using bottom-up refitting algorithm, which takes linear time. Since the connectivity of the deformable model is preserved, the refitting algorithm provides tight fitting bounding volumes to the underlying primitives. This hierarchy is used to check for collisions between the deformable model and the obstacles. We can also check for self-collision using the same hierarchy. Exact triangle-triangle intersection tests are then performed using [21]. We found this implementation to be sufficiently fast both for intersection tests and during updates, and incurred collision time on the order of milliseconds for even our most complex geometries–accounting for only 1-2% of the total simulation time.

After the positions are updated at each time step using the finite element method, intersections are found and the vertices of each pair of intersecting triangles are adjusted so the triangles no longer intersect. The velocities of the vertices are adjusted appropriately, with different formulations depending on whether the animator has selected elastic or inelastic collision response. Although we did not implement a friction model, given that we are already operating on all the vertices in collision, it would not be difficult to implement one here if the animator required this functionality.

### 4.5 Deformation Constraints

The animator may also specify multiple possible deformation constraints that will be imposed during the simulation. These constraints are related to the material properties of the model and can be used when a hard threshold must be enforced. At each time step, after the resolution of collisions, any specified deformation constraints are checked and if they are not met then the system is reset to a previous, valid state and a new path is found. Any physical quantity that can be extracted from an FEM could be used as a deformations constraint; we present three constraints that we found particularly useful.

The simplest deformation constraint is a constraint on the percent change in the volume of a tetrahedron. The volume of a tetrahedron is given by

$$V_{element} = \frac{1}{6}(p_1 - p_0) \cdot ((p_2 - p_0) \times (p_3 - p_0)$$

where $p_0 \ldots p_3$ are the locations of the vertices of the tetrahedron. Given this, we can easily find the percent change by taking $\Delta V(t) = \frac{V_0 - V(t)}{V_0}$.

We can also constrain the stress on a tetrahedral element, which can be computed as follows,

$$\sigma_{element}(t) = E_e B_e \hat{\mathbf{u}}$$

where $B_e$ and $E_e$ are the same matrices used to construct the stiffness matrix, $K$, and $\hat{\mathbf{u}} = \mathbf{x} - \mathbf{x}_0$.

Finally, the energy of the entire body can be found from the displacements and the stiffness

matrix,

$$U_{body}(t) = \frac{1}{2}\hat{\mathbf{u}}^T K \hat{\mathbf{u}}$$

It is worth noting that the stress and volume change calculations are computed per tetrahedron, so we can either enforce the constraint on the maximum value or the average, depending on the scenario.

### *4.6 Algorithm*

Once the animator has specified the necessary constraints on the model, the scene and the desired trajectory, the simulation is run. The agent will move along the path indicated by the animator through the scene, by running the following simulation:

**Algorithm 1** Simulation Loop

1: **while** not at the final waypoint **do**
2: Collect external and path following forces and apply them to each node ▷ section 3
3: Solve the linear system ▷ Eqn. 1
4: Update the positions of each node in the model using $\dot{\mathbf{x}}^{i+1}$ and $\mathbf{x}^i$ ▷ section 4.3
5: Identify and resolve collisions generated during the simulation step ▷ section 4.4
6: **if** any deformation constraints are not met **then** ▷ section 4.5
7: Reset the system to a prior, valid state
8: Reject the synethsized motion that results in the violation of the constraints
9: Compute a new trajectory through the remaining waypoints to the goal
10: **end if**
11: **end while**

## 5 Implementation and Results

The meshes used in our simulations are generated using TetGen, which implements the Delaunay refinement algorithm [22]. The $\mu$blas library from boost C++ libraries are used to perform all the vector and matrix calculations. All timing results for our benchmarks were gathered on a 2 GHz laptop with 2 GB of memory. After running the simulation, our final videos were rendered using Blender [23].

### *5.1 Benchmarks*

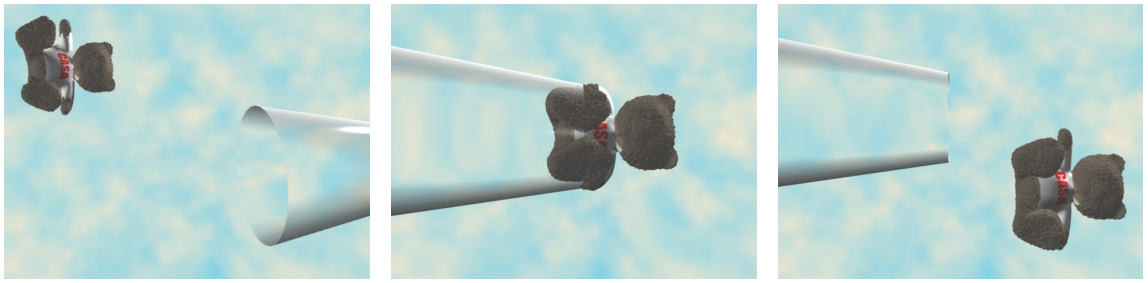
We have tested our prototype system on several challenging benchmarks:

**1. Squishy Bear:** In this benchmark, the bear must travel through the narrow tube from one end to the other (Fig. 5.1). The tube is rigid and the bear is deformable and has a Young's modulus of 0.05 GPa and Poisson's ratio of 0.33. To ensure that the bear actually travels through the tube (and not around it), the animator has placed waypoints inside the tube. We were able to achieve run times of 239 msec per FEM simulation time step while spending 2 msec on collision detection per time step – about 99% of the simulation time in this case is spent running the FEM.

**2. Snake among the Rocks:** In this scenario, the snake is initially positioned at one end of a field of rocks and is given the simple goal of reaching the other side (Fig. 5.1). All the rocks are rigid, however, the snake can deform and has a Young's modulus of 0.15 Gpa and Poisson's ratio of 0.33. We imposed three constraints on the path: first, the angle between successive segments of the path is limited to be less than $\frac{\pi}{8}$; second, the path is limited to be on the ground, and finally, a sinusoidal deviation from the path is specified to simulate the slithering motion of the snake. On this benchmark our algorithm spent 7 msec per time step doing collision detection.

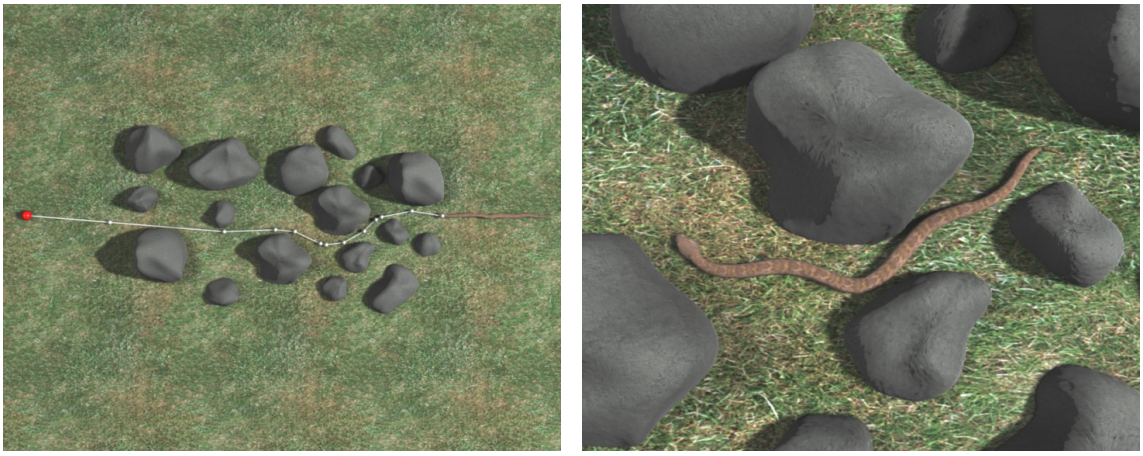
**3. Synchronized Fish Swimming:** In this benchmark we start with four fish on one side of an ocean scene containing rocks and ocean plants. Waypoints have been specified for each fish so that it takes a unique path through the scene to the other side. The animator has also given the fish goals such that upon reaching the other side, they spell out 'CASA'. Each fish is simulated individually, however, they each share the same model and material properties, with a Young's modulus of 0.1 Gpa and Poisson's ratio of 0.33.

**4. Catheter Insertion Procedure:** Catheters are often used to gather diagnostic information, deliver treatments, and perform minimal invasive surgery. Simulations can be carried out to select catheters of appropriate properties and sizes for pre-op planning. In our simulation, a 1.35 mm diameter and approximately 1 m long catheter is inserted into an artery in the leg with the goal of delivering chemotherapy medication to a tumor in the liver (Fig. 5.1). We model the catheter using $6,871$ tetrahedra defined by $3,185$ nodes and set its Young's modulus to 8,700 KPa for silicon [24]. The arteries, which can range in diameter from 2.5mm to 6mm are modeled using 8,330 nodes that define $5,548$ tetrahedra forming a single layer over the outside of the arteries. The Young's modulus for the arteries is set to 20 KPa, an average for a healthy human artery [25]. The model of the liver is simply rendered, since the catheter remains in the interior of the arteries for the entirety of the simulation. Our collision time per time step was 4.4 msec. The arteries model and liver model are both obtained from the 4D NCAT phantom



(a) Initial configuration (b) The bear squishing through the narrowest part of the tube (c) The bear after the simulation is finished, returned to its normal shape

Figure 2: ***A soft bear containing 5,252 movable nodes squeezes through a narrowing, rigid tube. This benchmark takes 239 msec per FEM simulation time step and 2 msec for collision detection per time step.***



(a) Initial configuration with the planned trajectory (b) The snake navigating through the rocks

Figure 3: ***A snake navigates rapidly through a bed of rocks. The rocks contain 5,302 fixed nodes and the snake is made up of 6,832 deformable nodes. Our algorithm spends 304 msec per FEM simulation time step and 7 msec per time step doing collision detection.***

[26].

### *5.2 Timings*

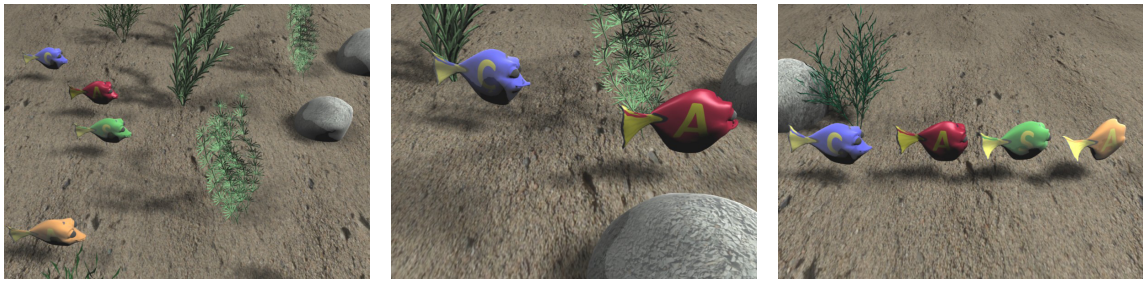
The results from each of our benchmarks is summarized in Table 5.2. The first two columns describe the geometric complexity of the scene, broken down in to rigid nodes, which are only considered during planning and collision response, and deformable nodes, which must be simulated using the FEM. The following three columns show the performance during the three primary phases of the algorithm. The planning time is a fixed cost at the beginning of the algorithm, whereas the simulation and collision times are the ***accumulated*** time for all time steps necessary to traverse the waypoints and reach the goal. The final column shows the time spent solving equation 1 and updating the node positions at each time step.

Due to the complexity of the narrow passages in the catheter benchmark, we biased the path sampling with the medial axis of the arteries. We also were able to accelerate the simulation of the catheter benchmark substantially by taking advantage of the fact that the arteries are not rotating, allowing us to avoid re-calculating the stiffness matrix at each time step.

## Conclusions and Future Work

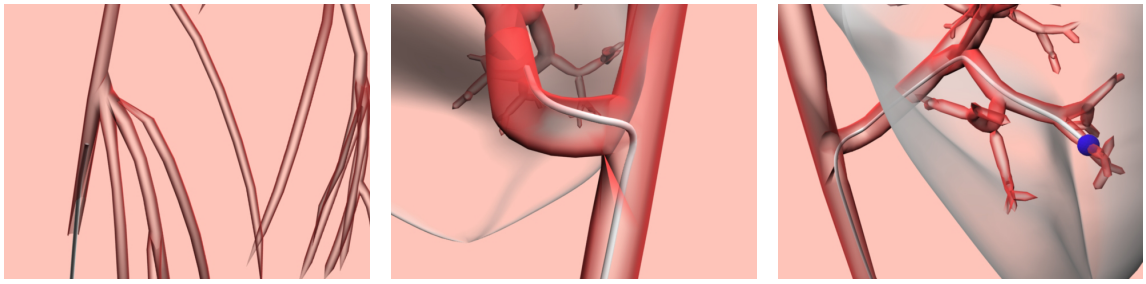
In this paper we presented a new method of constraint-based motion synthesis for deformable models. The animator first specifies a desired path through the scene for the model, then a physical simulation is run to generate physically-plausible motion along that path. Deformations of the model are simulated using a finite element method and computed based on the contact forces between the agent and the environment as well as constraint forces to move the agent along the target path.

There are several possible directions for future work. We would like to exploit many-core



(a) The initial setup of the fish (b) Fishes 'C' and 'A' navigating the environment (c) The fish forming 'CASA'

Figure 4: ***Four fish swim along the ocean floor to form 'CASA'. Each fish is simulated independently and contains 4,818 nodes. Our algorithm takes 237 msec per FEM simulation time step per fish (950 msec combined) and 27 msec for collision detection.***



(a) The catheter as it is being inserted into the femoral artery (b) The catheter turning from the aorta into the hepatic artery (c) The catheter reaching its goal in the liver

Figure 5: ***A catheter is inserted into a complex network of arteries. The catheter is simulated using 3,185 nodes and the arteries are simulated with 8,378 nodes. This simulation requires 142 msec per FEM time step (92 msec for the catheter and 50 msec for the arteries) and we spend 4.4 msec on collision detection per time step.***

computing to accelerate the performance of our algorithm. Although we were able to use this framework to generate pleasing animations for some scenarios, we only provide support for full deformable models that undergo large deformations. Incorporating a skeletal model for the deformable characters (as in [27]) would enable animation of more complicated models and agents. Incorporating this path generation mechanism with a local footstep planner could also provide the animator with the ability to generate realistic motion for humanoid characters more easily. In our current framework, we are able to specify simple kinematic constraints to simulate interesting locomotion patterns, such as those seen among snakes and fishes. One possible and natural extension is to incorporate internal actuation forces and more realistic frictional constraints to model more complex goal-directed behaviors of deformable, articulated characters.

## Acknowledgments

We would like to thank Nico Galoppo for helping with the stability and efficiency of our FEM and Jason Sewall for his help with Blender. This research was supported in part by ARO Contracts DAAD19-02-1-0390 and W911NF-04-1-0088, NSF awards 0400134, 0429583 and 0404088, DARPA/RDECOM Contract N61339-04-C-0043, Intel, and Carolina Development.

## References

[1] J. Popovic, S. Seitz, M. Erdmann, and A. Witkin. Interactive manipulation of rigid body simulations. *Computer Graphics (Proc. of ACM SIGGRAPH)*, pages 209–217, 2000.

[2] C. Twigg and D. James. Many-worlds browsing for control of multibody dynamics. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH)*, 2007.

[3] C. Wojtan, P. Mucha, and G. Turk. Keyframe control of complex particle systems using the adjoint method. *Proc. of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 15–23, 2006.

[4] A. Treuille, A. McNamara, Z. Popovic, and J. Stam. Keyframe control of smoke simulations. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH)*, pages 716–723, 2003.

| | Rigid Nodes | Deformable Nodes | Planning Time (s) | Simulation Time (s) | Collision Time (s) | Simulation Time per Frame (msec) |
|---|---|---|---|---|---|---|
| Total (1) | 1,053 | 5,252 | <0.1 | 358.8 | 3.12 | 239 |
| Bear | 0 | 5,252 | - | 358.8 | - | 239 |
| Scene | 1,053 | 0 | - | 0 | - | - |
| Total (2) | 5,302 | 6,832 | 207.1 | 1,830.9 | 43.4 | 304 |
| Snake | 0 | 6,832 | - | 1,830.9 | - | 304 |
| Scene | 5,302 | 0 | - | 0 | - | - |
| Total (3) | 4,477 | 19,272 | 274.8 | 2,665.7 | 77.6 | 950 |
| Fish | 0 | 19,272 | - | 2,665.7 | - | 950 |
| Scene | 4,477 | 0 | - | 0 | - | - |
| Total (4) | | 11,563 | <0.1 | 879.5 | 44.4 | 142 |
| Catheter | 0 | 3,185 | - | 565.7 | - | 92 |
| Scene | 0 | 8,378 | - | 313.8 | - | 50 |

Table 1: ***Performance of Our System:*** Columns one and two describe the geometric complexity of the scene, column three shows the time required for the planning preprocess, columns four and five show the accumulated time for the two primary runtime phases of the algorithm and the last column shows the time requires to update the FEM at each time step. Due to the complexity of the narrow passages in the catheter benchmark, we biased the path sampling with the medial axis of the arteries, substantially accelerating the planning phase.


[5] S. Redon and M. Lin. Practical local planning in the contact space. *Proc. of IEEE ICRA*, 2005.

[6] Andrew Witkin and William Welch. Fast animation and control of nonrigid structures. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 243–252, August 1990.

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

[8] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson. Physically based deformable models in computer graphics. *Eurographics STAR*, 2005.

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

[10] F. Lamiraux and L. Kavraki. Path planning for elastic objects under manipulation constraints. *International Journal of Robotics Research*, 20(3):188–208, 2001.

[11] L. Guibas, C. Holleman, and L. Kavraki. A probabilistic roadmap planner for flexible objects with a workspace medial-axis-based sampling approach. In *Proc. of IROS*, pages 254 – 259, 1999.

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

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

[14] R. Gayle, M. Lin, and D. Manocha. Constraint based motion planning of deformable robots. *IEEE Conf. on Robotics and Automation*, 2005.

[15] R. Gayle, P. Segars, M. Lin, and D. Manocha. Path planning for deformable robots in complex environments. *Proc. of Robotics: Science and Systems*, 2005.

[16] S. Rodriguez, J. Lien, and N. Amato. Planning motion in completely deformable environments. *Proc. of IEEE Int. Conf. Robot. Autom. (ICRA)*, 2006.

[17] S. M. LaValle. *Planning Algorithms*. Cambridge University Press (also available at http://msl.cs.uiuc.edu/planning/), 2006.

[18] Automation Creations Inc. Matweb: Material property data. http://www.matweb.com.

[19] M. Müller and M. Gross. Interactive virtual materials. *Proc. of Graphics Interface*, 2004.

[20] G. van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools*, 1997.

[21] T. Möller. A fast triangle-triangle intersection test. *Journal of Graphics Tools*, 2(2):25–30, 1997.

[22] J.R. Shewchuk. Tetrahedral mesh generation by delaunay refinement. *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, 1998.

[23] Blender 2.45. http://www.blender.org.

[24] M. Cerverai, M. Dolz, J.V. Herraez, and R. Belda. Evaluation of the elastic behaviour of central venous pvc, polyurethane and silicone catheters. *Journal of Physics in Medicine and Biology*, 1989.

[25] K. Kim, W.F. Weitzel, J.M. Rubin, H. Xie, X. Chen, and M. ODonnell. Arterial elastic modulus reconstruction from in-vivo strain image using arterial pressure equalization. *IEEE Ultrasonics Symposium*, 2004.

[26] W.P. Segars. *Development of a new dynamic NURBS-based cardiac-torso (NCAT) Phantom*. PhD thesis, University of North Carolina at Chapel Hill, 2001.

[27] N. Galoppo, M.A. Otaduy, S. Tekin, M. Gross, and M.C. Lin. Soft articulated characters with fast contact handling. *Computer Graphics Forum*, 26:243–253(11), 2007.