Finding Narrow Passages with Probabilistic Roadmaps: The Small Step Retraction Method

Mitul Saha and Jean-Claude Latombe

Artificial Intelligence Laboratory
Stanford University, Stanford, California, 94305, USA
{mitul, latombe}@cs.stanford.edu

***Abstract* - The efficiency of Probabilistic Roadmap (PRM) planners drops dramatically in spaces with narrow passages. This paper presents a new method – small-step retraction – that helps PRM planners find paths through such passages. The method consists of slightly fattening the robot's free space, constructing a roadmap in the fattened free space, and repairing colliding portions of this roadmap by retracting them out of collision. The fattened free space is not explicitly computed. Instead, the robot links and/or obstacles are thinned around their medial axis. A robot configuration lies in fattened free space if the thinned objects do not collide at this configuration. Two repair strategies are used. The "optimist" strategy waits until a complete path has been found in fattened free space before repairing it. The "pessimist" strategy repairs the roadmap as it is being built. The former is faster, but the latter is more reliable. A simple combination yields an integrated planner that is both fast and reliable.**

Index terms – Path planning, probabilistic roadmap, narrow passages, small-step retraction.

I. Introduction

Probabilistic Roadmaps (PRM) are an effective approach to plan collision-free paths for articulated robots and virtual characters in geometrically complex environments [1, 2, 8, 15, 16, 18, 24]. A PRM planner samples robot configurations at random with some probabilistic distribution and retains the collision-free ones as nodes – called *milestones* – of a graph (the *roadmap*). It connects each milestone to a small number of neighboring milestones by *local paths* (usually, straight-line segments in configuration space) and retains collision-free connections as the roadmap edges. Sampled configurations and connections between milestones are tested using an efficient collision checker [7, 9, 20, 21]. In this way, the PRM planner avoids the prohibitive cost of computing a complete representation of the robot's (collision-)free space F.

It has been proven that, under rather general assumptions on F's geometry, PRM planners converge quickly toward a solution path, if one exists [15, 17]. But this convergence is much slower when paths must go through narrow passages in F [14, 18, 25]. *Filtering* and *retraction* methods have been proposed to deal with this issue (Section II).

In this paper, we present a new retraction method, which we call *small-step retraction*, because it only tries to retract barely colliding configurations out of collision. It consists of (1) pre-computing "thinned" geometric models of the robot and/or the obstacles, (2) building a roadmap using the thinned models, and (3) repairing portions of this roadmap that are not in F by retracting them out of collision. Object thinning is done so that the free space associated with the thinned models – we call it the *fattened* free space F^* – is a superset of actual free space F (Section IV). Narrow passages in F get wider in F^*, making it easier to connect a roadmap through them (Section III). But it may also happen that F^* contains passages that were not in F – we call them *false* passages. Paths through such passages cannot be repaired. So, we propose two repair strategies (Section V). One – the "optimist" strategy – assumes that false passages will not be an issue. It postpones repairs until a complete path has been found in F^*; then, it repairs only the colliding sections of this path. It is usually very fast, but it fails when the path generated in F^* goes through a false passage. Fortunately, this rarely happens in practice. Instead, the "pessimist" strategy immediately repairs the milestones that are not in F. It is slower, but more reliable. Combining the two strategies yields an integrated planner – SSRP (for Small-Step Retraction Planner) – that is both fast and reliable.

This small-step retraction method has several advantages over previous retraction methods: (1) Most previous methods try to repair all colliding configurations, most of which are deeply buried into obstacle regions. This is computationally expensive. Only repairing those configurations where thinned objects do not overlap is much faster. (2) The method in [14] also performs a form of small-step retraction. But to test whether a colliding configuration should be repaired, it computes the penetration distance of the robot in the obstacles. This computation is notoriously expensive for non-convex objects [20]. (3) Our method works well even when there are no narrow passages in F. In contrast, previous retraction methods incur a significant overhead in running time. (4) While most previous narrow-passage methods were designed for multi-query PRM planners, ours works with single-query planners [15]. In practice, single-query planners are often much faster per query than multi-query ones.

II. Related Work

Neither the notion of a narrow passage, nor its impact on PRM planning is straightforward. In [17], the clearance of a path is used to estimate planning time. However, path clearance incompletely characterizes narrow passages. A more thorough analysis yields the notion of the *expansiveness* of free space F, which was used to provide tighter bounds on the convergence of a PRM planner [15].

Several milestones sampling strategies have been used to alleviate the narrow-passage issue. Two types of strategies – filtering and retraction strategies – explicitly address this issue.

Filtering strategies – e.g., Gaussian sampling [5] and bridge test [13] – over-sample configuration space and retain each configuration sampled in F as a milestone with a probability estimating its likelihood of being in a narrow passage. At each step, the Gaussian strategy first samples a configuration c from the configuration space uniformly at random and then a second configuration c' using a Gaussian distribution of mean c. If only one of these two configurations is in F, it is retained as a milestone; else both configurations are discarded. This strategy results in a greater density of milestones near F's boundary. The bridge-test strategy samples three configurations, instead of two, to target narrow passages more explicitly. Filtering strategies spend more time on average per milestone than more direct strategies. But they eventually produce a smaller and better distributed set of milestones, so that the planner later spends much less time connecting these milestones.

Retraction strategies exploit configurations sampled in collision space to guide the search for promising milestones. They consist of moving colliding configurations out of collision. One technique casts rays from a colliding configuration c and performs discrete walks along these rays until a configuration tests collision-free [3]. But in high-dimensional spaces, rays may easily miss narrow passages, especially if c is deeply buried in collision space. Other retraction techniques use the medial axis of the workspace or of the free space [10, 11, 12, 19, 25]. In general, existing retraction techniques are expensive because they retract all configurations sampled in collision space. Empirical observations (Section III) suggest that it is sufficient to only retract configurations near F's boundary, which is what our small-step retraction approach does. This same idea has been previously explored in [4, 14]. In [14] the penetration depth of the robot at a colliding configuration c in the obstacles was computed to decide whether it is worth to retract c into F. The high cost of this computation prevented this technique from being tested in geometrically complex environments. Instead, SSRP uses an efficient test to detect if a sampled configuration is barely colliding. Moreover, the twofold strategy of SSRP handles false passages efficiently and reliably.

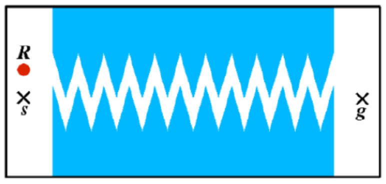
The idea of shrinking robot geometry was first introduced in [3], where a colliding path in configuration space is iteratively transformed into a collision-free path by growing back the robot links to their original geometry. However, this transform may get trapped into dead-ends. Also, the shrinking technique in [3] only applies to robot links modeled as rectangular boxes.

III. Preliminary Observations

Several components of our small-step retraction method were suggested by observations made during preliminary tests in 2-D configuration spaces. These observations were later confirmed by our experiments with SSRP (Section VI). For lack of space, we only report here on one key observation.

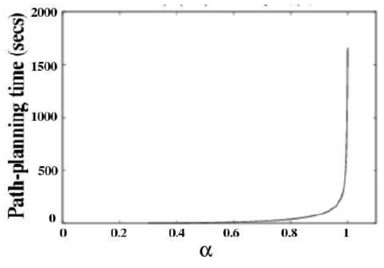
Consider the example in Fig. 1(a). The robot $\boldsymbol{R}$ is a disc (shown red) of radius r. Its configuration is defined by its center's coordinates. The start and goal configurations are s and g, respectively. The obstacles create a long and jaggy narrow passage of width w that the robot must traverse to reach g from s. The figure shows the robot's workspace, but the configuration space is also 2-D, with similar geometry. The passage's width in configuration space is w-2r.

We ran the PRM planner SBL described in [23] on this example, with decreasing values of r. Smaller values of r yield wider passages in configuration space. For each value of r, we ran SBL 100 times with different seeds of the random number generator. The table and corresponding plot in Fig. 1(b)-(c) shows the average planning times, for values of the ratio $\alpha = 2r/((1-\varepsilon)w)$ ranging between 0.3 and 1, where ε was set to 0.005. $\alpha = 1$ corresponds to the largest radius $r = (1-\varepsilon)w/2$, hence the narrowest passage in configuration space (with a width of $0.005\times w$). $\alpha = 0.3$ corresponds to the widest passage with a width of roughly $0.7\times w$. The average planning time decreases dramatically when α decreases from 1 to 0.95. For $\alpha = 0.987$ it is almost 38 times smaller than for $\alpha = 1$.



(a)

α	Path-planning time (secs)
1.000	1659.0
0.997	346.0
0.993	160.0
0.987	67.0
0.984	44.0
0.700	5.23
0.500	1.32
0.300	0.87



(b) (c)

Fig. 1: Planning a path through a narrow passage of varying width

This result is easy to explain: widening the narrow passage yields a large relative increase of its volume, so that the probability that each configuration sampled at random falls into the widened passages increases significantly. This observation is at the core of the small-step retraction strategy. Fattening free space by only a small amount – which is achieved by thinning the robot and/or the obstacles – greatly simplifies the task of a PRM planner. Of course, a path in the fattened free space F^* may not fully lie in the actual free space F. The purpose of small-step retraction is to repair the sections that are not in F.

Other tests show that, in the presence of multiple narrow passages, SBL generates paths through the "easiest" one with much higher frequency. We use this observation to design the overall strategy of SSRP. As mentioned in the introduction, F^* may contain false narrow passages, but these passages are often narrower than the widened true passages. Hence, in most cases, a path in F^* will go through the true passages; so, it will be repairable into a path lying in F. The optimist strategy of SSRP makes this assumption, but the pessimist strategy does not. Since the former is usually orders of magnitude faster than the latter, SSRP runs it first, and uses the pessimist strategy only as a back-up.

IV. Object Thinning

The purpose of object thinning is to construct a version of the robot and obstacle geometry that the planner's collision checker can use to test if a configuration c is barely colliding or not.

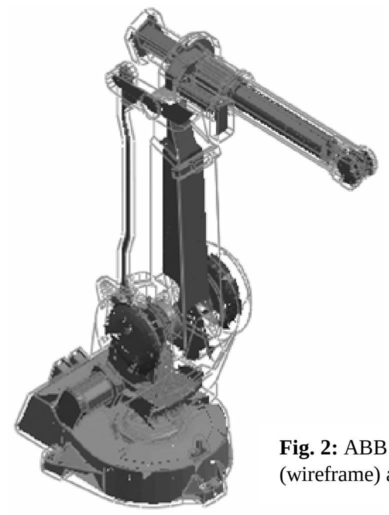


Fig. 2: ABB IRB-2400 robot: Original (wireframe) and thinned (shaded) geometry

In SSRP, object thinning is a pre-computation step that is done at most once for each object, prior to performing any planning operation. Various thinning techniques could be used. The only requirement is that the space occupied by a thinned object must be contained in the space occupied by the original object. So, if $\boldsymbol{R}(c)$ and $\boldsymbol{R}^*(c)$ respectively stand for the space occupied by the original robot and the thinned robot at configuration c, then we must have $\boldsymbol{R}^*(c) \subset \boldsymbol{R}(c)$. Note that many straightforward scaling methods do not guarantee such a relation for non-convex objects.

SSRP thins a rigid object (e.g., a robot link, an obstacle) by first computing this object's medial axis (MA) and then shifting the object's boundary inward toward the MA [22]. The MA is computed using an efficient technique that makes use of graphics hardware to speedup basic operations [6]. To thin a robot, we thin each link separately, but leave the kinematics (i.e, the relative positions of the joints) unchanged. Fig.2 shows both the original geometry of the ABB IRB-2400 robot (wireframe) and its thinned geometry (shaded)

Experiments with SSRP show that it is often sufficient to thin only the robot (or the obstacles) to achieve major planning speed-up.

V. Small-Step Retraction Planner

SSRP uses the pre-existing SBL planner [23] as a building block, but we could have used almost any other single-query PRM planner as well. SSRP combines two algorithms (each calling SBL). OPTIMIST, which is very fast, but not fully reliable, is invoked first. If it fails to find a path, then PESSIMIST, a slower but more reliable algorithm, is invoked.

A. OPTIMIST

OPTIMIST uses SBL to construct a roadmap in F^*. It assumes that paths generated by SBL traverse true passages (see Section I), hence can be easily repaired. So, only after a complete path has been found in F^* between the query configurations s and g does OPTIMIST try to repair this path by retracting the colliding sections into F. If it does not quickly repair the path, it returns failure. The algorithm is given below, where M refers to the geometric models of the robot and the obstacles and M^* to their thinned models:

```
Algorithm OPTIMIST($s, g, M, M^*$)
    1. $\tau^* \leftarrow$ SBL($s, g, M^*$)
    2. Return Repair($\tau^*, M$)
```

At Step 1, SBL constructs a path τ^* in F^*. Step 2 calls Repair to repair the sections of τ^* that are in $F^* \backslash F$. The Repair algorithm returns either a path entirely in F, or failure. It first retracts the milestones on τ^* out of collision, then its edges. The following algorithm, Repair-Conf, is used to repair every milestone c on τ^* that lies in $F^* \backslash F$ by sampling configurations inside balls $B(c,\rho)$ of increasing radii ρ centered at c, starting with some small given radius ρ_{min} (the factor $\eta > 1$ used to increase ρ is an input constant):

```
Algorithm Repair-Conf($c, M$)
    1. $\rho \leftarrow \rho_{min}$
    2. For $k = 1, \ldots, K$ do
        2.1. Sample a configuration $c'$ uniformly at random
             from $B(c, \rho)$
        2.2. If $c' \in F$ then return $c'$ else $\rho \leftarrow \rho \times \eta$
    3. Return failure
```

If Repair-Conf fails after K iterations, the milestone c and the path τ^* are considered non-repairable. Else, the milestone c on τ^* is replaced by the returned configuration c'. Once all milestones on τ^* have been repaired, every edge e on τ^* that intersects $F^* \backslash F$ is repaired. This is done recursively by calling Repair-Conf(c,M) with c set to the mid-point of e. Any failure causes OPTIMIST to fail. Because repairable paths in F^* usually lie close to the boundary of F, the parameter K is set relatively small (100, in our implementation) to avoid wasting time on non-repairable paths.

B. PESSISMIST

SBL(s,g,M^*) sometimes constructs paths through false passages, leading OPTIMIST to fail. To avoid encountering the same kind of a failure, PESSIMIST immediately repairs every configuration sampled in $F^*\backslash F$, instead of waiting until a complete path in F^* has been found. This is done by modifying SBL slightly. Within PESSIMIST, each time SBL samples a new configuration c, it proceeds as follows:

1. If $c \in F$, then add c to the set of milestones
2. Else if $c \in F^*\backslash F$ and Repair-Conf(c, M) returns a configuration c' then add c' to the set of milestones
3. Else discard c

PESSIMIST does not repair edges that do not fully lie in F. It lets SBL continue sampling more configurations, until a path in F is found or a timeout condition is reached.

For every configuration c sampled outside F, PESSIMIST calls the collision checker twice (once to detect that c is not in F, and once to check if it is in F^*). It tries to repair all configurations sampled in $F^*\backslash F$, although relatively few repaired configurations will end up being on a final path. Hence, on average, PESSIMIST spends more time per milestone than either OPTIMIST or plain SBL. But, by repairing milestones immediately, it avoids the fatal mistakes of OPTIMIST. Moreover, because its sampling strategy yields better distributions of milestones than plain SBL, it usually needs smaller roadmaps to generate paths than plain SBL.

C. Overall Planner

As the experiments reported in Section VI will show, when OPTIMIST succeeds, it is much faster than PESSIMIST (often by very large factors). But PESSIMIST is more reliable and in general still significantly faster than plain SBL. These results led us to design SSRP as follows:

```
Algorithm SSRP($s, g, M, M^*$)
    1. Repeat $N$ times:
          If OPTIMIST($s, g, M, M^*$) returns a path
             then return this path
    2. Return PESSIMIST($s, g, M, M^*$)
```

Our experiments show that if OPTIMIST fails a few times in a row, then it continues failing consistently. So, N is set small (5, in our implementation). Since OPTIMIST is much faster than PESSIMIST, the impact of successive failures of OPTIMIST on the total running time of SSRP is small.

VI. Experimental Results

SSRP is written in C++ on top of the PRM planner SBL available at http://robotics.stanford.edu/~mitul/mpk. All results reported below were obtained on a 1GHz Pentium III PC with 1GB RAM.

A. Test environments

Fig. 3 shows 7 environments with difficult narrow passages. In (a)-(b) the robot move among thin obstacles. In (b), it is in a bar cage and must find a way for its endpoint in and out of the cage. In (c) the robot must move from an initial configuration where the end-effector is outside a car body to a goal configuration where it is deep inside the car. The only way is through the door window. In (d), the robot's end-effector forms a close loop that must be taken out of one of the tall vertical poles into the other pole. Here, F contains two narrow passages separated by a large open area. Environment (e) is inspired by spot welding tasks. The welding tool must be inserted on both sides of a flat object. The toy example (f) is designed to create a short false passage in F^*. Finally, the so-called alpha puzzle in (g) is a classical test for PRM planners [2]. One object is arbitrarily chosen to be free-flying and the other to be a fixed obstacle. Object geometry and small clearance create a very narrow passage in F that can only be traversed by a tight coordination of rotation and translation. The alpha puzzle exists in versions named 1.0 to 1.5. We used the 1.0 and 1.1 versions, which yield the narrowest passages.

In examples (a), (b), (c), (f) and (g), we only thinned the robot, while in examples (d) and (e), we only thinned the obstacles, because they have much simpler geometry. (Tests show that in all these examples thinning both the robot and the obstacles results in no significant additional planning speedup.) Thinning is considered a pre-computation step; its running time is not included in the planning times below.

Fig. 4 shows two additional test environments (h) and (i) for which there is no narrow passage in free space. They were also used to evaluate our planner, since many practical planning problems do not involve difficult narrow passages.

Table I gives the number of polygons in the meshes describing the objects in all environments, except (f).

B. Results

Table II lists the running times of OPTIMIST, PESSIMIST, SSRP, and plain SBL on problems in the environments of Fig. 3. Each time is expressed in seconds and is an average over 100 runs on the same pair of query configurations, except for SBL in example (g) and Pessimist in example (g)-1.0, which found no path in several runs, some taking more than 30 hours of computation (which we indicate with ">100000").

On problems (a)-(e) OPTIMIST succeeded consistently, but on the last two it failed consistently. The columns $T_{OPTIMIST}$ and $T_{PESSIMIST}$ give the average running times of OPTIMIST and PESSIMIST (for better comparison, we ran PESSIMIST even when OPTIMIST succeeded). Except for (g)-1.0, PESSIMIST never failed. The column T_{SSRP} indicates the total planning time of SSRP, equal to $T_{OPTIMIST}$ in examples (a)-(e) and (g)-1.0 and to $5\times T_{OPTIMIST}+T_{PESSIMIST}$ in examples (f) and (g)-1.1. The last column gives the average time of plain SBL, which succeeded consistently on all problems, but the last two.

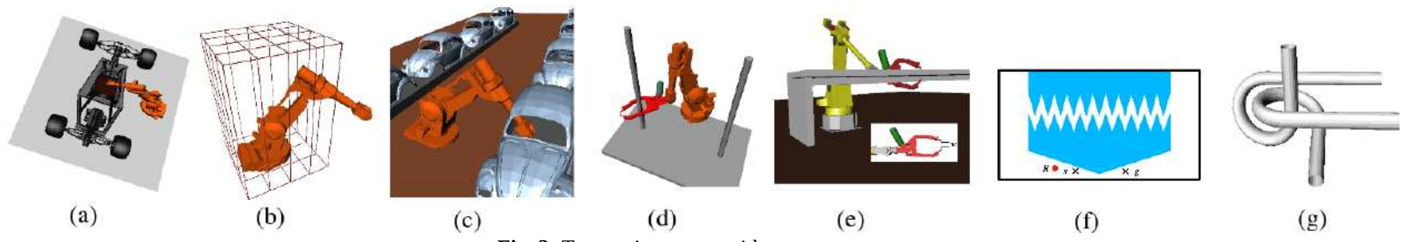


Fig. 3: Test environments with narrow passages

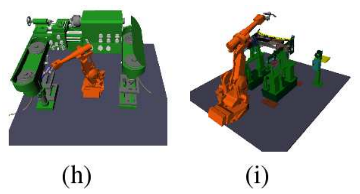


Fig. 4: Test environments without narrow passage

TABLE I: Number of polygons in models

	(a)	(b)	(c)	(d)	(e)	(g)	(h)	(i)
robot	4053	4053	4053	7596	4505	1008	4853	4853
thinned robot	23585	23585	23585	7596	4505	8976	24385	24385
obstacles	43530	432	4028	140	48	1008	74681	83934
thinned obstacles	43530	432	4028	286	80	1008	74681	83934

TABLE II**:** Performance of OPTIMIST, PESSIMIST, SSRP, and plain SBL in the environments of Fig. 3

	% success OPTIMIST	$T_{OPTIMIST}$	$T_{PESSIMIST}$	T_{SSRP}	T_{SBL}
(a)	100	9.4	1284	9.4	12295
(b)	100	32	1040	32	5955
(c)	100	2.1	15	2.1	41
(d)	100	492	634	492	863
(e)	100	65	351	65	631
(f)	0	12	325	386	572
(g)-1.1	0	111	2810	3365	>100000
(g)-1.0	100	14069	>100000	14069	>100000

TABLE III: Average number of milestones generated by OPTIMIST, PESSIMIST, and plain SBL in the environments (a) through (d) of Fig. 3

	OPTIMIST	PESSIMIST	SBL
(a)	3202	95559	250515
(b)	20581	76356	132451
(c)	3023	16025	30154
(d)	81015	118829	149359

Table III lists the average numbers of milestones in the roadmaps generated by OPTIMIST, PESSIMIST and plain SBL over 100 runs, for the examples (a) through (d) of Fig. 3.

Finally, Table IV lists the same results as in Fig. 11, but for the two examples of Fig. 4. In both cases, Optimist, Pessimist, and plain SBL succeeded in all runs.

TABLE IV: Performance of OPTIMIST, PESSIMIST, SSRP, and plain SBL in the environments of Fig. 4

	$T_{OPTIMIST}$	$T_{PESSIMIST}$	T_{SSRP}	T_{SBL}
(h)	1.68	1.59	1.68	1.60
(i)	2.59	2.16	2.59	2.40

C. Discussion

The running times of SBL in examples (h)-(i) are much smaller than in examples (a)-(g). This gives a good indication of the difficulty of finding paths through narrow passages in examples (a)-(g).

In all examples (a)-(g), OPTIMIST is much faster than PESSIMIST, so it is worth running it a few times before calling PESSIMIST. Even when OPTIMIST fails in examples (f) and (g)-1.1, the impact on the running time of SSRP is small. In examples without narrow passages (h) and (i), OPTIMIST is only as fast as PESSIMIST, but succeeds consistently.

In every example, OPTIMIST succeeds or fails consistently.

PESSIMIST alone is significantly faster than SBL in examples with narrow passages, because repairing configurations sampled in *F***F* leads to placing milestones in narrow passages with higher probability. Hence, PESSIMIST finds paths with smaller roadmaps as shown in Table III.

On average across various problems with narrow passages, SSRP is faster than PESSIMIST alone, because OPTIMIST often succeeds and costs little when it fails. SSRP is much faster – often by more than one order of magnitude – than plain SBL alone and more reliable.

The results in Table IV show that on the examples without narrow passages (h)-(i) SSRP is as fast as plain SBL. In fact, in those examples, OPTIMIST, PESSIMIST, SSRP, and plain SBL take about the same amount of time.

Recall that in (g)-1.1, the clearance between the two parts is greater than in (g)-1.0. In both cases, we used the same thinned objects. In (g)-1.1, thinning deforms the narrow passage greatly, to the extent that rotation is almost no longer needed. Thus, OPTIMIST fails to retract many configurations sampled in the deformed passage into the true passage of *F*. The deformation of the narrow passage in (g)-1.0 is less dramatic and OPTIMIST can repair configurations sampled in *F***F*. But, in this example, paths in *F* consists of many tiny segments, so that PESSIMIST, which only repairs configurations sampled in *F***F* (and not connections), fails to find a solution path in reasonable time.

VII. CONCLUSION

The main contribution of this paper is a new retraction method – small-step retraction – that helps PRM planners find paths through narrow passages. The core idea underlying this method is that retracting sampled configurations that are barely colliding is sufficient to speed up the generation of roadmap milestones within narrow passages. Barely colliding configurations are relatively easy to retract out of collision. In addition, they can be efficiently identified by running a classical collision checker on thinned geometric models pre-computed using a medial-axis-based technique. A slight drawback of the method is that it increases pre-computation time. Another idea of our work is the combination of two algorithms, OPTIMIST and PESSIMIST, which complement each other well.

The entire method was implemented as a new PRM planner – SSRP – that extends the pre-existing SBL planner. SSRP was tested on many examples. Results show that on examples with narrow passages it is significantly faster, and more reliable, than SBL, while on examples without narrow passages, it is as fast as SBL. The design of SSRP has been mostly guided by empirical observations. Future work should try to extend formal analysis of PRM planners – for instance, expansiveness [15] – to better understand the role of small-step retraction and its possible shortcomings.

ACKNOWLEDGMENT

This work was partially supported by grants from ABB and General Motors. We thank Y.C. Chang, F. Prinz, F. Schwarzer, and P. Isto for their insightful comments, and Dr. Z. Gan at ABB Automation for providing us a CAD model of the IRB-2400 robot.

REFERENCES

[1] M. Akinc, K.E. Bekris, B.Y. Chen, A.M. Ladd, E. Plaku, and L.E. Kavraki. Probabilistic Roadmaps of Trees for Parallel Computation of Multiple Query Roadmaps. Proc *11th Int. Symp. Robotics Research*, Siena, Italy, Oct. 19-22, 2003.

[2] N.M. Amato, O.B. Bayazit, L.K. Dale, C. Jones and D. Vallejo. OBPRM: An Obstacle-Based PRM for 3D Workspace. In P.K. Agarwal et al. (eds.), *Robotics: The Algorithmic Perspective*, A K Peters, Natick, MA, pp. 155-168, 1998.

[3] B. Baginski. Local Motion Planning for Manipulators Based on Shrinking and Growing Geometry Models. *Proc. IEEE Int. Conf. Robotics and Automation*, Minneapolis, MN, pp. 3303-3308, 1997.

[4] O.B. Bayazit. *Solving Motion Planning Problems by Iterative Relaxation of Constraints.* Ph.D. Thesis, Dept. of Computer Science, Texas A&M University, College Station, Texas, U.S.A., May 2003.

[5] V. Boor, M.H. Overmars, and A.F. van der Strappen. The Gaussian Sampling Strategy for Probabilistic Roadmap Planners. *Proc. IEEE Int. Conf. on Robotics and Automation*, Detroit, MI, pp. 1018-1023, 1999.

[6] Y.C. Chang, J. M. Pinilla, K. Ramaswami, and F. Prinz. Near Parallel Computation of MAT of 3D Polyhedra. In preparation, 2005.

[7] J. Cohen, M. Lin, D. Manocha, and M. Ponamgi. I-Collide: An Interactive and Exact Collision Detection System for Large Scale Environments. *Proc. ACM Interactive 3D Graphics Conf.*, pp. 189-196, 1995.

[8] J. Cortés, T. Siméon, and J.P. Laumond. A Random Loop Generator for Planning the Motions of Closed Kinematic Chains Using PRM Methods. *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 2141-2146, 2002.

[9] S. Gottschalk, M. Lin, and D. Manocha. OBB-Tree: A Hierarchical Structure for Rapid Interference Detection. *Proc. ACM SIGGRAPH'96*, pp. 171-180, 1996.

[10] L. Guibas, C. Holleman, and L. Kavraki. A Probabilistic Roadmap Planner for Flexible Objects with a Workspace Medial-Axis Based Sampling Approach. *Proc. IEEE Int. Conf. Intelligent Robots and Systems*, 1999.

[11] K.E. Hoff III, T. Culver, J. Keyser, M. Lin, and D. Manocha. Interactive Motion Planning Using Hardware-Accelerated Computation of Generalized Voronoi Diagrams. *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, April 2000.

[12] C. Holleman and L. Kavraki. A Framework for Using the Workspace Medial Axis in PRM Planners, *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, pp. 1408-1413, April 2000.

[13] D. Hsu, T. Jiang, J. Reif, and Z. Sun. The Bridge Test for Sampling Narrow Passages with Probabilistic Roadmap Planners. *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 4420-4426, 2003.

[14] D. Hsu, L. Kavraki, J.C. Latombe, R. Motwani, and S. Sorkin. On Finding Narrow Passages with Probabilistic Roadmap Planners. In P.K. Agarwal et al. (eds.), *Robotics: The Algorithmic Perspective*, A K Peters, Natick, MA, pp. 151-153, 1998.

[15] D. Hsu, J.C. Latombe and R. Motwani. Path Planning in Expansive Configuration Spaces. *Int. J. Comp. Geometry and Applications*, 9(4-5):495-512, 1999.

[16] P. Isto, Constructing Probabilistic Roadmaps with Powerful Local Planning and Path Optimization, *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 2323-2328, 2002.

[17] L.E. Kavraki, M. Kolountzakis, and J.C. Latombe. Analysis of Probabilistic Roadmaps for Path Planning. *IEEE Tr. Robotics and Automation*, 14(1):166-171, Feb.1998.

[18] L.E. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmars. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces, *IEEE Tr. Robotics and Automation*, 12(4):566-580, 1996.

[19] J. M. Lien, S. L. Thomas, and N. M. Amato. A General Framework for Sampling on the Medial Axis of the Free Space. *Proc. IEEE Int. Conf. Robotics and Automation*, 2003.

[20] M. Lin and D. Manocha. Collision and Proximity Queries. In J.E. Goodman and J. O'Rourke (eds.), *Handbook of Discrete and Computational Geometry*, 2nd edition, Chapter 35, pp. 787-807, Chapman&Hall/CRC, 2004.

[21] S. Quinlan. Efficient Distance Computation Between Non-Convex Objects. *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 3324-3329, 1994.

[22] M. Saha, J.C. Latombe, Y.C. Chang, F. Prinz. Finding Narrow Passages with Probabilistic Roadmaps: the Small-Step Retraction Method. *Manuscript* submitted for publication, 2004.

[23] G. Sánchez-Ante and J.C. Latombe. A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking. *Int. J. Robotics Research*, 21(1):5-26, Jan. 2002.

[24] K. Yamane, J.J. Kuffner, and J.K. Hodgins. Synthesizing Animations of Human Manipulation Tasks. *Proc. SIGGRAPH'04*, 2004.

[25] S.A. Wilmarth, N.M. Amato, and P.F. Stiller. MAPRM: A Probabilistic Roadmap Planner with Sampling on the Medial Axis of the Free Space. *Proc. IEEE Int. Conf. Robotics and Automation* Detroit, MI, pp. 1024-1031, 1999.