On Improving the Clearance for Robots in High-Dimensional Configuration Spaces*

Roland Geraerts and Mark H. Overmars
Institute of Information and Computing Sciences
Utrecht University
3584 CH Utrecht, the Netherlands
{*roland,markov*}@*cs.uu.nl*

***Abstract*— In robot motion planning, many algorithms have been proposed that create a path for a robot in an environment with obstacles. Since it can be hard to create such paths, most algorithms are aimed at finding *a* solution. However, for most applications the paths must be of a good quality as well. That is, paths should preferably be short, be smooth, and should preserve some clearance from the obstacles. In this paper, we focus on improving the clearance of paths. Existing methods only extract high clearance paths for rigid, translating bodies. We present an algorithm that improves the clearance along paths of a broader range of robots which may reside in arbitrary high-dimensional configuration spaces. Examples include planar, free-flying and articulated robots.**

***Index Terms*— motion planning, path optimization**

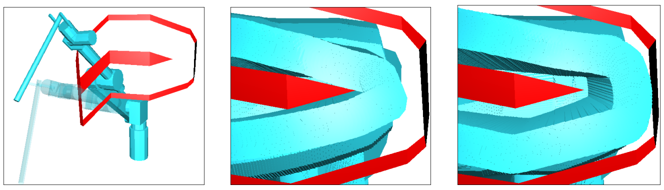

(a) Articulated robot (b) Initial path (c) Optimized path

Fig. 1. Improving the clearance along a path traversed by an articulated robot with six degrees of freedom. Fig. (a) shows the start and (blurred) goal placement of the robot arm. In (b), we zoom in on the initial path. The swept volume of the significant parts of the robot is shown. As can be seen, the clearance along this path is very small. Our new algorithm successfully increases the clearance along the path which is visualized in (c).

I. Introduction

Motion planning is one of the fundamental problems in robotics. The motion planning problem can be defined as finding a path between a start and goal placement of a robot in an environment with obstacles. The last decade, efficient algorithms have been devised to tackle this problem. They are successfully applied in fields such as mobile robots, manipulation planning, CAD systems, virtual environments, protein folding and human robot planning. See e.g. the books of Latombe [1] and Lavalle [2] for many interesting results.

One important aspect of motion planning is the quality of the path. The quality is often measured in terms of length and clearance. Many applications require a short path since redundant motions will take longer to execute. In practical situations, the path often has to keep some minimum amount of clearance to the obstacles because it can be difficult to measure and control the precise position of a robot. Traveling along a path with a certain amount of minimum clearance reduces the chances of collisions due to these uncertainties. In a nuclear power plant for example, it may be desirable to minimize the risk of heat or radioactive contamination [3]. In this type of environment, calculating a safe path (off-line) is more significant than calculating a low quality path on-line. Clearance is used for other purposes as well: in [4] for instance, it is used to guide multiple units in a virtual environment.

Indeed, enlarging the clearance is important in various applications. It is though far from trivial how to do this. Previous work limited their efforts to rigid, translating bodies. In this paper, we present a new algorithm that improves the clearance along paths for a broader range of robots including planar, free-flying and articulated robots, see Fig. 1.

A. Related work

Many motion planning algorithms create a roadmap (or graph) which represents collision-free motions that can be made by the moving object in an environment with obstacles. From this graph a path is obtained by a Dijkstra's shortest path query. Since these calculations can be performed off-line, we refer to this stage as preprocessing. The paths usually are optimized in a post-processing stage.

In [5], an augmented version of Dijkstra's algorithm is used to extract a path based on other criteria than length. The minimum clearance along the path is maximized by incorporating a higher cost for edges that represent a small amount of clearance. Unfortunately, the paths rarely provide optimal solutions because they are restricted to the randomly generated nodes in the roadmap. Even if the nodes are placed on the medial axis [6], the edges will in general not lie on the medial axis, and hence the extracted paths do not have a guaranteed amount of clearance.

*This research was supported by the Dutch Organization for Scientific Research (N.W.O.). This research was also supported by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-2001-39250 (MOVIE - Motion Planning in Virtual Environments).

Another category of algorithms create a path along the Generalized Voronoi Diagram (GVD). The GVD (or medial axis) for a robot with n degrees of freedom (DOFs) is defined as the collection of k-dimensional geometric features $(0 \leq k < n)$ which are in general $(n+1-k)$-equidistant to the obstacles. As an example, consider Fig. 2(a) that shows the outline of a solid bounding box and a part of the medial axis for a translating robot. The medial axis of this robot consists of a collection of surfaces, curves and points. The surfaces are defined by the locus of 2-equidistant closest points to the bounding box. The curves have 3-equidistant closest points and the points have 4-equidistant closest points to the bounding box. These features are connected if the free space in which the robot operates is also connected [7]. Hence, the GVD is a complete representation for motion planning purposes. Most importantly, paths on the GVD have appealing properties such as large clearance from obstacles.

Unfortunately, an exact computation of the GVD is not practical for problems involving many degrees of freedom (DOFs) and many obstacles as this requires the expensive and intricate computation of the configuration space obstacles. Therefore, the GVD is approximated in practice.

In [8], the GVD is approximated by applying spatial subdivision and isosurface extraction techniques. Although the calculations are easy, robust and can be generalized to higher dimensions, the technique only works for disjoint convex sites and consumes an exponential amount of memory which makes this technique impractical for problems involving many DOFs. Another approach incrementally constructs the GVD by finding the maximal inscribed disks in a two dimensional discretized workspace [9]. While this algorithm is also extensible to handle higher-dimensional problems, it suffers from the same drawback as the preceding algorithm. In [10], a technique is described that exploits the fast computation of a GVD using graphics hardware for motion planning in complex static and dynamic environments. The technique is though limited to a three-dimensional workspace for rigid translating robots.

The above preprocessing methods concentrate on creating a data structure from which paths can be extracted. In [11], the authors proposed a post-processing algorithm that adds clearance to a path by retracting it to the medial axis of the workspace. A path Π is divided into n configurations. Such a configuration corresponds to a particular placement of the robot in the environment. Each configuration is retracted to the medial axis (except for the start and goal configuration) as follows. First, the closest point (on the obstacles) between the robot and the obstacles is calculated. Then the robot is iteratively moved away from this point until the robot has 2-equidistant nearest points to the obstacles.

This method provides a fast and accurate retraction of possibly non-convex rigid bodies to the medial axis in two- and three-dimensional spaces. Unfortunately, as the retraction

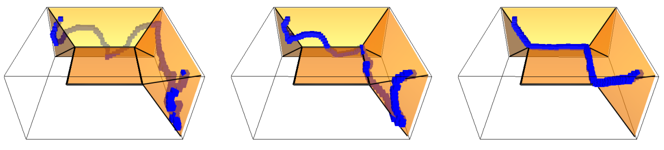

(a) Initial path (b) Retracted path (c) Optimal path

Fig. 2. Retraction of a path in an environment that only consists of a solid bounding box. A part of the medial of this box is shown. Fig. (a) shows the initial path. This path is retracted to the medial axis by the algorithm from [11] in (b). Fig. (c) shows a path having the optimal amount of clearance. This path was obtained by our new algorithm.

is performed by a series of translations of the robot, the method is not suitable for increasing the clearance along a path traversed by an articulated robot or a free-flying robot for which the rotational DOFs are more important than the translational ones. In addition, the method will in general not produce a maximal clearance path because the retraction is completed when the configurations on the path are placed somewhere on the medial axis. A configuration could have a higher clearance if it was retracted to a *ridge* of the medial axis. We define a ridge as the locus of points in configuration space $\mathcal{C}$ that represents a locally maximum clearance. See Fig. 2 for an example. The crooked path from Fig. 2(a) was retracted to the medial axis by the algorithm from [11]. Fig. 2(b) shows that the retracted configurations sway on the medial axis surfaces. In Fig. 2(c), the path is retracted toward the ridges, resulting in an optimal clearance path.

In conclusion, there exists no efficient algorithm that can compute high-clearance paths for robots residing in high-dimensional configuration spaces. We will propose an algorithm that efficiently retracts a path toward the ridges of the medial axis for robots with an arbitrary number of DOFs. This algorithm may be applied in high-cost environments such as a factory in which a manipulator arm operates.

B. *Paper outline*

This paper is organized as follows: in section II, we will introduce our retraction algorithm. We elaborate on the algorithmic details in section III and show that the algorithm is correct. Then, we apply the algorithm on problems involving a planar, a free-flying and a 6-DOF articulated robot in section IV. Finally, we conclude in section V that our method is successfully able to improve the clearance along paths traversed by a broad range of robots.

II. Retraction Algorithm

The retraction algorithm tries to iteratively increase the clearance of the configurations on the path by moving them in a direction for which the clearance is higher. We define clearance as the Euclidean distance between the pair of closest points on the robot and obstacles.

We say that two configurations are adjacent if the distance between them is at most a predetermined step size. This step size is dependent on the robot and obstacles. Now let a path Π be represented as a series of adjacent configurations $\Pi_0, \cdots, \Pi_{n-1}$. Our problem now is as follows. Convert a given path Π into a path Π' such that for each Π'_i the clearance is either locally maximal or is larger than a given constant c_{min}. Our solution consists of a number of iterations. In each iteration, we choose a random direction dir^+. Then, for each configuration Π_i, we count the number of steps the robot can make in direction dir^+ before it hits an obstacle. We refer to this as the *directional clearance*. We also calculate the directional clearance for the opposite direction dir^-. The robot is moved in the direction that results in the largest directional clearance if this update leads to a larger clearance. If the directional clearance is equal in both directions, then we do not move the robot, i.e. we do not update configuration Π_i. We stop retracting the path when the path has c_{min} clearance or when the clearance does not improve anymore.

By updating the configurations, the path is forced to stretch and shrink during the retraction which causes the following two problems. First, the distance between two adjacent configurations in the path can become *larger* than the maximum step size. This happens for example when the path is pushed away from the obstacles. If this occurs, we insert some extra configurations in between them. Second, multiple configurations can be mapped to a small region by which the distance between two non-adjacent configurations is *smaller* than the step size. This occurs for example when pieces of the path are traversed twice. As we will see in the following section, these can easily be removed.

An impression of a retraction is visualized in Fig. 3. This figure shows an initial and a final path traversed by a square robot in a simple two-dimensional workspace. The lines between them are the guided random walks of the configurations. We call these walks *guided* because a configuration is updated only if its clearance increases. Note that at some places, extra configurations were inserted while at other places configurations were removed. After 40 iterations, the initial path was successfully retracted to the medial axis, resulting in a large clearance path. While this example shows a retraction for a robot with only two DOFs, the retraction can also be applied to robots with more DOFs such as an articulated robot with six joints.

In section III, we will elaborate on the most important algorithmic details.

III. Algorithmic Details

A path consists of a series of configurations. We require that the distance between each pair of adjacent configurations is at most $step$. Hence, we need a distance metric which will be described in section III-A. The clearance of the configurations is improved by moving them in a random direction. In section III-B, we show how to compute such a direction. Furthermore, we show how to move the robot in this direction. We also need to compute the directional clearance of a configuration which will be described in section III-C. After an iteration of the algorithm, the distance between two adjacent configurations can change. To keep a valid path, we need to insert and delete appropriate configurations. We elaborate on this in section III-D. Finally, we show in section III-E that the algorithm indeed retracts a path toward the ridges of the medial axis.

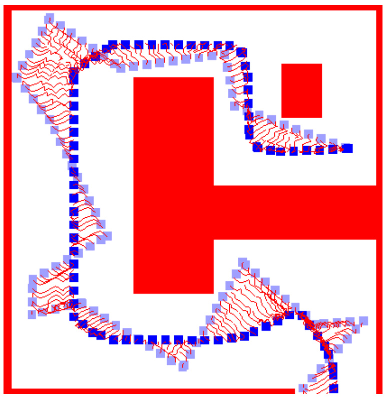


Fig. 3. An impression of the retraction algorithm. The algorithm retracts the initial path (traversed by a square robot) to the medial axis. For each configuration on the path, the guided random walk (small curve) is drawn.

A. Distance metric

The importance of choosing a good distance metric is discussed in [12]. Such a metric often incorporates weights which can be used to control the relative importance of the DOFs of the robot. We distinguish three types of DOFS: translation, rotation$_1$ (rotation about the x-, y-, or z-axis) and rotation$_3$ (rotation in SO3). For example, a free-flying robot can be described by three translational DOFs and one rotational$_3$ DOF, and an articulated robot with six joints may be described by six rotational$_1$ DOFs.

We calculate the distance between two configurations q and r by summing the weighted partial distances for each DOF $0 \leq i < n$ that describes the configurations, i.e.

$$d(q, r) = \sqrt{\sum_{i=0}^{n-1} [w_i d(q_i, r_i)]^2}.$$

The calculation of distance $d(q_i, r_i)$ is dependent on the type of the DOF. For translation, we set $d(t', t'')$ to $|t' - t''|$. We split the calculation for a rotational$_1$ DOF in two parts: if the range is smaller than 2π, which occurs often for revolution joints in manipulator arms, we take the same distance measure as for a translational DOF. If the rotational DOF is periodic, i.e. the orientation at 0 radians equals the orientation at 2π radians, we take the smallest angle. More formally, we set $d(r', r'')$ to $|r' - r''|$ if r is not periodic, otherwise $d(r', r'') = \min\{|r'-r''|, r'-r''+2\pi, r''-r'+2\pi\}$.

We use unit quaternions to represent rotations in 3D. The distance between two quaternions r' and r'' can be calculated by taking the shortest angle over a 4D-sphere, i.e. $d(r', r'') = \min\{2\arccos(r' \cdot r''), 2\pi - 2\arccos(r' \cdot r'')\}$. We do not use three Euler angles to represent an orientation because this representation has several drawbacks quaternions do not have, such as the 'gimbal lock' and the difficulty to define proper methods for sampling, interpolation and distance metric [13].

B. Random direction vector

We need to create a random direction q' such that the distance from configuration q to $q \oplus q'$ equals $stepsize$. We set $stepsize$ to $\frac{2}{3}step$ as this is needed in the proof that the algorithm is correct. The direction q' is composed of elements from $\{q'_{tra}, q'_{rot_1}, q'_{rot_3}\}$ such that $\sqrt{\sum_{i=0}^{n-1} p_i^2} = stepsize$, where $p_i = w_i d(q_i, q_i + q'_i)$. We choose a random value rnd_i between 0 and 1 for each DOF i of q'. Then we set the partial distance p_i for DOF i to $d(q_i, q_i + q'_i) = \frac{rnd_i \, * \, stepsize}{||rnd * w||}$. We now know how much each DOF contributes to the total distance. We set the translational and rotational$_1$ components of q'_i to $\pm p_i$. The calculation of the rotational$_3$ component is more complicated. We represent this component as a random 3D unit axis $a = (a_x, a_y, z_z)$ and an angle of revolution θ about that axis. (Since a revolution of more than π radians makes no sense, we constrain θ to $0 \leq \theta \leq \pi$.) This representation can easily be converted to a quaternion, i.e. $q'_{rot_3} = (a_x \sin(\theta/2), a_y \sin(\theta/2), a_z \sin(\theta/2), \cos(\theta/2))$. If we set θ to p_i, the distance between quaternion q_{rot_3} and $q'_{rot_3} * q_{rot_3}$ equals p_i.

Besides choosing a random vector, we also need to reverse such a direction. For translational and rotational$_1$ DOFs, we only have to negate the corresponding value in the vector. For a rotational$_3$ DOF, which is represented by the unit quaternion (x, y, z, w), the opposite direction is obtained by negating the x, y and z components, i.e. $(-x, -y, -z, w)$.

Finally, we need to add a direction q' to configuration q. For translational and rotational$_1$ DOFs, we can add up the values. If rotational$_1$ DOF is periodic, we have to make sure that the value remains in the range between 0 and 2π. For the rotational$_3$ DOF, q'_{rot_3} has to be multiplied by q_{rot_3}.

C. Directional clearance

We measure the directional clearance as the distance between a configuration q and configuration $q' = q \oplus (k * q_{dir})$, where k is the number of steps that can be taken in direction q_{dir} before an obstacle is hit. Furthermore, if a value in q' is not within its bounds anymore (such as the bounds of the workspace or bounds of the joints) no more steps are taken. It can be useful to set a maximum on the number of steps because this leads to a smaller number of collision checks which results in a more efficient algorithm. For example, in some applications, it is not useful to have a clearance that is larger than some threshold c_{min}. The calculation of the directional clearance resembles the work done in [6].

D. Path validation

As a path is forced to stretch and shrink during the retraction, this path may not be valid anymore after an iteration the algorithm. A path Π is valid if $\forall i : d(\Pi_i, \Pi_{i+1}) \leq step$. In this section we will show how to construct a new valid path.

First, we make a copy of Π (which is the path before updates were taken place), i.e. $\Pi' \leftarrow \Pi$. After one iteration, the configurations of Π' may have been updated. We construct a new path Π'' which will contain all configurations from Π' and new configurations to assure that Π'' will be valid.

For all i, we check if $d(\Pi'_i, \Pi'_{i+1}) > step$. This may occur when a configuration Π'_i is moved. In general, Π'_i can be left unchanged, moved in direction dir^+ or moved in direction dir^-. There are two cases in which this distance will be larger than $step$. We handle them as follows.

In the first case, Π'_i is left unchanged while Π'_{i+1} is updated. First we append Π'_i to Π''. Then, we either append Π_{i+1} or $\pi =$ Interpolate$(\Pi_i, \Pi'_{i+1}, 0.5)$ to Π''. We append the configuration that has the largest clearance. A similar procedure can be performed if Π'_i is updated while Π'_{i+1} is left unchanged.

In the second case, both Π'_i and Π'_{i+1} are updated. Again, we first we append Π'_i to Π''. Then, we either append both Π_i and Π_{i+1} or $\pi =$ Interpolate$(\Pi'_i, \Pi'_{i+1}, 0.5)$ to Π''. If the clearance of π is larger than the minimum clearance of Π_i and Π_{i+1}, we append π, and vice versa.

During the construction of Π'', configurations may be interpolated. For translational and rotational$_1$ DOFs, we can use linear interpolation. The interpolation between two quaternions can be performed by spherical linear interpolation (SLERP), see [13] for implementation issues.

We mentioned that a path can shrink during the retraction. As a consequence, multiple configurations may be mapped to a small region. We remove the configurations Π_i for which holds that $d(\Pi_{i-1}, \Pi_{i+1}) \leq step$. After the deletion of Π_i, Π_{i-1} will be adjacent to Π_{i+1}. As required, the distance between them will be at most $step$.

Due to space limitations, the proof that the algorithm will keep the distance between each two adjacent configurations below (or equal to) $step$ is ommitted here.

E. Retraction toward the ridges of the medial axis

In this section, we will show that the algorithm retracts a path toward the *ridges* of the *medial axis*. We define the medial axis as the locus of points in $\mathcal{C}$-space having at least two equidistant closest points on the boundary of the $\mathcal{C}$-space obstacles. We define a ridge as the locus of points in $\mathcal{C}$-space that represent a locally maximum clearance.

In each iteration of the algorithm, we only update a configuration if its clearance increases. Such an update can lead to an insertion and a deletion of configurations. If a configuration π is inserted, then the clearance of π will be equal to or higher than the clearance of the configuration before it was updated. If a configuration is deleted, then it will not play a role anymore. Hence, the clearance along the path never decreases.

If we would not pin down the start and goal configurations of the path then the following would happen. As we move the configurations with a particular step size that is larger than zero, there is a moment for which they will have obtained a (locally) maximal amount of clearance. Hence, the configurations would eventually be placed on the ridges.

Unfortunately, as the start and goal configurations *are* pinned down, not all configurations will be placed on the ridges. However, these configurations will eventually obtain a high clearance as only improvements are allowed.

IV. Experiments

In this section, we will investigate how much our new algorithm (retraction to the medial axis of the $\mathcal{C}$-space) can improve the clearance along three paths. We compare this with the algorithm from [11] (retraction to the medial axis of the workspace). We integrated these algorithms in a single motion planning system called SAMPLE (System for Advanced Motion PLanning Experiments), which we implemented in Visual C++ under Windows XP. SAMPLE automates conducting experiments, i.e. statistics are automatically generated and processed. All experiments were run on a 2.66GHz Pentium 4 processor with 1 GB internal memory. We used Solid 3.5 for collision checking [14].

We considered the environments depicted in Fig. 4. They have the following properties:

Planar This is a simple 2D environment that contains an ugly path traversed by a square robot that can only translate. The bounding box of the workspace is 100x100 and the dimensions of the robot are 1x1.

Hole The free-flying robot, which has six DOFs (three translational DOFs and a rotation$_3$ DOF), consists of four legs and must rotate in a complicated way to get through the hole. Only where the path goes through the hole, the clearance is small. The bounding box of the workspace is 40x40x40. The dimensions of the hole are 5x5x0.5. The bounding box of the robot is 5x5x10 and the legs are 1 thick.

Manipulator The articulated robot, which has six rotational$_1$ DOFs, operates in a constrained environment. The bounding box of the workspace is 10x10x10.

We chose the weights for the DOFs as follows: 1.0 for the translational DOFs, 6.0 for the rotation$_1$ DOFs and 5.5 for the rotation$_3$ DOF. The weights for the rotational DOFs are larger than the weights for the translational DOFs to compensate for the different ranges of these DOFs. The value of the step size *step* is related to these weights. We set this variable to 0.5 for the Planar environment, 0.5 for the Hole environment and 0.01 for the Manipulator environment.

We recorded the following statistical data of a path Π: the minimum and *average* clearance. The average clearance gives an indication of the amount of free space in which the path can be moved without colliding with the obstacles. The average clearance equals to $1/|\Pi| * \sum_{i=0}^{n-1}\text{Clearance}(\Pi_i)$.

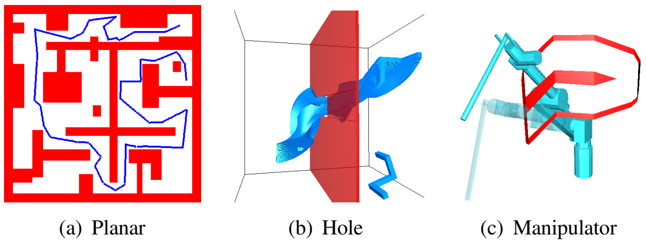

(a) Planar (b) Hole (c) Manipulator

Fig. 4. The three test environments and their corresponding paths

Results

The results of the experiments are stated in Table I and visualized in Fig. 5. The results for the MA ($\mathcal{C}$-space) technique were obtained by taking the average over 100 runs. For the MA (workspace) technique, we only performed one run per environment since this technique is deterministic.

Planar Since the robot has two translational DOFs, a retraction in the workspace results in a path having the optimal amount of clearance. The statistics show that our new technique approaches these optimal values (within 40 iterations). In general, for robots that have two translational DOFs, we recommend the workspace technique because this technique is faster (1.5 versus 15 seconds).

Hole The workspace technique doubled the amount of minimum and average clearance along the path. The path was calculated in 1.5 seconds. The $\mathcal{C}$-space technique outperforms the workspace technique because all DOFs were taken into account. The minimum amount of clearance along the path was further improved by a factor of three, as can be seen in the table and Fig. 5. The path converged after 100 iterations (which took 50 seconds on average).

Manipulator The path traversed by the manipulator cannot be improved by the workspace technique because this technique can only manipulate robots that have two or three translational DOFs. The minimum clearance along the initial path was very close to zero. The $\mathcal{C}$-space technique successfully introduced a relative large amount of extra clearance along the path. Also the average clearance was 2.5 times larger. These improvements are shown in Fig. 5. For clarity, we only visualized a part of the sweep volume of the manipulator. The path converged after 25 iterations (which took 60 seconds on average).

The running times indicate that improving the clearance along paths may be too slow to be applied online. However, in applications where safety is important the running times are not that crucial. For example, due to the difficulty of measuring and controlling the precise position of a manipulator arm, the arm can be damaged if it moves in the vicinity

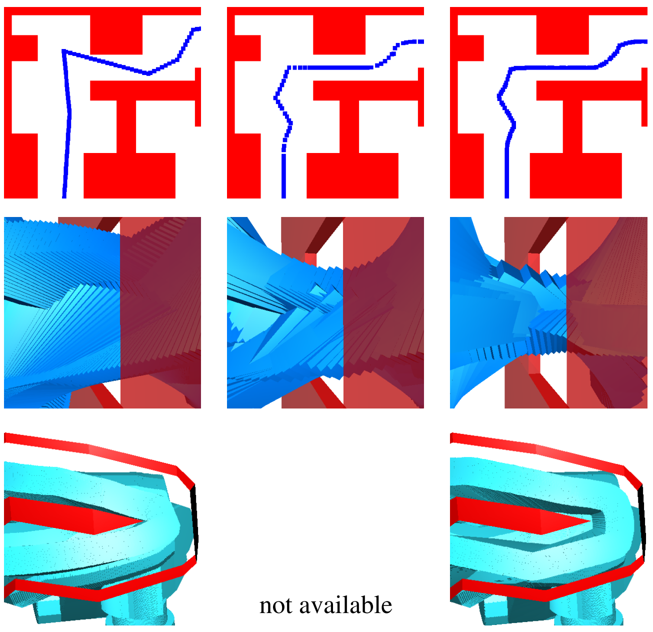
	Planar environment	
	min	**avg**
Initial path	0.35	2.84
MA (workspace)	1.80	4.49
MA ($\mathcal{C}$-space)	1.79	4.47

	Hole environment	
	min	**avg**
Initial path	0.21	1.55
MA (workspace)	0.44	3.37
MA ($\mathcal{C}$-space)	1.27	4.69

	Manipulator env.	
	min	**avg**
Initial path	0.01	0.13
MA (workspace)	n.a.	n.a.
MA ($\mathcal{C}$-space)	0.16	0.33

TABLE I

CLEARANCE STATISTICS FOR THE THREE PATHS. A LARGER NUMBER INDICATES A BETTER RESULT.



(a) Initial paths (b) MA (workspace) (c) MA ($\mathcal{C}$-space)

Fig. 5. A close-up of the paths in the three environments. The pictures in the left column show pieces of the initial paths. The paths in the middle column are retracted to the medial axis of the workspace while the paths in the right column are retracted to the medial axis of the $\mathcal{C}$-space. (See our website http://www.cs.uu.nl/∼roland for animations of these paths.)

of obstacles. Improving the clearance at the cost of a few minutes of calculation time can prevent damage to the robot and its environment.

V. CONCLUSIONS AND FUTURE WORK

We presented a new algorithm that improves the clearance along paths traversed by a broad range of robots which may reside in high-dimensional configuration spaces. The algorithm retracts paths of planar, free-flying and articulated robots toward the ridges of the medial axis of the $\mathcal{C}$-space by a series of guided random walks. The algorithm improved upon existing algorithms since higher amounts of clearance for a larger diversity of robots are obtained.

We expect that the running times can be dramatically decreased by incorporating learning techniques. This will a topic of future research. We are currently investigating how to create a graph from which large clearance paths can be extracted in real time, see [15]. Such a graph must capture the connectivity of the free $\mathcal{C}$-space for a particular virtual environment. We believe that this approach will enhance the quality of motion planners.

REFERENCES

[1] J.-C. Latombe, *Robot Motion Planning*. Kluwer, 1991.

[2] S. LaValle, *Planning Algorithms*. http://msl.cs.uiuc.edu/planning, 2005.

[3] V. Boor, A. Kamphuis, C. Geem, E. Schmitzberger, and J. Bouchet, "Formalisation of path quality," Delivery MOLOG project, 2000.

[4] A. Kamphuis and M. Overmars, "Finding paths for coherent groups using clearance," in *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, 2004, pp. 19–28.

[5] J. Kim, R. Pearce, and N. Amato, "Extracting optimal paths from roadmaps for motion planning," in *IEEE International Conference on Robotics and Automation*, 2003, pp. 2424–2429.

[6] J.-M. Lien, S. Thomas, and N. Amato, "A general framework for sampling on the medial axis of the free space," in *IEEE International Conference on Robotics and Automation*, 2003, pp. 4439–4444.

[7] H. Choset and J. Burdick, "Sensor-based exploration: The hierarchical generalized voronoi graph," *International Journal of Robotics Research*, vol. 19, no. 2, pp. 96–125, 2000.

[8] J. Vleugels and M. Overmars, "Approximating voronoi diagrams of convex sites in any dimension," *International Journal of Computational Geometry & Applications*, vol. 8, pp. 201–221, 1998.

[9] E. Masehianand, M. Admin-Naseri, and S. Khadem, "Online motion planning using incremental construction of medial axis," in *IEEE International Conference on Robotics and Automation*, 2003, pp. 2928–2933.

[10] K. Hoff, T. Culver, J. Keyser, M. Lin, and D. Manocha, "Interactive motion planning using hardware-accelerated computation of generalized voronoi diagrams," in *IEEE International Conference on Robotics and Automation*, 2000, pp. 2931–2937.

[11] R. Geraerts and M. Overmars, "Clearance based path optimization for motion planning," in *IEEE International Conference on Robotics and Automation*, 2004, pp. 2386–2392.

[12] N. Amato, O. Bayazit, L. Dagle, C. Jones, and D. Vallejo, "Choosing good distance metrics and local planners for probabilistic roadmap methods," in *IEEE International Conference on Robotics and Automation*, 1998, pp. 630–637.

[13] J. Kuffner, "Effective sampling and distance metrics for 3D rigid body path planning," in *IEEE International Conference on Robotics and Automation*, 2004, pp. 3993–3998.

[14] G. van den Bergen, *Collision Detection in Interactive 3D Environments*. Morgan Kaufmann, 2003.

[15] D. Nieuwenhuisen, A. Kamphuis, M. Mooijekind, and M. Overmars, "Automatic construction of roadmaps for path planning in games," in *International Conference on Computer Games: Artificial Intelligence, Design and Education*, 2004, pp. 285–292.