Multi-Level Free-Space Dilation for Sampling Narrow Passages in PRM Planning

David Hsu*†, Gildardo Sánchez-Ante†, Ho-lun Cheng* and Jean-Claude Latombe‡

*Department of Computer Science, National University of Singapore, Singapore 117543, Singapore
†Computer Science Program, Singapore MIT Alliance, Singapore 117543, Singapore
‡Department of Computer Science, Stanford University, Stanford, CA 94305, USA

***Abstract*— Free-space dilation is an effective approach for narrow passage sampling, a well-recognized difficulty in probabilistic roadmap (PRM) planning. Key to this approach are methods for dilating the free space and for determining the amount of dilation needed. This paper presents a new method of dilation by shrinking the geometric models of robots and obstacles. Compared with existing work, the new method is more efficient in both running time and memory usage. It is also integrated with collision checking, a key operation in PRM planning. The efficiency of the dilation method enables a new PRM planner which quickly constructs a series of dilated free spaces and automatically determine the amount of dilation needed. Experiments show that both the dilation method and the planner work well in complex geometric environments. In particular, the planner reliably solved the most difficult version of the alpha puzzle, a benchmark test for PRM planners.**

I. Introduction

Probabilistic roadmap (PRM) planning is a successful approach for motion planning of robots with many degrees of freedom (DOFs) in complex geometric environments (see, *e.g.*, [1], [3], [8], [10], [13], [15], [17], [24], [25]). The main idea of PRM planning is to sample at random a robot's configuration space $\mathcal{C}$ with a suitable probability distribution and capture the connectivity of $\mathcal{C}$ in an extremely simplified representation, called a probabilistic roadmap. A roadmap is a graph whose nodes are the collision-free configurations sampled from $\mathcal{C}$ and whose edges are simple collision-free paths between the nodes. This way, PRM planners avoid the prohibitive cost of constructing a complete representation of $\mathcal{C}$.

It has been argued that the success of PRM planners depends critically on the assumption that $\mathcal{C}$ verifies favorable visibility properties [12]. Indeed, many experiments have shown that PRM planners behave poorly, when $\mathcal{C}$ contains regions with poor visibility, in particular, narrow passages. Narrow passages are small regions whose removal changes the connectivity of $\mathcal{C}$. Due to their small volumes, the probability of sampling at random in narrow passages is low. This makes it difficult for PRM planners to capture the connectivity of $\mathcal{C}$ well.

To address this difficulty, one approach is to dilate the free space $\mathcal{F}$ [11], [19], defined as the collision-free subset of $\mathcal{C}$. The dilated free space $\mathcal{F}'$ has better visibility properties. In particular, dilation widens narrow passages and makes the planning problem easier. A solution path found in $\mathcal{F}'$ is then deformed into one in $\mathcal{F}$. The key element of this approach is an effective dilation method. Another important element is to determine the amount of dilation needed. Too little dilation may fail to improve the visibility properties of the free space sufficiently for a solution path to be easily found; too much dilation may change the free space so much that it is difficult to deform the solution path in $\mathcal{F}'$ into one in $\mathcal{F}$.

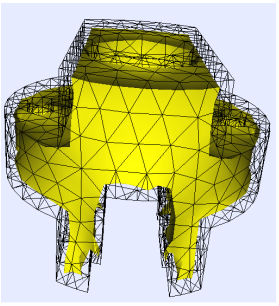


Fig. 1. The geometric model of a mechanical part from the ABB IRB 2400 robot before shrinking (wireframe) and after shrinking (shaded in yellow).

In this paper, we present a new method of dilation by shrinking the geometric models of robots or obstacles (see Fig. 1 for an example). This method guarantees that a shrunken model M' is always contained inside the original model M so that $\mathcal{F}'$ is a dilation of $\mathcal{F}$, *i.e.*, $\mathcal{F} \subseteq \mathcal{F}'$. It has several important advantages when used to dilate $\mathcal{F}$ in PRM planning:

- This method is efficient in both running time and memory usage. Compared with earlier dilation methods based on medial axes [19], the new method is more efficient and much simpler to implement.
- This method can shrink M in a local region without affecting others and thus avoid unnecessarily changing the free space during dilation.
- This method creates a data structure fully integrated with that used for collision checking, a key operation in PRM planning. There is no need to re-compute the bounding volume hierarchy for collision checking when a new shrunken model is generated. This saves both computation time and memory space.

Taking advantage of the efficiency of this dilation method, our new PRM planner constructs a series of dilated free spaces, $\mathcal{F}(s)$, for $0 \leq s \leq 1$, where $\mathcal{F}(0) = \mathcal{F}$ is the original free space and $\mathcal{F}(1)$ is the maximally dilated free space. To determine the amount of dilation, the planner performs a binary search on the dilation parameter s and constructs $\mathcal{F}(s)$ on the fly. In our experiments, the planner reliably solved the most difficult version of the alpha puzzle, a benchmark test for PRM planners.

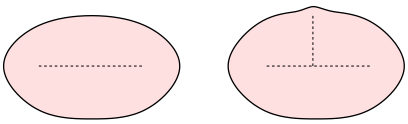


Fig. 2. The dashed lines indicate the medial axes of the shaded objects. A small perturbation on the object changes its medial axis dramatically.

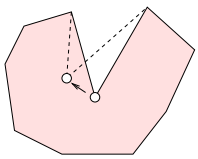


Fig. 3. Moving a vertex p arbitrarily may fail the requirement $M' \subseteq M$. The figure shows a cross section of a polyhedron.

II. Previous Work

The difficulty of narrow passage sampling and its importance to PRM planning have long been recognized [15], [11]. We can characterize narrow passages formally using visibility properties in $\mathcal{C}$ [13]. Thus, many techniques have been proposed to increase sampling density in regions of $\mathcal{F}$ expected to have poor visibility (see [5] for a comprehensive survey). One approach is to sample more densely near the boundary of $\mathcal{F}$ [1], [3], because points in narrow passages, which have poor visibility, lie close to the boundary. Another approach is to use workspace information to locate regions with poor visibility [24], [8], [10], [16], [25], because narrow passages in the workspace often suggest the presence and the location of narrow passages in the configuration space. Instead of trying to locate regions with poor visibility, an alternative is to check the definition of visibility explicitly [23].

Locating regions of $\mathcal{F}$ with poor visibility is often difficult, because we do not have an explicit representation of $\mathcal{F}$. A different approach is to dilate $\mathcal{F}$ in order to improve its visibility properties [11], [19]. One way of dilating $\mathcal{F}$ is to allow a small penetration of the robot into the obstacles [11]. Unfortunately, penetration distance is difficult to compute efficiently [18], which has prevented this technique from being used in complex geometric environments. The more recent small-step retraction planner (SSRP) dilates $\mathcal{F}$ by shrinking the geometric models of the robot or the obstacles towards their medial axes and greatly improves the planner's performance [19]. However, a drawback of SSRP is that the medial axis is difficult to compute even approximately. The medial axis may also change dramatically by a small perturbation of the geometry. See Fig. 2 for an example. Due to these difficulties, the work on medial axis computation in three dimensions has been limited [7], [9], [21], [25]. Furthermore, the amount of dilation, an important parameter in SSRP, must be chosen manually. Our new planner also uses the free-space dilation approach, but our dilation method is much more efficient in both running time and memory usage, and the planner chooses the amount of dilation automatically.

The idea of shrinking the geometry of a robot has also been used in motion planners quite different from PRM. For example, the planner in [2] uses shrinking to estimate the penetration of a path into the obstacles and tries to modify the path iteratively to reduce the amount of shrinking needed, until a collision-free path is obtained. However, this planner has a very limited shrinking method that applies only to simple geometric objects such as rectangular boxes.

Algorithm 1 `Shrink`(M, s)

1: **for** each surface vertex $p \in M$ **do**
2: star$(p) \leftarrow \{\sigma \in T \mid p \in \sigma\}$.
3: $v_p \leftarrow \bigcup$ star(p).
4: Compute a point $\tau_p \in \text{kr}(v_p)$ so that p can move freely within $\overline{p\tau_p}$.
5: $p(s) \leftarrow p + (s \cdot \min\{\varepsilon, \|p\tau_p\|\}) \frac{p\tau_p}{\|p\tau_p\|}$

III. The Shrinking Algorithm

Our shrinking algorithm takes as input a polyhedral model M. It first computes a tetrahedralization T of M, *i.e.*, partitions the interior of M into a set of tetrahedra. Although some polyhedra cannot be tetrahedralized, *e.g.*, the Schönhardt polyhedron, tetrahedralization is always possible if we add a small number of additional vertices to M [4], [6]. To shrink M, we move each surface vertex of M towards the interior of M and ensure that the shrunken model M' is always contained inside M, *i.e.*, $M' \subseteq M$. The amount of shrinking is controlled by setting a parameter ε, which determines the maximum distance that each vertex can move. This implies that for every point on M, there exists a point on M' such that the distance between the two points are less than ε. Formally, the Hausdorff distance between the surfaces of M and M' is at most ε. The shrinking process is sketched out in Algorithm 1.

A. Moving a surface vertex

To move a surface vertex p of M, we first compute the volume to which p can move in order to ensure $M' \subseteq M$. Arbitrary movement of p may fail the requirement. See Fig. 3 for an example. This volume is the *kernel of the star of* p, for brevity, also called the kernel of p.

In the tetrahedralization of M, the *star* of a vertex p, star(p), is the set of all tetrahedra to which p belongs. It can be computed by collecting the tetrahedra adjacent to p in T (line 2 of Algorithm 1). Taking the union of these tetrahedra gives the volume v_p occupied by the star (line 3).

We then compute a point τ_p in the kernel of v_p, which is a polyhedron, so that p can move freely within the line segment $\overline{p\tau_p}$ (line 4). Formally, the kernel of a polyhedron v_p is

$$\text{kr}(v_p) \quad = \quad \{x \in v_p \mid \overline{xr} \subseteq v_p \text{ for all } r \in v_p\}.$$

A polyhedron with a non-empty kernel is *star convex*. Intuitively, if we imagine that v_p is a room, the kernel is the set of points from which we can see the entire room. One way of computing the kernel is to first take all the triangles on the boundary of v_p. For each triangle, there is a plane that contains the triangle and divides the space into two half-spaces. Take the half-spaces containing v_p. The kernel kr(v_p) is the intersection of all such halfspaces.

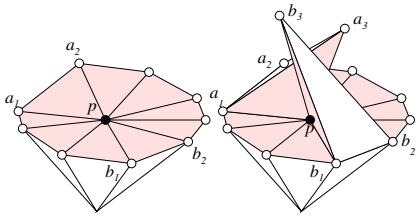


Fig. 4. The construction of a surface vertex p whose kernel is a singleton. The shaded triangles are the surface triangles. After adding two new tetrahedra (right) to the tetrahedralized model (left), the kernel of p is a singleton. The reason is that the kernel must lie within the intersection of the two wedges formed by the two new tetrahedra, and they only intersect at p.

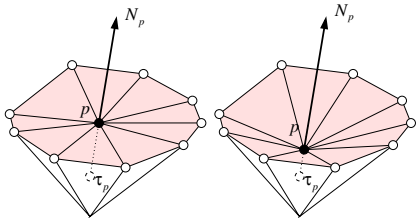


Fig. 5. After computing the point τ_p, we move vertex p towards τ_p.

By definition, if p moves within its kernel, the new polyhedron is always contained within the original one, *i.e.*, $M' \subseteq M$. The result is stated below formally.

Theorem 1 *If every surface vertex of a model M moves within the kernel of its star one by one, the shrunken model M' is contained within M.*

In some cases, the kernel may be a singleton, namely $\mathrm{kr}(v_p) = \{p\}$. Although v_p is always star convex in our case, the vertex p cannot move to any other place if its kernel is a singleton. See Fig. 4 for an example. In such cases, we can either leave the vertex unmoved, or cut out from the model a sufficiently small neighborhood of the vertex.

Now we may choose τ_p to be any point in $\mathrm{kr}(v_p)$ and move p within $\overline{p\tau_p}$. To be specific, we have decided to move p along the direction of the approximated surface normal at p, N_p, which is estimated with the *Mean Weighted by Angle* method [14]. Let ℓ be a line going through p and having the same direction as N_p. We compute the intersection of ℓ and $\mathrm{kr}(v_p)$. Since $\mathrm{kr}(v_p)$ is convex, the intersection is a line segment $\overline{p\tau_p}$. However, it is possible that ℓ does not intersect $\mathrm{kr}(v_p)$ except at p. In this case, we simply project ℓ onto the boundary of $\mathrm{kr}(v_p)$.

The method desribed above for finding τ_p is conceptually simple, but the more efficient method is to use linear programming, which computes τ_p as an extreme point lying in the direction of $-N_p$ and obeying all the half-space constraints.

Finally, given a shrinking factor $s \in [0, 1]$ and the maximum distance ε that each vertex can move, we move p towards τ_p to a new position (line 5):

$$p' \quad = \quad p + (s \cdot \min\{\varepsilon, \|p\tau_p\|\}) \frac{p\tau_p}{\|p\tau_p\|}.$$

See Fig. 5 for an illustration.

B. Integration with collision checking hierarchies

Most hierarchical collision checking algorithms [18] precompute bounding-volume trees. Every leaf node of a tree contains a bounding volume enclosing a surface triangle of a model. Every non-leaf node u contains a volume enclosing the volumes contained in the children of u. Our planner may use several versions of a model with different shrinking factors. We want to avoid constructing a new tree for every version. Since every vertex moves a maximum distance ε, we can enlarge the size of each leaf bounding volumes by ε. For each surface triangle, this enlargement ensures that the bounding volume now encloses every version of the triangle for any shrinking factor. Other than this extension, we do not need any other changes in the collision checker and requires no additional memory space.

IV. The Multi-Level Dilation Planner

Narrow passages are difficult to sample, because of their small volumes. Using Algorithm 1, we shrink the geometry of the robot or the obstacles and dilate the free space $\mathcal{F}$. In the dilated free space $\mathcal{F}'$, narrow passages become wider, thus simplifying planning. A solution path found in $\mathcal{F}'$ can then be deformed into one in the original free space $\mathcal{F}$. An important issue here is to determine the dilation parameter s, which controls the amount of shrinking. To address this issue, we take advantage of the efficiency of our dilation method and perform a binary search on s.

Algorithm 2 shows the main steps of our *multi-level dilation planner* (MLDP). To start, we set the dilation parameter s to 0.5 (line 3) and dilate $\mathcal{F}$ by calling $\texttt{Shrink}(M, s)$, which uses the method described in Section III to shrink the geometric model M of the robot and the obstacles (line 4). We then invoke a PRM planner in the dilated free space $\mathcal{F}'$ to find a path between the given initial configuration q_{init} and the goal configuration q_{goal} (line 5). Any PRM planner can be used here. In our implementation, we chose an efficient single-query PRM planner called SBL [20]. If no path is found in $\mathcal{F}'$, then s may be too small and the free space is not sufficiently dilated to allow a path to be easily found. We then increase s (line 7). If a path γ' is found in $\mathcal{F}'$, γ' may still be in collision in the actual free space $\mathcal{F}$. We must "repair" γ' by deforming it into a valid path in $\mathcal{F}$ (line 9). If the deformation fails, then the dilation parameter is too large: the dilation may have altered the connectivity of $\mathcal{F}$ and introduced spurious passages, thus making it difficult to deform a path in $\mathcal{F}'$ into one in $\mathcal{F}$. We then must decrease s (line 11). After updating s, we repeat the process until a maximum number K of iterations is reached.

A. Finding a path in the dilated free space

SBL takes as input the shrunken model M' as well as q_{init} and q_{goal}. SBL grows two trees rooted at at q_{init} and q_{goal}, respectively. The nodes of each tree are sampled configurations, called *milestones*. In every iteration, SBL picks a milestone q in one of the two trees and samples at random in the neighborhood of q until a collision-free configuration q' is obtained. It then adds q' as a child of q in the corresponding

Algorithm 2 Multi-level dilation planner (MLDP).

$s_{\text{low}} \leftarrow 0, s_{\text{high}} \leftarrow 1$.
for $i = 1, 2, \ldots, K$ **do**
  $s \leftarrow (s_{\text{low}} + s_{\text{high}})/2$.
  $M' \leftarrow \texttt{Shrink}(M, s)$.
  $\gamma' \leftarrow \texttt{SBL}(q_{\text{init}}, q_{\text{goal}}, M')$.
  if $\gamma' = \text{NIL}$ **then**
    $s_{\text{low}} \leftarrow s$.
  else
    $\gamma \leftarrow \texttt{Repair}(\gamma', M)$.
    if $\gamma = \text{NIL}$ **then**
      $s_{\text{high}} \leftarrow s$.
    else
      return γ.
return NIL.

tree and creates an edge between q' and the closest milestone in the other tree, thus establishing a candidate path between q_{init} and q_{goal}. SBL then checks whether there is a collision-free straight-line connection between every pair of consecutive milestones in the candidate path. If so, SBL returns the path as the solution. Otherwise, SBL removes from the trees the edge with collision and proceeds to the next iteration. SBL exits with failure if it does not find a path after generating a given maximum number of milestones. SBL is probabilistically complete, with fast convergence rate. More details are available in [20].

B. Repairing a path

Suppose that $\texttt{SBL}(q_{\text{init}}, q_{\text{goal}}, M')$ returns successfully with a path γ'. The path γ' may contain milestones or edges which lie in the dilated free space $\mathcal{F}'$ but not in the actual free space $\mathcal{F}$. Thus, we must deform γ' so that it lies entirely in $\mathcal{F}$.

To repair a milestone q, we sample in a neighborhood of q. If a new free milestone is obtained, we continue to the next milestone in γ'. If not, we increase the size of the neighborhood and sample again. This process repeats until a given maximum number of iterations is reached or a free milestone is obtained.

To repair an edge between two milestones q and q', we break the edge at its midpoint q_{m} and check q_{m} for collision. If q_{m} is in collision, we use the procedure in the previous paragraph to repair it. If the repair succeeds and provides a new milestone q'_{m}, the edge is split into two new edges, one between q and q'_{m} and one between q'_{m} and q'. We then recursively repair the two new edges until a given resolution is reached.

V. Experimental Results

A. Shrinking

To test our shrinking algorithm, we used an ABB manipulator robot (see Fig. 6). We tetrahedralized each link using TetGen [22] and applied Algorithm 1 to compute a shrunken version ($s = 1$). Table I lists for every part of the robot, the numbers of tetrahedra (N_{tetra}), the number of triangles on the surface (N_{tri}), as well as the time for tetrahedralization (T_{tetra}) and the time for shrinking (T_{sh}).

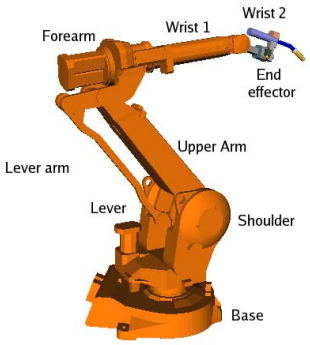


Fig. 6. ABB IRB 2400 manipulator.

TABLE I
Running times for shrinking the ABB robot.

Part	N_{tetra}	N_{tri}	T_{tetra} (sec.)	T_{sh} ($\times 10^{-5}$ sec.)
base	2491	5890	0.8	1.3
shoulder	22360	52399	11.6	11.4
upper arm	2424	5857	1.5	1.5
lever arm	1246	2878	0.6	0.6
lever	2096	4906	0.9	1.0
forearm	4089	9466	1.5	2.0
wrist 1	12810	29575	5.0	6.0
wrist 2a	2025	4829	0.9	1.1
wrist 2b	6660	15619	2.4	3.4
total	56201	131419	25.2	28.3

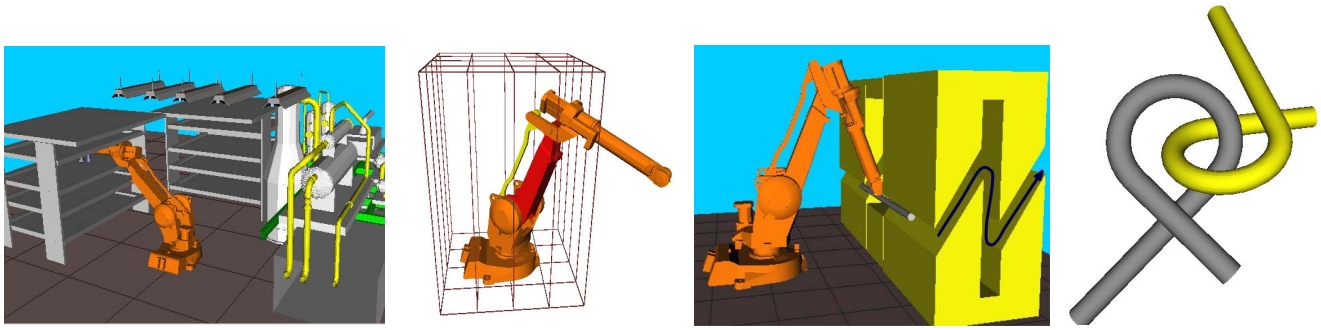
For this experiment, we ran TetGen and Algorithm 1 on a PC with an 1.6 GHz processor. The total time for tetrahedralizing the robot is 25.2 seconds, and the time for shrinking the robot is 2.8×10^{-4} seconds. In comparison, the shrinking method based the medial axis took 4671 seconds to compute the approximate medial axis of the robot and 8.3 seconds to compute a shrunken version of the robot on a PC with an 1 GHz processor [19]. It should be noted that our computer is slightly faster, but the drastic improvement in running times cannot be possibly explained by the computer speed alone.

In general, shrinking is considered as precomputation for robots. However, if the robot is too thin and cannot be shrunken effectively, we may have to shrink the obstacles in the environment instead. Since the environment changes more often, we must shrink on the fly. The efficiency of our shrinking method then becomes a significant advantage.

B. Comparing MLDP with SBL

SBL is an efficient single-query planner capable of solving complex motion planning problems for robots with dozens of DOFs [20]. To compare the performance of MLDP with that of SBL, we used four 3D environments depicted in Fig. 7.

- The environment in Fig. 7(a) consists of a robot arm that has to maneuver inside a cluttered space. There are narrow passages connected by a wide-open region.
- In the environment shown in Fig. 7(b), the robot arm barely fits between adjacent bars of the cage. The robot must retract its arm, maneuver inside the cage, and get the arm out again through the ceiling of the cage.



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

Fig. 7. Test environments.

TABLE II

COMPARISON OF THE RUNNING TIMES OF SBL AND MLDP.

Env.	SBL (sec.)	MLDP (sec.)
7(a)	215	13
7(b)	5203	11
7(c)	> 100000	434
7(d)	> 100000	5850

TABLE III

RUNNING TIME STATISTICS OF MLDP.

Env. (Fig.)	Iter.	`Shrink` ($\times 10^{-4}$ sec.)	`SBL` (sec.)	`Repair` (sec.)	Total (sec.)
7(a)	2	1.8	10.0	3.3	13.3
7(b)	1	0.9	8.7	2.8	11.5
7(c)	3	1.1	428.6	5.7	434.3
7(d)	3	1.2	5703.4	145.9	5849.3

- The environment in Fig. 7(c) contains a long narrow passage that changes orientation several times. To solve the query, the robot must align horizontally the long bar held by the end-effector and navigate through the narrow space while keeping the bar horizontal.
- Fig. 7(d) shows the alpha puzzle, a benchmark test for PRM planners [1]. The goal is to separate two intertwined tubes. There are several versions of the problem, depending on the clearance between the tubes. The hardest is version 1.0, which is used in our experiments.

We tested MLDP and SBL on these four environments, using a PC with a 2.8 GHz processor and 512 MB of memory. The average running times of 20 runs are shown in Table II. The numbers do not consider tetrahedralization time.

Table II shows that MLDP drastically improves the performance of a single-query planner such as SBL, fully demonstrating the benefits of multi-level dilation. For instance, the improvement is more than three orders of magnitude for the environment in Fig. 7(b). For two environments, SBL was unable to find a path after a maximum number of iterations was reached. We indicated this by "> 100000".

C. Running times breakdown for MLDP

We measured the running times for the main functions in MLDP. The averages over 20 runs are shown in Table III. The table lists the number of iterations required by MLDP to find the solution and the time required by the functions `Shrink`, `SBL`, and `Repair`.

For the environment in Fig. 7(a), the configuration space contains some short narrow passages connected to a wide open region. The initial and goal configurations lie inside them. Two iterations of MLDP were enough to solve the query.

For the environment in Fig. 7(b), a solution path was found in the first iteration of MLDP ($s = 0.5$). Apparently, the milestones and edges of the path γ' found in the dilated free space $\mathcal{F}'$ are close to the surface of the obstacles, which allows `Repair` to quickly deform γ' and find a path in $\mathcal{F}$.

The environment in Fig. 7(c) is more complicated. A long narrow passage which changes orientation is created by defining a very small clearance between the upper and lower parts of the obstacles. If we shrink too much, we drastically change $\mathcal{F}$. Spurious passages are created and repairing a path going through them is very difficult. In a typical run of MLDP on this environment, it finds a path for $s = 0.5$, but fails to repair the path. The model at $s = 0.25$ is then computed, but `SBL` fails because of inadequate dilation of $\mathcal{F}$. A new shrunken version is computed for $s = 0.375$. In this case, both `SBL` and `Repair` succeed. The solution path followed by the robot's end-effector is illustrated schematically in 7(c).

Finally, the environment in Fig. 7(d) possesses an interesting property. To solve the query, tight coordination between translation and rotation is required. If we shrink too much, a simple translation will "solve" the problem, but repairing such a path would be almost impossible. If we shrink too little, the space is not dilated enough to help `SBL` find a path. MLDP took three iterations to solve the query.

These results show that multi-level dilation helps to improve the planner's performance. They also show the efficiency of our dilation method, which only adds small overhead to the total running time.

D. The importance of multi-level dilation

Free-space dilation improves the performance of PRM planners, provided that a suitable value of the dilation parameter is chosen. In this experiment, we fixed the value of the dilation parameter s at $0, 0.2, 0.4, \ldots, 1$ and run `SBL`. If a path is found, we then run `Repair` until a maximum number of trials is reached. For these parameter values, the running time of `SBL` ranges from roughly 10 seconds to more than 10,000

seconds, and the running time of Repair ranges from less than 1 second to roughly 1,000 seconds. For $s = 0$ or 0.2, SBL could not find a path and Repair was not run. For $s = 0.8$ or 1, SBL found a path in $\mathcal{F}$' but Repair could not fix it. Clearly, the bigger the dilation, the easier it is for SBL planner to find a solution, but the harder it is for Repair to succeed. Unfortunately, it is difficult to know in advance the right level of dilation. MLDP provides an automated way to address this problem.

VI. Conclusion and Future Work

We have introduced a new method for dilating the free space by shrinking the geometric models of robots and obstacles. It is efficient in both running time and memory usage. After preprocessing, computing a new shrunken model takes milliseconds on a desktop PC for a model with more than 100,000 triangles. Multiple shrunken models can be represented in the same data structure, which is also fully integrated with that for hierarchical collision checking, a key operation in PRM planning. The efficiency of the new method enables us to develop MLDP, a new PRM planner that quickly constructs a series of dilated free spaces and automatically determine the amount of dilation needed. Experiments show that both the dilation method and the planner work well in complex geometric environments with difficult narrow passages.

Of course, free-space dilation does not work for all geometric environments. Some environments cannot be shrunken effectively, *e.g.*, those with a cloud of points or thin bars as obstacles. However, in many practical problems, such as the one in Fig. 7(a), dilation is effective, and MLDP provides significant computational advantages.

During this work, we have observed that the maximum amount of shrinking possible on a model depends substantially on the tetrahedralization of the model. We are investigating this relationship and developing new tetrahedralization methods in order to increase the maximum amount of shrinking possible. In addition, our algorithm can shrink a model in a local region without affecting others. We plan to develop a new planner that uses this ability to determine which regions of a model cause a path to be in collision. This is important for mechanical assembly design, where one wishes to find out which parts of a model should be modified in order to enable certain paths.

Acknowledgements

D. Hsu's research is partially supported by a grant from the National University of Singapore. Jean-Claude Latombe's research is partially supported by NSF grants ACI-02-5671 and IIS-0412884.

References

[1] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," in *Robotics: The Algorithmic Perspective: 1998 Workshop on the Algorithmic Foundations of Robotics*, P. Agarwal *et al.*, Eds. Wellesley, MA: A. K. Peters, 1998, pp. 155–168.

[2] B. Baginski, "Local motion planning for manipulators based on shrinking and growing geometry models," in *Proc. IEEE Int. Conf. on Robotics & Automation*, 1996, pp. 3303–3308.

[3] V. Boor, M. Overmars, and F. van der Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners," in *Proc. IEEE Int. Conf. on Robotics & Automation*, 1999, pp. 1018–1023.

[4] H.-L. Cheng and T. Tan, "Approximating polygonal objects by deformable smooth surfaces," in *Mathematical Foundations of Computer Science*, 2005.

[5] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. The MIT Press, 2005, ch. 7.

[6] D. Cohen-Steiner, E. D. Verdiere, and M. Yvinec, "Conforming delaunay triangulations in 3d," in *ACM Symp. on Computational Geometry*, 2002, pp. 199–208.

[7] T. Dey and W. Zhao, "Approximate medial axis as a voronoi subcomplex," in *ACM Symp. on Solid Modeling & Applications*, 2002, pp. 356–366.

[8] M. Foskey, M. Garber, M. Lin, and D. Manocha, "A Voronoi-based hybrid motion planner," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2001, pp. 55–60.

[9] M. Foskey, M. Lin, and D. Manocha, "Efficient computation of a simplified medial axis," in *ACM Symp. on Solid Modeling & Applications*, 2003, pp. 96–107.

[10] L. Guibas, C. Holleman, and L. Kavraki, "A probabilistic roadmap planner for flexible objects with a workspace medial-axis based sampling approach," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 1999, pp. 254–260.

[11] D. Hsu, L. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin, "On finding narrow passages with probabilistic roadmap planners," in *Robotics: The Algorithmic Perspective: 1998 Workshop on the Algorithmic Foundations of Robotics*, P. Agarwal *et al.*, Eds. Wellesley, MA: A. K. Peters, 1998, pp. 141–154.

[12] D. Hsu, J. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," in *Proc. Int. Symp. on Robotics Research*, 2005.

[13] D. Hsu, J. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *Int. J. Computational Geometry & Applications*, vol. 9, no. 4-5, pp. 495–512, 1999.

[14] S. Jin, R. Lewis, and D. West, "A comparison of algorithms for vertex normal computation," *The Visual Computer*, vol. 21, no. 1-2, pp. 71–82, 2005.

[15] L. Kavraki, P. Švestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration space," *IEEE Trans. on Robotics & Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[16] H. Kurniawati and D. Hsu, "Workspace importance sampling for probabilistic roadmap planning," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2004, pp. 1618–1623.

[17] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robotics Research*, vol. 20, no. 5, pp. 278–400, 2001.

[18] M. Lin and D. Manocha, "Collision and proximity queries," in *Handbook of Discrete and Computational Geometry*, J. Goodman and J. O'Rourke, Eds. Chapman & Hall/CRC, 2004, pp. 787–807.

[19] M. Saha and J. Latombe, "Finding narrow passages with probabilistic roadmaps: The small step retraction method," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2005.

[20] G. Sánchez and J.-C. Latombe, "On delaying collision checking in PRM planning – application to multi-robot coordination," *Int. J. of Robotics Research*, vol. 21, no. 1, pp. 5–26, 2002.

[21] E. Sherbrooke, N. Patrikalakis, and E. Brisson, "Computation of the medial axis transform of 3-D polyhedra," in *Proc. ACM Symp. on Solid Modeling & Applicaitons*, 1995, pp. 187–200.

[22] H. Si, "TetGen, a quality tetrahedral mesh generator and three-dimensional Delaunay triangulator," Weierstrass Institute for Applied Analysis and Stochastics, Tech. Rep., 2004.

[23] T. Siméon, J. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *J. Advanced Robotics*, vol. 14, no. 6, pp. 477–494, 2000.

[24] J. van den Berg and M. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners," in *Proc. IEEE Int. Conf. on Robotics & Automation*, 2004, pp. 453–460,.

[25] Y. Yang and O. Brock, "Adapting the sampling distribution in PRM planners based on an approximated medial axis," in *Proc. IEEE Int. Conf. on Robotics & Automation*, 2004.