Forward-only Simulation for Agent Search

GIT-GVU-08-01

Andrew Cantino
College of Computing
Georgia Institute of Technology
cantino@gmail.com

Greg Turk
College of Computing
Georgia Institute of Technology
turk@cc.gatech.edu

Charles Isbell
College of Computing
Georgia Institute of Technology
isbell@cc.gatech.edu

ABSTRACT
We present work on optimization search in a domain where random restarts are unavailable, a property of many real-world problems. In particular, we present a new technique for planning and control of agents that have high degree-of-freedom manipulators such as octopus tentacles. The specific type of manipulator that we investigate is known as a muscular hydrostat. A challenge for creating agents that use such manipulators is finding appropriate control parameters to move the end effector to a goal position in a cluttered environment. We physically model hydrostats as collections of masses, springs, and muscles, and perform goal-seeking through optimization over muscle parameters. We introduce a method of optimization called minima filling, a general approach that successfully finds a global minimum without the use of random restarts and through only incremental, stochastic movements. Using minima filling as a base, we construct probabilistic roadmaps (PRMs) and use them to navigate a simulated muscular hydrostat around obstacles in cluttered environments.

Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.2.9 [**Artificial Intelligence**]: Robotics—*Manipulators, Kinematics and dynamics*; I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Animation*

General Terms
random restarts, search, optimization, forward simulation, muscular hydrostats

1. INTRODUCTION
In many scenarios, an agent must optimize or search through a space of solutions. Often, however, due to the nature of the domain, only forward simulation from a known configuration is possible, thus ruling out the use of random restarts, the basis of most stochastic optimization techniques. Examples of real-world problems in this class include motion planning for high degree-of-freedom (DOF) manipulators; path planning for embodied agents that have estimated distances to goal, but no overhead world-views[1]; and the design of complex, physically simulated systems where no closed-form solution is available, such as bridge and airfoil design.

We are interested in studying optimization and search in this class of problems. Consider the many movies and video games that include animations of creatures with tentacles such as octopi, robots or space aliens. To the best of our knowledge, all current planning for animation of tentacle-like manipulators is done by a human animator. We would like to construct an agent that can generate animations of tentacle motion in a possibly cluttered space with minimal human intervention.

Our goal is to find the appropriate control parameters to move a tentacle end-effector from a starting position to a goal position in a cluttered environment. An animator should be able to specify high-level goals for an agent, such as instructing a virtual octopus to reach for a fish that is trying to hide in a small rock cave. The nature of this problem, discussed below, does not allow for random restarts, as we can only achieve plausible tentacle configurations through forward simulation.

While our interest is in controlling virtual agents such as characters for animation, our results for this problem are directly applicable to agents with high degree-of-freedom manipulators in other domains, such as robotics. Additionally, our solution to this search problem is general and is applicable to other agent simulation problems where random restarts are unavailable.

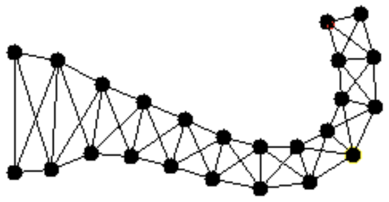
Figure 1: A simulated muscular hydrostat.

In this paper, we model a class of prototypical high DOF manipulators called muscular hydrostats: constant volume, extremely high DOF dexterous manipulators that lack skeletal structure, such as mammalian tongues, elephant trunks, and octopus tentacles. We use a physics-based mass-spring model (see Figure 1) where muscle contraction and expansion is simulated through variation of spring rest lengths. The physical model additionally incorporates

[1]In the case of an embodied agent, random restarts are arguably equivalent to the agent teleporting around its world.

constraint forces to meet the muscular hydrostat constant volume assumption. The high DOF of our simulated tentacles combined with a cluttered landscape of obstacles, results in a very high dimensional configuration space (C-space), necessitating the use of local optimization and search algorithms to control the tentacle motion. As we can only achieve plausible tentacle configurations through forward simulation, random restarts are unavailable. Thus, many standard optimization techniques are unsuitable and we fall in the class of problems mentioned above.

Our main contribution is an optimization method that escapes local minima by modifying the optimization landscape to reduce, or "fill", minima, avoiding the need for random restarts and taking only incremental steps through C-space. This technique is applicable to problems in our domain of interest. Our second contribution is an application of probabilistic roadmaps (PRMs) that avoids explictly computing all free paths between waypoints in order to do motion planning. Using our techniques, we are able to use a modified version of A* search to traverse the roadmap and maneuver a tentacle around obstacles in real time. We demonstrate the efficacy of our approach on simulated muscular hydrostats with 33 degrees of freedom in cluttered 2D environments.

2. MUSCULAR HYDROSTATS

The octopus tentacle is the canonical example of a muscular hydrostat, first termed as such by Kier and Smith [12]. Unlike the bone-supported arms of vertebrates, octopus tentacles are completely supported by their internal musculature. Every point along an octopus's arm can bend in any direction, twist, expand, and contract [8] by taking advantage of the muscle tissue's low compressibility and a combination of lateral, transverse, and oblique muscles [19]. Because of the constant volume constraint imposed by the tissue's low compressibility, contraction of muscles in one direction causes expansion or bending in perpendicular directions. Further detail about octopi is beyond the scope of this paper. We refer the interested reader to Yekutieli et al. [22] for an excellent general overview of recent research on octopus locomotion.

Previous work exists on the simulation of muscular hydrostats and their cousins hydrostatic skeletons. Hydrostatic skeletons are incompressible fluid-filled skeletons found in creatures such as worms and leeches. They differ from muscular hydrostats in that they use incompressibility of fluid as opposed to incompressibility of muscle tissue as their means of support. Alscher and Beyn [1] model leech motion by modeling the motion of connected constant-volume hexahedral segments made of springs. More recently, Yekutieli et al. [20, 21] modeled octopus tentacles similarly as connected constant-volume segments, and were able to reproduce stereotyped reaching movements characteristic to the octopus. While our model is similar to that of Yekutieli et al., we further extend previous work by exploring non-biologically inspired motions and by using the model to facilitate an investigation of high DOF motion control and optimization.

As Figure 2 shows, we model tentacles as a series of connected segments, each with its own area, four associated muscles, and point masses. The two top and bottom muscles are unique to a segment, while the left and right muscles, and all point masses, are shared with the segment's neighbors. Additionally, each tentacle segment contains two non-actuated diagonal cross springs for support against shearing. See Figure 1 for an example of an 11-segment tentacle.

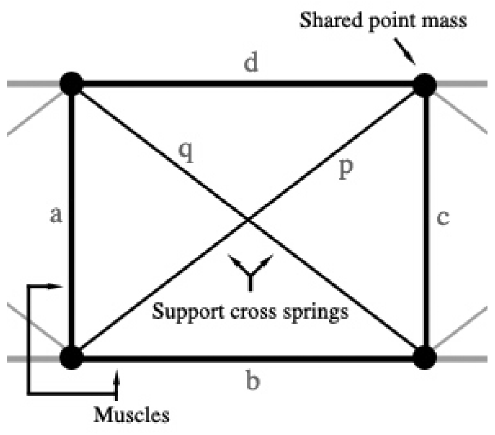

Figure 2: Tentacles are modeled as a series of connected segments, each with its own area, four associated muscles, and point masses. Additionally, each tentacle segment contains two non-actuated diagonal cross springs for support against shearing.

Each muscle is a spring governed by Hook's Law, or $F = -kx$, where k is the spring constant and x is the spring's displacement from its natural rest length. To cause muscle actuation, the spring's rest length is arbitrarily modified by an activation factor that can be either positive or negative:

$$F = -k(\mathit{Length} - \mathit{RestLength} - \mathit{Activation}) \quad (1)$$

Muscular hydrostats have a constant volume constraint imposed by the incompressibility of muscle tissue. To enforce this constraint, we apply restorative forces when a segment's area deviates from its starting area, or *rest area*. This restorative force is calculated as follows. Every segment records its area when first created. During simulation, the area constraint force on a given segment face (a, b, c, or d) is then found according to Equation 2, always acting normal to the given segment face[2]. In Equation 2, A is the current area of the segment, R is its rest area, V is an *area constraint constant*, and L is the length of the segment face whose force we are calculating.

$$F = -(A - R)VL \quad (2)$$

The simulator uses a 4th order Runga-Kutta integrator to numerically solve the interaction dynamics for all instantiated masses and springs on every time step. Additionally, we perform dynamic time-stepping. Each iteration, the integrator solves one full step and two half steps and compares the results. If they differ by more than a predefined threshold, the time step is geometrically reduced for the next iteration, and the double half-step result is kept. Otherwise, the time step is increased for the following iteration. When the integrator is doing well, time step size increases, allowing for faster simulation, and when the integrator performs poorly, time step size decreases until simulation accuracy improves sufficiently.

3. GOAL SEEKING

Our initial goal for motion control is to have the simulated tentacle reach for and touch a target. A tentacle, $\mathbf{t}$, is represented as an n-dimensional vector of muscle activation parameters. Note

[2]As forces can only be applied to masses in the actual simulation, the force at each mass is approximated as the vector average of the forces on its neighboring sides.

that we cannot use inverse kinematics to position the end effector because the angles between segments are determined indirectly through simulation based on muscle activation. We define a cost function $C(\mathbf{t})$ that maps an n-DOF tentacle $\mathbf{t} \in \mathbb{R}^n$ to a scalar:

$$C(\mathbf{t}) = c_1|\mathbf{g} - \hat{\mathbf{g}}|^2 + c_2|\mathbf{v}|^2 + c_3\sqrt{|\mathbf{t} - \mathbf{t_{rest}}|} + c_4 m(\mathbf{t}) \quad (3)$$

Our actual form of $C(\mathbf{t})$, shown in Equation 3, is a heuristically-defined linear combination of the distance from the tentacle endpoint $\hat{\mathbf{g}} \in \mathbb{R}^2$ to a goal position $\mathbf{g} \in \mathbb{R}^2$; the velocity of the endpoint (we prefer tentacle configurations with lower endpoint velocities over configurations with large velocities); a smoothness term defined in terms of the deviation of every muscle parameter from its rest length; and a minima-filling term m that we describe shortly. In this equation, $\mathbf{v}$ is the velocity of $\hat{\mathbf{g}}$ at the time that $C(\mathbf{t})$ is evaluated, $\mathbf{t_{rest}}$ is $\mathbf{t}$ in its most relaxed configuration with all muscles at their rest lengths, and c_1, c_2, c_3, and c_4 are free parameters weighting the linear combination. Notice that with c_1 large, $\mathbf{g}$ is near the global minimum of $C(\mathbf{t})$.

To find a tentacle configuration with a particular end position, we use stochastic descent. We start from an initial tentacle configuration $\mathbf{t}$, and repeatedly check to see if $Sim(\mathbf{t} + \epsilon)$ results in a lower cost than $\mathbf{t}$, where $\epsilon \in \mathbb{R}^n$ is a short randomly-generated vector for perturbation, and $Sim()$ represents forward simulation for a set number of time steps. We transition to any lower cost configurations that we find.

3.1 The Problem of Local Minima

Stochastic descent quickly decends into a local minimum. Unfortunately, that local minimum may be far away from our goal position $\mathbf{g}$. Because stochastic descent only takes transitions that result in lower cost, it cannot find its way out of these minima. A common remedy for this problem uses random restarts[3]: whenever a minimum is detected by a long sequence of failed transitions, a randomly generated $\mathbf{t}$ is created and stochastic descent restarts from there. Unfortunately, we can only achieve a tentacle $\mathbf{t}$ by simulating from an initial configuration, so we cannot use random restarts.

3.2 Minima Filling

We now present our method of minima filling. We initially start in a base tentacle configuration with all muscles at their rest length. Given a goal position $\mathbf{g}$, we use stochastic descent over Equation 3 until we reach a minimum, as determined by failure to find a $\mathbf{t} + \epsilon$ that results in a lower cost configuration after $transition_{max}$ attempts. If our current tentacle endpoint $\hat{\mathbf{g}}$ is not within a minimum distance of the goal $\mathbf{g}$, we determine that we are stuck in a local minimum and we place a *minimum fill point* at $\hat{\mathbf{g}}$. A *minimum fill point* is a heuristically-defined function with a steep falloff that acts to "fill up" a local minimum. We incorporate minimum fill points into $C(\mathbf{t})$ through the $m(\mathbf{t})$ term in Equation 3:

$$m(\mathbf{t}) = \sum_{i=1}^{N} \frac{1}{|\hat{\mathbf{g}} - \mathbf{p_i}|^4} \quad (4)$$

where N is the total number of minimum fill points that have been used, $\hat{\mathbf{g}}$ is again the tentacle endpoint, and $\mathbf{p_i} \in \mathbb{R}^n$ is the ith fill point.

[3]We will discuss the use of simulated annealing as another alternative later.

The net result of creating minimum fill points when a local minimum is detected is to modify the cost space such that the space near the minimum is now unattractive. Although it could take multiple fill points to fill a local minimum, the technique eventually allows stochastic descent to continue. Our technique takes advantage of the fact that in our problem, we can tell when we have not yet reached the global minimum, informing us that continued search is needed. Of course, this is not generally true; however, even in cases where we are not sure if we have achieved a global minimum, we can apply this technique simply by remembering the best minima we have seen so far and by using some other criterion to halt search.

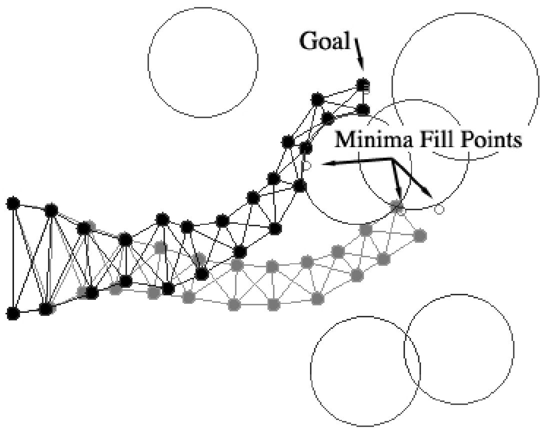

Figure 3: Tentacle (faded) stuck in a local minima, and then reaching goal after dropping minima fill points. Round obstacles clutter the 2D space.

In our problem, local minima are primarily the result of obstacles placed in the environment that create implicit barriers in cost space by disallowing some tentacle configurations. Figure 3 shows a (faded) tentacle stuck in a local minimum created by a concavity between obstacles. It also shows the same tentacle successfully reaching the goal after creating three fill points. As this is a stochastic approach, some runs take deployment of more or fewer fill points before the goal is reached. The cost-space modification approach of using minimum fill points is successful at solving most optimization landscapes that we encounter with our tentacle simulations. As we shall see, it also acts as the necessary preliminary step for construction of probabilistic roadmaps.

4. ROADMAP CONSTRUCTION

One of the applications we are interested in solving is animation: we wish to create smooth animations of the tentacle navigating a cluttered space in real time. Our minima-filling technique is generally successful at moving a tentacle from a starting configuration to a configuration that touches a goal point; however, execution is time-consuming and the resulting search trajectory involves exploration of many successive local minima. As a result, it is most useful to think of the search as providing a target configuration, not an animation of the tentacle moving between start and goal configurations.

Because of these limitations, we use our minima-filling technique as a preprocessing step to build a probabalistic roadmap (PRM) for a particular cluttered environment. PRMs are a path planning technique where a pre-processing stage generates a map of transitions among collision-free configurations in C-space, and an online stage returns valid paths among query configurations at runtime. PRMs are often used because complete motion planning is believed to be PSPACE-hard[11].

We first sample the C-space by randomly picking goal points and using minima filling to reach each goal in turn. For each goal, we store in a roadmap samples of intermediate configurations and the final configuration that reaches the goal. We use a slightly modified version of A* search to do path planning on this roadmap.

Typically, roadmaps are graphs where any two points (configurations) are connected if a collision-free path exists between them. Determining collision-free connections is computationally expensive even when dynamics are known and can be solved in closed form. In our case, we would have to simulate between every pair of neighboring points to determine connectivity. Due to the computational complexity of this task, we choose instead to use an approximation of PRMs in which pairs of configurations are assumed connected if both their endpoints are close together in $\mathbb{R}^2$ and their tentacle space representations are close together in $\mathbb{R}^n$ (that is, they have $|\mathbf{t_1} - \mathbf{t_2}|$ as small as possible).

We define a roadmap node's neighborhood as the k ($=$ 20 for our work) configurations whose endpoints are closest to the query node's, and for which a transition from the query node to the neighborhood node would not directly cause the tentacle to pass through an obstacle. This is partial collision detection: it avoids configurations that are similar in $\mathbb{R}^n$ but for which a transition would directly impinge upon an obstacle. We do this by checking that the line between every mass in tentacle one does not pass through an obstacle on the way to its corresponding mass in tentacle two. Note that this will fail to detect when a transition from one configuration to another is impossible because the tentacle becomes caught on an obstacle, even though it does not actually pass through it.

Given the neighborhoods defined above, we use A* search to perform path planning, and we interpolate tentacles between search nodes. We define our A* path cost as $TotalCostSoFar + EuclideanDistToNextNode^2 + C * |\mathbf{t} - \mathbf{t_{nextnode}}|^2$, where C is a free parameter.[4] To be complete, A* requires a heuristic function that consistently underestimates the distance to goal. Our heuristic is admissible because the sum of the squares of the tentacle similarities over the complete sequence of tentacles followed by A* will always be larger than the similarity of the current tentacle to the tentacle closest to the goal.

5. RESULTS

We compare the effectiveness if our minima-filling technique to stochastic hill climbing using the obstacles shown in Figure 3. Our minima-filling technique successfully reaches a randomly selected goal point from a randomly selected start point 82.8% of the time. Compare this to 52.8% of the time for stochastic hill climbing[5]. The times when minima filling failed can partially be attributed to having a limited number of simulation cycles available, as both RHC and minima filling were capped at a maximum number of cycles. When the maximum number of allowed cycles is increased 100x and the maximum number of fill points is increased 10x, we see the success rate of our minima filling technique increase to 98.4% out of 126 trials. Stochastic hill climbing will not do better with increased time, as it still quickly falls into a local minimum.

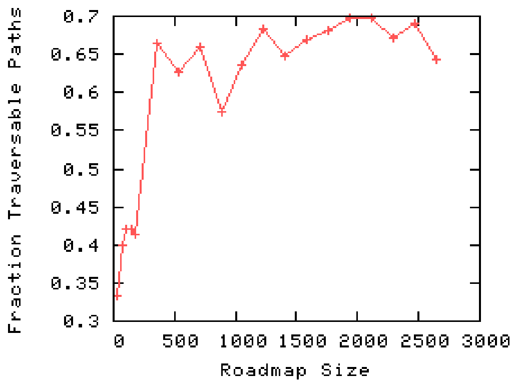

Figure 4: The fraction of successful roadmap traversable paths as a function of roadmap size.

As Figure 4 shows, increased roadmap size leads to higher chances of successful traversal. For each roadmap size, we generated and checked roadmap traversals for many random start and end points for which a straight-line trajectory would impinge on an obstacle.

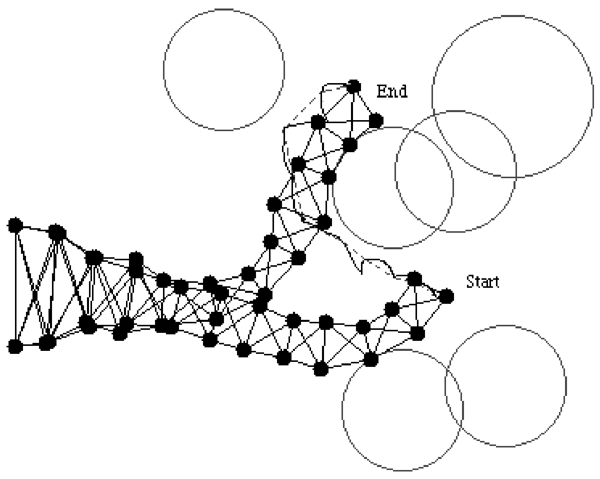

Figure 5: This figure depicts tentacle configurations near the beginning and near the end of a PRM-generated motion path. The dashed line shows the predicted path based on the PRM nodes, while the solid path shows the actual path of the tentacle endpoint.

Figure 5 shows the result of running PRM path generation. The two tentacles shown are configurations found near the beginning and end of the path. The dashed line shows the predicted path based on the locations of the PRM nodes and the solid line shows the actual path taken by the tentacle endpoint. While not as smooth as the predicted path, this path is much smoother than what would be generated by a stochastic minima-filling search for the goal. Once we have a populated PRM, about 2-3 seconds are required for A* to generate a path similar to the one seen in Figure 5 on a modern PC.

6. RELATED WORK

[4]In some experiments we use a slightly different, but still admissible, heuristic.

[5]Both RHC and minima filling success rates were calculated as the average of 180 trials.

In this section we discuss existing work in stochastic optimization, high degree-of-freedom manipulators in robotics, and probabilistic roadmap construction.

6.1 Stochastic Optimization

The literature on optimization methods is vast, and it is well beyond the scope of this paper to do it proper justice. Many techniques require gradients to operate. As our function is difficult to differentiate (due to the area constraint force), and empirically sampling the gradient is computationally infeasible, as every sample must be forward simulated from the current position, we use stochastic optimization. In particular, we use stochastic hill climbing, combined with a method of minima filling.

To our knowledge, existing research in this area explores two basic directions: spaces with differentiable cost functions, and spaces where large steps can be taken. Typical methods with differentiable cost functions are Bridging [15] and the use of Filled Functions [16, 23, 14]. Filled Functions successively generate new cost functions where minima have been turned into maxima. These approaches do not work for us because of the requirements of our domain. Our approach is actually most like variations on Tabu search [7] for continuous domains [17, 4]. Tabu search is a metaheuristic search that remembers where it has been and uses cycles of *intensification* and *diversification* to find good solutions.

Our approach is different from continuous modifications of Tabu search in a number of ways. First, we explicitly modify the cost landscape to fill minima. Second, we do not need to maintain explicit *Tabu lists* because we implicitly memorize explored regions of the space by filling minima. Third, Tabu search revisits locations in the intensification phase and when aspiration criteria are met, while we never have to explicitly revisit regions. Instead, we revisit regions if we fall back into them because they have not been completely filled by minimum fill points. Finally, we do not need to make large transitions in state space (*e.g.* random restarts)—and, in fact, we cannot—while Tabu search sometimes jumps around during the intensification phase.

Another popular stochastic search technique worth mentioning here is simulated annealing. In simulated annealing, search is allowed to probabilistically take locally "bad" steps. It has proven a powerful theoretical and practical technique. Although we do not report in detail here, we found empirically that simulated annealing performs poorly in our domain, even when run for a large number of iterations and with a slow cooling cycle. We conjecture that our cost landscape is full of shallow bumps, and that each bump provides a great deal of information. Therefore, it is best to follow a gradient all the way to its local minima before dismissing it.

An alternative stochastic search technique is MIMIC [3] and related families of algorithms. These techniques attempt to characterize the entire cost space with a probability distribution, successively refining that distribution until only "good" solutions are probable. Typically, these distributions are represented by dependency trees or other parametric functions. One could view the minima fillers as non-parametric representations of such distributions, where each filler warps the probability space around it. This approach generalizes less well across the space, but may be better able to capture the local nature of obstacles, at least insofar as locality in C-space and Euclidean space are related.

6.2 Robotics and Path Planning

Although the robotics community has explored tentacle-like designs for robots, most of these *hyper-redundant* robots are not muscular hydrostats, as current technological limitations require bone-like structures. Chirikjian and Burdick [5] created a kinematic model for hyper-redundant manipulators in which the manipulator is treated as a "backbone curve" that captures the basic shape of the robot. Hannan and Walker [9] built a robotic "elephant trunk", and developed motion planning and control algorithms for it by approximating its four 2-DOF sections as having constant curvature. That is, each section is treated as an arc. Due to coupling springs between each section, it acts as if it has 33 degrees of freedom. Given the assumption of constant curvature, Hannan and Walker are able to derive closed form kinematic equations and use them for motion planning. We take a more computationally intensive approach but make no simplifying shape assumptions, optimizing directly on the hydrostat's muscle parameters. There has also been a large amount of research on serpentine robots, for example [10]. With serpentine robots, the body follows the same path as the head. This is not generally true with our tentacle model.

In general, the computational complexity that we encountered when building our roadmaps is a common issue with PRMs. For a good overview of methods for making PRMs tractable in high dimensions, see [6]. Our PRMs are relatively primitive by modern standards, and we present them primarily as an application for our muscular hydrostat motion control problem. There are a number of existing results that could help improve our techniques. Lazy PRM [2], which assumes roadmap paths are clear when building the PRM, and tests paths only as needed, could be used to test node connectivity at runtime. This would avoid our sometimes-ineffective assumption that similar configurations are traversable. Additionally, the work by Song et al. [18] in roadmap refinement at runtime could also increase performance. Some work has been done toward PRM optimization [6]. Our PRM generation methods use simple random sampling, and could benefit from the work of [13] into rapidly-exploring random trees.

7. DISCUSSION

This research addresses an example of a class of problems in which an agent is embodied, as with a robot, or is expensively simulated, as with our tentacle models. In these types of problems, random restarts are not available. We have shown that our minima filling technique works well in one such domain, and we believe that it will prove beneficial for exploration or optimization in other problems of this class. In future work, we would like to apply our minima filling technique to other domains where expensive simulation is required.

In this paper, we present work into simulation and motion control for muscular hydrostats: constant volume, extremely high degree-of-freedom dexterous manipulators, of which octopus tentacles are an example. We seek to minimize the hydrostat's endpoint distance to a goal point in a cluttered environment. We use a method of minima filling that modifies our cost space so that a stochastic descent algorithm can escape local minima incrementally. We find that this technique significantly improves the performance of standard stochastic gradient descent on our problem domain.

Building upon our minima filling technique, we construct a probabilistic roadmap to enable rapid tentacle motion planning. Our domain is difficult because we cannot determine collisions without the use of expensive forward simulation, so we assume that tentacles with similar tentacle-space configurations and similar endpoints are connected and we perform A* motion planning over these neighborhoods.

A number of methods for improving our PRM generation and utilization are available as detailed in section 6.2. Most prominently, Lazy PRM could be used to test roadmaps at runtime and backtracking could be employed when collisions are detected so that a new roadmap route can be tried. Once a clear path is found, a final motion can be generated.

Additionally, future research should compare minima filling directly to implementations of continuous Tabu search and other similar optimization methods in domains where random restarts are available.

8. REFERENCES

[1] C. Alscher and W.-J. Beyn. Simulating the motion of the leech: A biomechanical application of daes. *Numerical Algorithms*, 19:1–12, 1998.

[2] R. Bohlin and L. E. Kavraki. Path planning using lazy prm. In *Proc. IEEE Int. Conf. on Robotics & Automation*, 2000.

[3] J. D. Bonet, C. Isbell, and P. Viola. Mimic: Finding optima by estimating probability densities. In *Advances in Neural Information Processing Systems*, volume 9, 1997.

[4] R. Chelouah and P. Siarry. Tabu search applied to global optimization. *European Journal of Operational Research 123/2 Special issue on combinatorial optimization*, pages 30–44, 2000.

[5] G. S. Chirikjian and J. W. Burdick. A model approach to hyper-redundant manipulator kinematics. *IEEE Transactions on Robotics and Automation*, 10(3), 1994.

[6] L. K. Dale. *Optimization Techniques for Probabilistic Roadmaps*. PhD thesis, Texas A&M University, 2000.

[7] F. Glover. Tabu search: A tutorial. *Interfaces*, 20:74–94, 1990.

[8] Y. Gutfreund, T. Flash, Y. Yarom, G. Fiorito, I. Segev, and B. Hochner. Organization of octopus arm movements: A model system for studying the control of flexible arms. *The Journal of Neuroscience*, 16(22):7297–7307, 1996.

[9] M. W. Hannan and I. D. Walker. The 'elephant trunk' manipulator, design and implementation. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics Proceedings*, page 14, 2001.

[10] W. Henning, F. Hickman, and H. Choset. Motion planning for serpentine robots. *Proceedings of ASCE Space and Robotics*, 1998.

[11] Y. K. Hwang and N. Ahuja. Gross motion planninga survey. *ACM Computing Surveys*, 24:219 – 291, 1992.

[12] W. Kier and K. Smith. Tongues, tentacles and trunks: The biomechanics of movement in muscular-hydrostats. *Zoological Journal of the Linnean Society*, 83:307–324, 1985.

[13] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 473–479, 1999.

[14] X. Liu. Finding global minima with a computable filled function. *Journal of Global Optimization*, 19:151–161, 2001.

[15] Y. Liu and K. L. Teo. A bridging method for global optimization. *Journal of Australian Mathematical Society Series B*, 41:41–57, 1999.

[16] G. Renpu. A filled function method for finding a global minimizer of a function of several variables. *Mathematical Programming*, 46:191–204, 1990.

[17] P. Siarry and G. Berthiau. Fitting of tabu search to optimize functions of continuous variables. *Intr. Journal for Numerical Methods in Engineering*, 40:2449–2457, 1997.

[18] G. Song, S. Miller, and N. M. Amato. Customizing prm roadmaps at query time. In *Proc. IEEE Int. Conf. on Robotics & Automation*, pages 1500–1505, 2001.

[19] I. D. Walker, D. M. Dawson, T. Flash, F. W. Grasso, R. T. hanlon, B. Hochner, W. M. Kier, C. C. Pagano, C. D. Rahn, and Q. M. Zhang. Continuum robot arms inspired by cephalopods. In *Proceedings of SPIE*, volume 5804, pages 303–314, 2005.

[20] Y. Yekutieli, R. Sagiv-Zohar, R. Aharonov, Y. Engel, B. Hochner, and T. Flash. Dynamic model of the octopus arm. i. biomechanics of the octopus reaching movement. *Journal of Neurophysiology*, 94:1443–1458, 2005.

[21] Y. Yekutieli, R. Sagiv-Zohar, R. Aharonov, Y. Engel, B. Hochner, and T. Flash. Dynamic model of the octopus arm. ii. control of reaching movements. *Journal of Neurophysiology*, 94:1459–1468, 2005.

[22] Y. Yekutieli, G. Sumbre, T. Flash, and B. Hochner. How to move with no rigid skeleton? the octopus has the answers? *Biologist*, 49(6), 2002.

[23] L.-S. Zhang, C.-K. Ng, and D. L. andWei Wen Tian. A new filled function method for global optimization. *Journal of Global Optimization*, 28:17–43, 2004.