Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners

Jur P. van den Berg

Mark H. Overmars

institute of information and computing sciences, utrecht university

technical report UU-CS-2003-037

www.cs.uu.nl

Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners

Jur P. van den Berg Mark H. Overmars

November 2003

Abstract

The probabilistic roadmap (PRM) planner is a popular method for robot motion planning problems with many degrees of freedom. However, it has been shown that the method performs less well in situations where the robot has to pass through a narrow passage in the scene. This is mainly due to the uniformity of the sampling used in the planner; it places many samples in large open regions and too few in tight passages. In this paper, a technique based on a robot independent cell decomposition of the free workspace is proposed to guide the probabilistic sampling, such that the distribution of samples tends more toward the interesting regions in the scene. It is shown that this leads to improved performance on difficult planning problems in 2D and 3D workspaces.

1 Introduction

Motion planning is of great importance in various fields, such as robotics, virtual environments, maintenance planning, and computer-aided design. Although a number of exact and complete methods have been proposed for the robot motion planning problem (see [16] for an overview), these are seldomly used because they are only applicable to the simplest instances of the planning problem. For more complicated problems, where the robot has many degrees of freedom, these methods are computationally infeasible. Therefore, the focus has shifted toward probabilistic and heuristic methods, sacrificing completeness for speed and applicability.

A technique often used nowadays is the Probabilistic Roadmap (PRM) planner [2, 3, 12, 14, 17, 18]. The idea behind it is that a roadmap is created that represents the connectivity of the free part of the configuration space. The nodes of the graph are randomly sampled collision-free configurations that are connected by a simple and fast local planner (typically a straight-line motion in configuration space is used). The method is capable of solving motion planning queries in complex environments, and has been used in many practical situations.

However, the method has trouble in finding paths through narrow passages in the scene. This is mainly due to the uniformity of the sampling; it places many samples in *open* regions and too few in tight passages (a thorough analysis is given in [11]). This problem has received much attention of researchers in the field. The earliest strategies trying to tackle this problem use information from the roadmap adaptively during construction. They add additional nodes in the neighborhood of nodes that were only connected to a few neighbors [13, 15]. Later methods involve sampling more densely near obstacle boundaries [1, 6, 10], or far from obstacles, on the medial axis [20]. Despite the amount of work done on this topic (see the proceedings of the yearly IEEE International Conference on Robotics and Automation (ICRA)

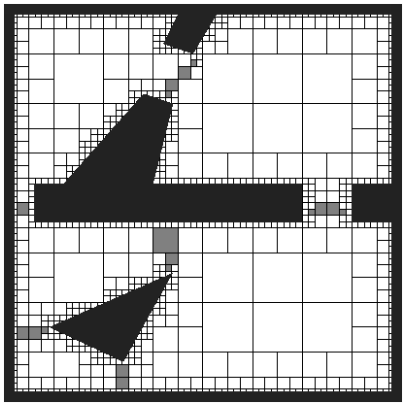

Figure 1: A cell decomposition of the workspace of a 2D example scene. The grey cells form the watersheds.

and the Workshop on Algorithmic Foundations of Robotics (WAFR) for many contributions), a generic solution has not yet emerged. The general observation remains, however, that the way of sampling is crucial for the result [7, 8, 9].

Whereas previous methods are mainly obstacle-based strategies, we propose a method that uses the shape of free workspace to guide the sampling in configuration space. Information from the workspace can only be used effectively when the configuration space more or less resembles the workspace. This means that narrow passages in the configuration space should correspond to narrow passages in the workspace, and that configurations in difficult regions can be mapped straightforwardly to the corresponding points in the workspace.

This holds for a large class of problems in robot motion planning: free-flying robots in two or three dimensional scenes. If the size of the robot is not too large compared to the size of the scene, each difficult region in the configuration space can be related to a relatively narrow passage in the workspace.

Our method identifies the narrow passages in the workspace using a cell decomposition approach. Because the workspace is only 3-dimensional, this can easily be done. Subsequently, cells of the decomposition are grouped into regions of interest by labeling them.

The labeling-part is the most crucial in this context. For this, we propose a method called *watershed labeling*, inspired by a technique from the field of image processing, called watershed segmentation [19]. It is a powerful method that separates large open regions from each other by so-called watersheds. These watersheds are positioned inside the corridors connecting these regions. As a result, the narrow passages are labeled differently than the open regions. See Fig. 1 for an example.

This information is used to effectively steer the sampling toward the most interesting regions of the scene. To this end, each of the labeled regions is assigned a weight indicating the chance that a sample in picked inside this region. To be precise, for a sample we pick the position in the region and the other degrees of freedom randomly. Narrow passage regions will receive relatively larger weights than open regions, which results in more samples in narrow passages. This will lead to faster connection of loose components in the probabilistic roadmap, and hence a quicker convergence toward a roadmap capturing the free space.

The global outline of this paper is as follows: in Section 2 we briefly recall the PRM method; in Section 3 we describe our global approach. The details are filled in in Section

4. In Section 5 we discuss watershed labeling and in Section 6 we will present experiments showing that the method indeed leads to significant improvements. Finally, we discuss the work presented and put forward some possible future work in Section 7.

2 PRM Basics

The motion planning problem is generally formulated in terms of the configuration space C, the set of all possible configurations of the robot. The dimension of the configuration space corresponds to the number of the robot's degrees of freedom. Each obstacle in the workspace, in which the robot actually moves, transforms into an obstacle in configuration space. Together, they form the forbidden part C_{forb} of the configuration space. $C_{free} = C \backslash C_{forb}$ denotes the set of all collision-free configurations.

Probabilistic roadmap planners generally work in two stages: a preprocessing stage, sometimes referred to as learning stage, and a query stage. In the preprocessing stage, a roadmap is constructed that forms a discrete representation of the connectivity of the free configuration space. The nodes of the roadmap are free samples, randomly chosen from the configuration space. There is an edge between two nodes, if a local planner finds a collision-free path between the corresponding configurations. The paths searched for are mostly simple straight line segments in the configuration space.

In the query stage, the constructed roadmap is used to find a path between a start and goal configuration. In general, these configurations are not present in the roadmap, so they are added to the roadmap using the local planner. Using Dijkstra or other graph searching methods, a path between the start and goal configuration is then easily found.

An important aspect of PRM is the way samples are picked from the configuration space. In the basic PRM this is done uniform randomly, but this yields many samples in open regions of the free space and few in narrow regions, while the inverse would be preferable. In this paper, we will focus on a sampling strategy yielding more samples in interesting regions.

3 Global approach

The idea of our method is to use the shape of the free workspace to guide the sampling in configuration space. For this purpose, the preprocessing stage of PRM is subdivided in two phases, a datastructure phase and a sampling phase. In the datastructure phase, a *robot independent* datastructure is built upon the workspace, identifying narrow and interesting regions. This proceeds as follows:

First, the free workspace, denoted by W_{free}, is decomposed into a collection of cells. Second, cells are grouped into regions of interest by labeling the cells according to some local properties. The set of cells having the same label is called a labeled region.

The subdivision of the workspace is used to steer the sampling. So, each of the labeled regions is assigned a weight indicating the chance that a sample in picked inside this region. Since narrow passages should receive relatively more samples than open regions, narrow passage regions are assigned a relatively large weight.

The datastructure phase is algorithmically depicted in Algorithm 1. In Section 3.2 and 3.3 we will discuss the datastructure phase in detail.

The sampling phase is straightforward. A sample is picked with its translational degrees of freedom inside a labeled region that is selected according to the weight distribution. The

Algorithm 1 DATASTRUCTUREPHASE

1: Decompose F into cells
2: Group cells into labeled regions by assigning a label to each of the cells
3: Assign each of the labeled regions a weight defining its sampling probability

other degrees of freedom are chosen uniform randomly. This sample is then treated as in standard PRM. In Section 3.4, the sampling phase is discussed in detail.

3.1 Relation workspace - configuration space

We want to use the workspace geometry as a guide for sampling in the configuration space, but the shape of the configuration space C not only depends on the workspace, but also on the shape and type of the robot moving through the workspace. We concentrate on free-flying robots here, i.e. the placement of the robot is described by its position and orientation (see the conclusions for some remarks about other robot types). The position of a robot is defined as the position of its reference point in workspace. To guarantee a strong resemblance between workspace and configuration space, we demand that this reference point is located in the interior of the robot. To be precise, the reference point of the robot is defined to be the center of the robot's largest inscribed sphere. Let r_{in} denote the radius of this sphere and be called the inner radius of the robot. The outer radius r_{out} is defined by the smallest circumscribed sphere with the reference point of the robot as its center.

The mapping between workspace and configuration space is straightforward: A point p in the workspace corresponds to the set of configurations in C which have p as their position. This set is called $C(p)$. The set of configurations in C corresponding to a set of points P from the workspace is denoted by $C(P)$. Reversely, the point in the workspace corresponding to a configuration c is denoted by $p(c)$.

The correspondence of workspace narrow passages to configuration space narrow passages is formalized as follows:

Theorem 3.1. *Let p be a point in W_{free}, and let $c \in C(p)$ be a configuration in C_{free}. Let $wcl(p)$ denote the workspace clearance for p, i.e. the Euclidean distance from p to the nearest workspace obstacle, and let $cl(c)$ denote the configuration space clearance of c. Then, $cl(c) \leq wcl(p) - r_{in}$.*

The above theorem states that points in the workspace close to obstacles, relate to points in the configuration space with even less clearance. So, difficult regions in the workspace certainly relate to difficult regions in the configuration space. The inverse relation is less strong:

Theorem 3.2. *Let c be a configuration in C_{free}, then $wcl(p(c)) \leq cl(c) + r_{out}$.*

This means that if the outer radius of the robot is not too large compared to the size and scale of the workspace, every configuration space narrow passage relates to a workspace narrow passage as well. In this case, the shape of the free workspace can very well be used to guide the sampling in the configuration space.

3.2 Cell decomposition

In the first step of our algorithm the free workspace is decomposed into a set of cells. Let this set of cells be denoted by X and the union of all the cells by D, i.e.: $D = \bigcup\{\chi \in X\}$. Two categories of methods to decompose a space into cells exist: exact and approximate ones. Exact cell decomposition methods result in a set of cells whose union exactly equals the free space of the scene, i.e. $D = W_{free}$. Examples are triangulations (tetrahedralizations in 3D), trapezoidal maps and generalized Voronoi diagrams [4]. These methods extensively use the geometry of the obstacles in the scene. Since scenes can consist of tens of thousands of obstacles, geometric processing is often not feasible within the tight runtime demands. Therefore, we will focus on approximate cell decompositions. Examples are octrees and binary space partitions [16].

Approximate cell decompositions result in subsets of the free space, i.e. $D \subseteq W_{free}$. Mostly, they have a recursive nature. A coarse approximation of the free space becomes finer at each level, by subdividing cells that partially overlap both free and forbidden space. This may continue until a desired level of detail is reached. This is formalized as follows:

Definition 3.1. *Let $B_p(\epsilon)$ denote the ball of radius ϵ at position p. A cell decomposition is called ϵ-detailed if $\forall B_p(\epsilon) \subset W_{free} : p \in D$. Decompositions satisfying this criterion are denoted by D_ϵ.*

We require the cell decomposition technique to be able to produce an ϵ-detailed decomposition for every $\epsilon > 0$.

It is easy to see that the region in configuration space corresponding to the r_{in}-detailed decomposition $D_{r_{in}}$ totally contains the free configuration space:

Theorem 3.3. *Let r_{in} be the inner radius of the robot, and $D_{r_{in}}$ an r_{in}-detailed decomposition of W_{free}. Then $C_{free} \subset C(D_{r_{in}})$. Such cell decompositions are said to be totally covering the free configuration space.*

This observation is used later to prove the probabilistic completeness of our method.

The cell decomposition over the workspace should in principle be robot independent. This means that D has to be detailed enough for every robot. This is of course not possible, but when we choose a small enough ϵ, a subset of D_ϵ can be used for robots with an inner radius larger than ϵ. For robots that have no inscribed ball ($r_{in} = 0$), e.g. a point robot or a line-segment robot, approximate cell decomposition techniques cannot produce a totally covering decomposition. For this kind of robots, however, a weaker statement can be made:

Theorem 3.4. *Let D_ϵ be an ϵ-detailed cell decomposition of W_{free}. Then, $\forall c \in C_{free} \setminus C(D_\epsilon) : cl(c) \leq \epsilon$.*

This fact is used later to prove a weaker form of probabilistic completeness for our method.

3.3 Labeling and weights

Approximate cell decompositions usually result in a large number of relatively small cells, especially near the boundary of the free workspace. This is, however, not what we want; we want a clear distinction between different regions of interest in the scene, for instance open regions and corridors between them. Therefore, cells should be grouped to create larger regions. This is done by labeling each cell: cells belonging to the same region are given the same label. The set of cells that is given a same particular label i is called a labeled region

R_i. The union of all the labeled regions should form the total cell decomposition. This means that all cells should be given a label.

Each of the labeled regions should be assigned a weight to steer the sampling, i.e. labeled regions have a chance to receive a sample with a probability proportional to their weights. If each labeled region would be given its volume as its weight, the resulting sampling distribution is uniform over the total cell decomposition. So, to give narrow regions relatively more samples and open regions relatively less, their weights should be raised or lowered respectively. None of the labeled regions should be given a weight of 0; each portion of the configuration space must have a probability > 0 to receive a sample, in order for the method to remain probabilistically complete, which we will prove below.

3.4 Sampling

The sampling scheme we propose on the datastructure created above is simple: first, select a labeled region R according to the weight distribution. Second, a sample is picked uniform randomly inside $C(R)$. This is done as follows: the labeled region R consists of a number of cells, so a cell is picked randomly from R, with a probability for each cell proportional to its volume. Then, the translational degrees of freedom are chosen inside this cell. The other degrees of freedom are chosen randomly. Algorithmically, it looks as follows:

Algorithm 2 SAMPLINGPHASE

1: Choose randomly a labeled region R according to the weight distribution
2: Choose randomly a cell inside R according to the cell volumes
3: Create a new sample: pick its position inside the cell and the other degrees of freedom uniform randomly
4: Treat this sample as in standard PRM

If the cell decomposition is totally covering the free workspace, i.e. the robot used in the scene has a maximal inscribed ball of radius r_{in} and the workspace decomposition is at least r_{in}-detailed, we can easily show that the above sampling scheme is probabilistically complete.

Theorem 3.5. *Let D be a totally covering cell decomposition for a robot, then the sampling scheme above is probabilistically complete.*

Proof Theorem 3.3 states that C_{free} is totally contained within the configuration space region $C(D)$. Since the whole region $C(D)$ is sampled, C_{free} is totally sampled too. This means that if the sampling continues long enough, every small ball with radius > 0 in C_{free} will contain a sample, so every path with clearance > 0 will eventually be found. This implies that the method is probabilistically complete [18]. ■

For cell decompositions not totally covering the free configuration space, a weaker statement can be made.

Theorem 3.6. *Let D_ϵ be an ϵ-detailed cell decomposition of W_{free}, and assume a path with clearance $> \epsilon$ exists. Then, the above sampling scheme will find a path.*

Proof Theorem 3.4 implies that any path with clearance $> \epsilon$ in C_{free} is contained in $C(D_\epsilon)$. Since $C(D_\epsilon)$ is totally sampled, every small ball with radius > 0 in $C(D_\epsilon)$ will contain a sample if the sampling continues long enough. This implies that the method will eventually find a path [18]. ■

4 Filling in the details

A number of important details have yet to be filled in, such as the used approximate cell decomposition technique and the way the cells are labeled and assigned weights. Let us recall the requirements posed upon these choices: We require that

- The cell decomposition technique is able for every scene to produce an ϵ-detailed decomposition for every $\epsilon > 0$.
- The union of the labeled regions forms the total cell decomposition, i.e. all cells must be given a label.
- No weight assigned to a labeled region is 0.

4.1 Cell decomposition

A well-known and obvious approximate cell decomposition datastructure is the octree. An octree is a rooted tree in which every internal node has eight children. Every node in the tree corresponds to a cube. If a node has children, then their corresponding cubes are the eight octants of the cube of the parent node.

A node of the octree is subdivided when the cell contains both obstacles and free space. If the cell only contains free space or obstacles, then it is marked as *free* or *full* respectively, and it is not subdivided. The subdivision of the mixed cells may propagate down until some desired level of detail is reached. The root of the octree corresponds to a cube or rectangloid covering the entire workspace. The leaves of the tree with label *free* together form the approximate cell-decomposition of the workspace. Note that all cells in the octree are congruent to the initial cell covering the total scene.

It is easily shown that the octree decomposition technique fulfills our requirement:

Theorem 4.1. *For every scene and for every $\epsilon > 0$, the octree cell decomposition technique is able to produce an ϵ-detailed decomposition.*

Many related decomposition techniques exist that satisfy the above criterion as well, such as the binary space partition. For the sake of simplicity, however, we will use the octree cell decomposition in this paper.

4.2 Labeling and weights

The simplest instance of labeling cells is to give each cell a different label. In other words, each cell is treated as a different labeled region. The simplest weight distribution for this labeling is to give each labeled region a weight according to its volume. This results in uniform sampling over the free space, and in that sense our method of using cell decompositions can be seen as a generalization of standard PRM.

Another obvious distribution of weights is to give each labeled region the same weight, i.e. they all have the same chance to be sampled. This weight distribution has a rationale, since cells, and thus the labeled regions, tend to be smaller when they are closer to obstacles, for instance in narrow passages. Yet each of the cells has the same chance to contribute a sample, leading to a larger density of samples in narrow passages when compared to uniform sampling over the entire configuration space.

A problem of this method is that also many small cells are generated along the boundary of obstacles, leading to many samples close to boundaries. It lacks a way to make a distinction between narrow passages and regions close to obstacles. A similar problem was observed in previous attempts to deal with the narrow passage problem [1, 6]. Yet, experiments show that this method already works better than basic PRM in some scenes.

To improve this, we need a way to distinguish cells in narrow passages from cells near a boundary in an open area. In the next section, we will describe a *watershed* approach to achieve this.

5 Watershed labeling

The simple labeling scheme discussed above is of course not favorable; we want to be able to really distinguish regions of interest in the scene. For such a labeling method, we let us inspire by image segmentation methods from the field of image processing. In image segmentation, the problem is roughly the same. Labeled regions should be formed of pixels that somehow belong together, according to a property defined in terms of the image itself.

The *watershed transform* [19] is a well-known segmentation method. It has shown to be a powerful method in many applications. Watershed segmentation yields nice and convincing labeled regions. It separates open regions from each other by watersheds.

The watershed transform may very well be used as a basis for our labeling scheme, where the watershed regions represent narrow passages and the regions they separate represent the open regions in the scene.

5.1 Watersheds in topographical landscapes

The watershed method was originally developed for discrete binary images (2D or 3D). So, we need to adapt the algorithm to make it suitable for approximate cell decompositions. We concentrate on octrees here, but we believe that it is extendible to other cell decompositions as well. We use an analogon with a topographical landscape to explain the algorithm for 2D cell decompositions. The algorithm is easily extended to higher dimensions.

We interpret the sizes of the cells in the decomposition as elevations in a topographical landscape. Large cells correspond to low elevations (see Fig. 2 for an impression). In the landscape we may distinguish different *catchment basins*. These are areas in the landscape associated with a local minimum, such that every imaginary rain drop falling in the catchment basin would end up in the particular local minimum. The boundaries between the catchment basins are called *watersheds*.

To find the watershed regions, the landscape is flooded. Starting from the minima of lowest elevation (the largest free cells), the water will progressively fill up the different catchment basins. Now, at each cell where the water coming from two different minima would merge, we build a watershed. At the end of this flooding procedure, each minimum is completely surrounded by watersheds, which delimit its associated catchment basin. The catchment basins now each form an open labeled region and the watersheds separating the catchment basins identify the narrow passages between open labeled regions.

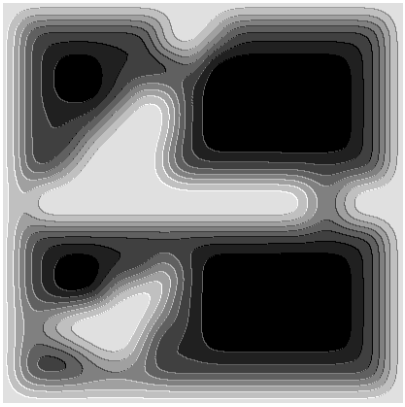

Figure 2: An impression of the topographical landscape induced by the cell decomposition of the example scene.

5.2 Watersheds for cell decompositions

In terms of an octree cell decomposition, each of the hierarchy levels in the octree correspond to a flood level in the topographical landscape, and rising the flood level corresponds to processing the cells in the next hierarchy level of the octree. Local minima correspond to isolated groups of cells of locally maximal sizes. These are given a unique label. The algorithm iteratively processes each of the hierarchy levels in the octree. We start with the highest hierarchy level, which contains the largest cells.

Processing a hierarchy level goes as follows: for each of the cells in the level, it is checked if it has an already labeled (larger) neighboring cell. If this is the case, the cell is appended to a first-in-first-out queue (see Algorithm 3, lines 2...4). Subsequently, each cell in the queue is processed. Again, its neighbors are inspected: if the neighbor cells that are already labeled all have the same label, the cell is given that label too. If there are neighbors with different labels, the cell is labeled as a watershed cell. Neighboring cells of the same hierarchy level that neither are already in the queue, nor are already labeled, are appended to the queue (see Algorithm 3, lines 6...15).

This process repeats until the queue is empty. Then, there may still be a number of cells in the current hierarchy level that have no label yet. These cells form new local minima and are given new labels. Groups of adjacent unlabeled cells are given the same label (see Algorithm 3, lines 17...26). They form together a local minimum.

After this, the algorithm proceeds to the next hierarchy level, carrying out the same process. This may continue until some desired level of detail is reached. Note that the watershed algorithm can be carried out simultaneously with a breadth-first construction of the cell decomposition.

The algorithm is given in pseudocode in Algorithm 3. A first-in-first-out queue is used in the algorithm. Three functions are defined on this queue:

- fifo_add(χ): Adds cell χ to the queue
- fifo_first(): Returns the cell at the front of the queue and removes it from the queue
- fifo_empty(): Returns *true* if the queue is empty and *false* otherwise

Furthermore, a function $N(\chi)$ is used. It returns a set containing all free neighboring cells of χ in the octree. Note that the function is only used for cells whose neighbors are either larger or have the same size, so this function is easily implemented.

Algorithm 3 WATERSHEDLABELING

```
 1: for all hierarchy levels of the cell decomposition do
 2:   for all cells χ in the current hierarchy level do
 3:     if there exist χ' ∈ N(χ) that is already labeled then
 4:       fifo_add(χ)
 5:
 6:   while not fifo_empty() do
 7:     χ ← fifo_first()
 8:     for all cells χ' ∈ N(χ) do
 9:       if χ' is already labeled then
10:         if χ is unlabeled then
11:           Give χ the same label as χ'
12:         else if χ has another label than χ' then
13:           Label χ as a watershed
14:       else if χ' is unlabeled and not in the queue then
15:         fifo_add(χ')
16:
17:   for all cells χ in the current hierarchy level do
18:     if χ is unlabeled then
19:       Give χ a new label
20:       fifo_add(χ)
21:       while not fifo_empty() do
22:         χ' ← fifo_first()
23:         for all cells χ'' ∈ N(χ') do
24:           if χ'' is unlabeled then
25:             Give χ'' the same label as χ
26:             fifo_add(χ'')
```

The watershed algorithm has a running time linear in the number of cells. Since no geometric operations have to be performed, such as collision checks, it is a purely combinatorial algorithm and therefore very fast. Building the cell decomposition of the workspace dominates the running time of the preprocessing phase.

With this technique, the narrow passages can be identified precisely (see Fig. 3a). Each of the regions associated with a local minimum gets a unique label. These regions form the open labeled regions. The watershed regions can be distinguished from each other when the labels of the regions they separate are maintained in the algorithm.

It is worth emphasizing that a narrow corridor is identified by a watershed, when it is narrow relative to the regions it connects. This implies that in workspaces that solely consist of narrow corridors, for instance a maze, nothing is marked as narrow passage. This is not a problem, since a narrow passage detector with an absolute measure would classify all of the free workspace as narrow passage. Both are useless for guiding random sampling; uniform sampling is the method of choice here anyway. An advantage of the relative approach is that a scene can be processed at different scales. When we have for instance a part of a city as our

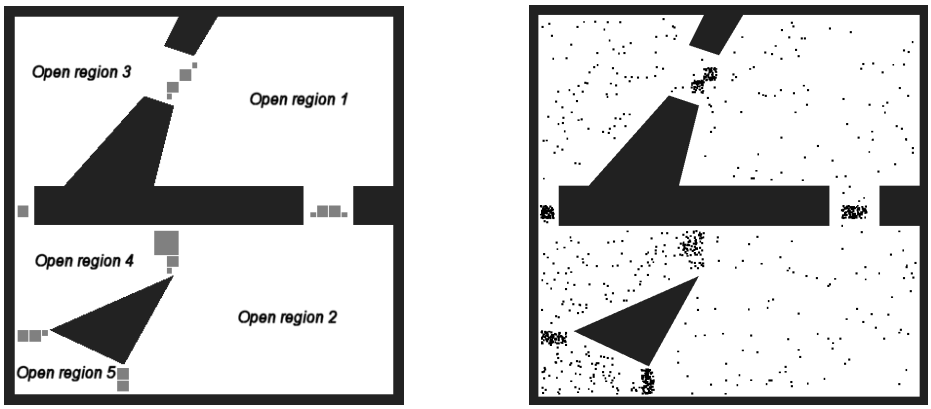

Figure 3: (a) Labeled regions in the example scene. The grey squares form watershed regions. (b) Distribution of samples in the example scene when each labeled region is given the same weight. The sample density in open region 5 is clearly higher than the sample density in open region 2.

workspace, the alleys are marked as narrow passage on the 'street scale', and doors separating rooms are marked as narrow passages on the 'house scale'.

In theory, every narrowing, small or not, is detected and marked as narrow passage. Fortunately, the watershed method allows for setting some thresholds, for instance a narrow passage should be a factor x more narrow than the regions it connects, or a narrow passage should be narrower than the absolute value of y. For the cell decomposition variant of the watershed algorithm, this problem is less pregnant, since cells of different levels already differ a constant factor in size.

5.3 Assigning weights

Assigning weights to the labeled regions can be done in many ways. We restrict ourselves to a simple weighting approach in this paper: each labeled region is given the same weight. Experiments showed that this works quite well for most scenes. In Fig. 3b an impression is given of how the samples are distributed using this method. More complicated weighting schemes are the subject of further research.

6 Experimental results

We experimented with our method on four different scenes: *rooms*, *clutter*, *hole* and *office* (see Fig. 4).

rooms in this scene there are three rooms with two narrow doors between them. The table must move through the doors to the other room. The watershed labeling yields three open regions in each of the rooms and two watershed regions in the doors between them.

clutter the clutter scene consists of 500 uniformly distributed tetrahedra. The configuration space will consist almost solely of narrow passages, so for this scene uniform random sampling is expected to work best. We test it with an L-shaped robot that has to travel from one corner to the opposite. The watershed labeling yields a disorganized distribution of

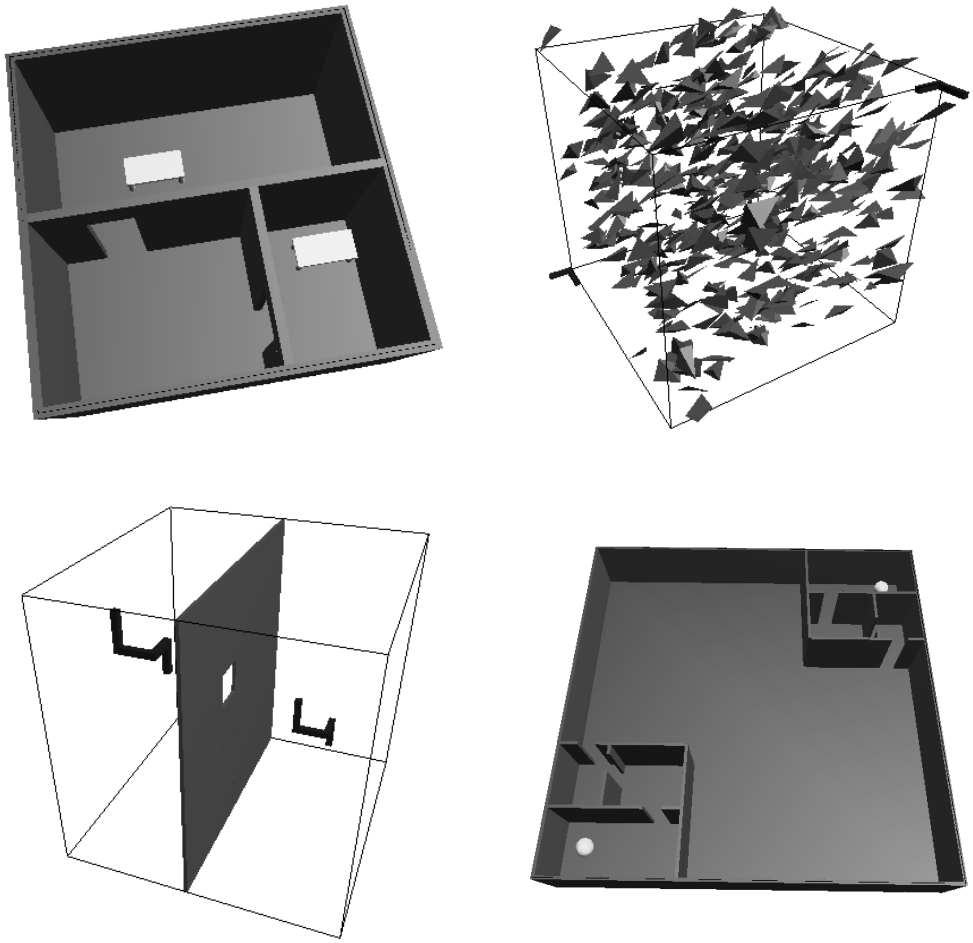

Figure 4: The scenes used in the experiments. (a) Rooms. (b) Clutter. (c) Hole. (d) Office.

'open' and watershed regions between the tetrahedra. In total, there are 296 different labeled regions.

hole this scene clearly has a narrow passage which is relatively hard for any kind of robot. A method to steer the sampling toward narrow passages should work very well in this case. We use a complicated robot having four legs. The watershed labeling yields two open regions on each side of the wall and a watershed region inside the hole.

office this scene has a large hall with two offices similar to the rooms scene in each of the corners. The robot (a sphere) has to move from one to the other office. We expect that denser sampling of smaller open regions can be of particular interest in this scene. The watershed labeling yields eight open regions. Three smaller ones in each of the offices and two large ones in the central hall.

6.1 Experimental setup

Our method was integrated in the motion planning system SAMPLE (System for Advanced Motion PLanning Experiments), implemented in Visual C++ under Windows XP. The experiments were run on a 2.4 GHz Pentium 4 processor with 1 GB internal memory. As basic collision checking package we use Solid [5].

In the experiments we consider the time needed to construct a covering roadmap for the scene. A roadmap is considered good enough when a pair of predefined query configurations is connected via the roadmap. For each scene such a query pair is devised (see Fig. 4). Because randomness is involved, we averaged the running times of 100 independent runs. Only for the hole scene, we performed 50 runs.

Since our method uses a preprocessing step to build a datastructure guiding the sampling, the actual running time is not directly comparable to traditional methods, where such a datastructure step is not present. Since our datastructure is robot independent, i.e.: for different types of robots a roadmap can be constructed using the same datastructure, the time needed to construct the datastructure is not added to the time needed to build the roadmap, but reported separately. For the rooms, clutter, hole and office scene it took respectively 0.55, 3.9, 0.06 and 2.0 seconds to build a watershed labeled cell decomposition with an appropriate level of detail. We believe that these times can be improved, but we did not concentrate on that.

We compare our method to previously proposed techniques that aim to better capture narrow passages in the sampling. In a recent publication [9], various sampling methods were compared to each other, among other things with the focus on narrow passages. Two conclusions can be drawn from that work: uniform random sampling performs quite well in many cases, and an obstacle-based variant in which only near-obstacle configurations are sampled, called *nearest contact*, appears to work well for narrow passage problems.

So, in our experiments we compare our method to these two sampling strategies. Fig. 5 summarizes our results.

6.2 Results

The results show that our method works particularly well for the rooms scene. This is because the watershed regions capture the narrow passages very well and enable an effective steering of the sampling. This results in considerably better performance of our method than uniform random and nearest contact.

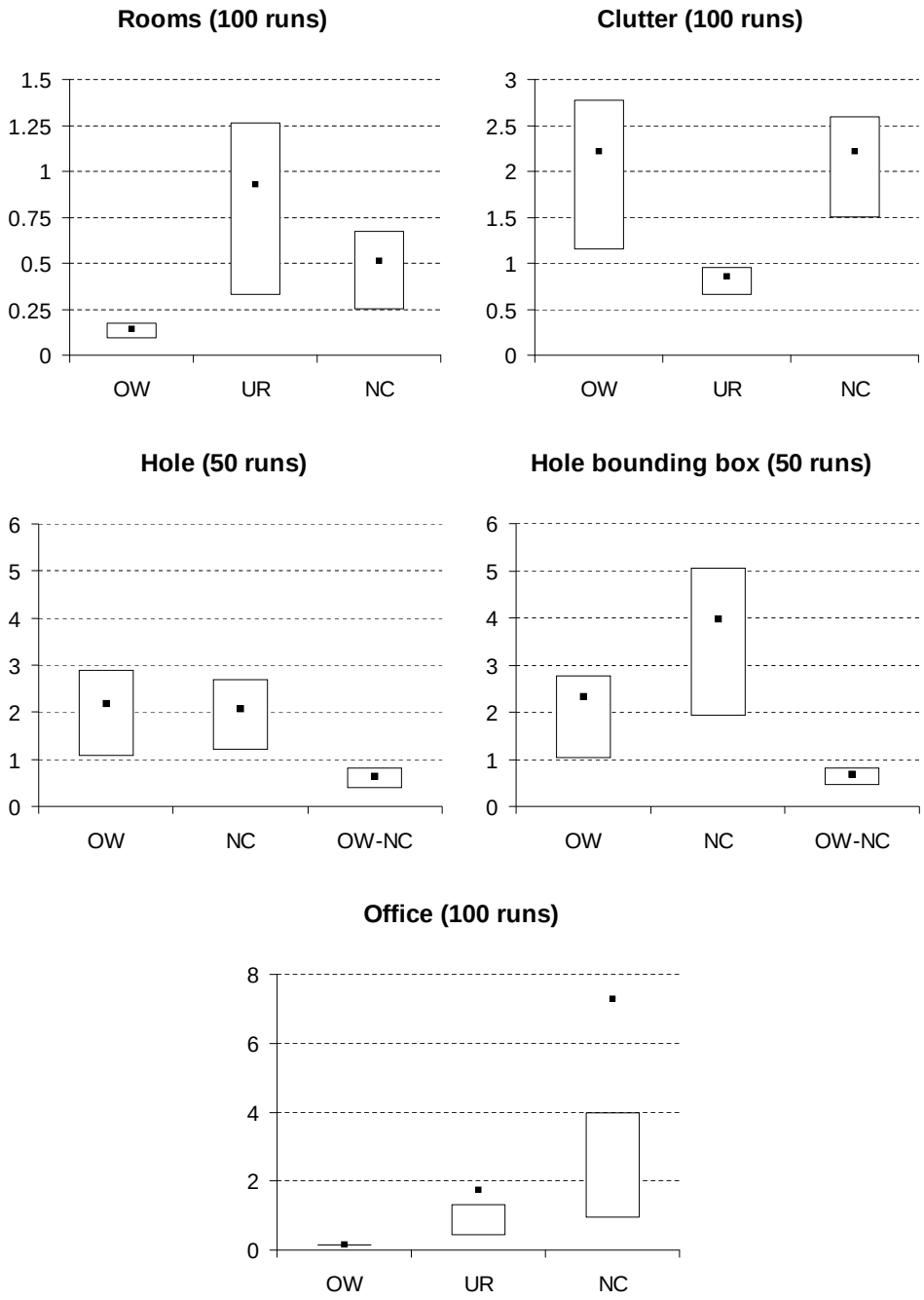

Figure 5: Running times in seconds for each of the experiments. OW is the octree-watershed method. UR is uniform random and NC is nearest contact. The boxes show the area between the 1st and 3rd quartile. The square shows the average value. For the hole scene, we report times for the scene without (c) and with (d) bounding box. The OW-NC method is a combination of our method and the nearest contact method. The results of the uniform random approach are not shown here, because they do not fit in the scale of the graph; the average running time is over 400 seconds.

For the clutter scene, uniform random works best, as was expected. Our method performs less well on this scene, although the performance is not worse than that of the nearest contact method.

For the hole scene our method performs comparable to nearest contact. The uniform random approach completely fails here. The nearest contact method performs relatively well for this scene, due to the complexity of the robot. The method has a clear disadvantage though: it generates samples close to the boundary of obstacles, but it cannot differentiate between narrow passages and open regions. This is made clear when a solid bounding box is created around the scene. The performance of our method is not much affected by this change, whereas the performance of nearest contact degrades drastically (see Fig. 5d).

In the nearest contact method, colliding configurations are pushed out of the obstacles. This is an effective, but expensive operation, so it should only be used in those places where it is useful. In that perspective, the combination of our method with the nearest contact method is interesting. Configurations are then picked according to our method, but when the configuration is colliding, it is pushed out of the obstacles. Because most colliding configurations created by our method lie indeed in narrow corridors, the expensive pushing operation is only used when relevant. We experimented with this method on the hole scene (see Figs. 5c and d). The results clearly show that this method combines the strengths of both methods. It outperforms all other methods by far. Note that, while the bounding box degrades the performance of the nearest contact method, it does not affect the combined method as hardly any sample will be generated near the bounding box.

For the office scene our method works best. Nearest contact fails here, because the obstacle boundaries in open regions are very large compared to narrow passage boundaries, and many useless samples are created there. For both the uniform random and the nearest contact method, the average running time does not lie within the first and third quartile. This implies that some runs takes a huge amount of time. For our method, this does not happen. The good and stable performance of our method in this scene is not only explained by the accurate labeling of narrow passages, but also because smaller open regions receive relatively more samples than large open regions. In the three rooms in each of the two offices, which may be regarded as the difficult regions of the scene, the sample density is much higher than in the large regions in the hall, where not so many samples are necessary. This results in a significant speedup of the construction of the roadmap.

7 Conclusions and future work

In this paper we presented a novel approach toward sampling in difficult regions. We used information from the workspace, extracted by an approximate cell decomposition, to guide the sampling. We showed that this approach has a great potential, especially in combination with watershed labeling. Watershed labeling has proven to be a powerful technique to identify narrow passages. We have used this successfully to speed up the construction of roadmaps in difficult scenes. Although we showed some promising results, we believe that a number of issues remain to be studied further.

A basic issue is the choice of the parameters and variables of PRM and our own method. In this paper, we used standard settings, but we think that fine-tuning these parameters will further improve our results.

Another improvement might be made when entire narrow passages are labeled. In our

current implementation, the watersheds are only one cell thick, while the narrow corridors may be much longer. It can also be useful to label the 'mouths' of the narrow passages. These often correspond with difficult regions in configuration space (e.g. for the hole scene).

As the watersheds form the difficult regions in the workspace we can also use this information to spend more work there, for example by only applying the nearest contact method for the samples chosen in the watersheds.

In this paper, only the sampling is guided by the cell decomposition, but it is possible to guide other parameters of the PRM method as well. For instance, the set of neighbors that is chosen for each sample can be selected acting on the cell decomposion. This may imply that neighboring samples are only chosen in the same or adjacent labeled regions, or that the maximum neighbor distance is adapted according to the sizes of the cells or regions. Also, the cell decomposition can be used as a datastructure for efficient neighbor searching. This will probably speed up our method even further.

Finally, we would like to make some remarks about various robot types. In this paper, we focused on free-flying robots. We believe, however, that the presented approach may be useful for other types of robots as well. For example, for an articulated robot with six links, the configuration space of the first three links resembles the configuration space of the entire robot, so our method may be applied in this space. In general: information from the configuration space of the dominating degrees of freedom (for free-flying robots, these are the translational dof's) may be used for guiding the sampling in the overall configuration space, but this remains to be studied further.

Acknowledgment

The authors would like to thank Dennis Nieuwenhuisen for developing the Callisto collision and visualization toolkit and Roland Geraerts for writing the SAMPLE software.

This research was 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).

References

[1] N. Amato, O. Bayazit, L. Dale, C. Jones, D. Vallejo; OBPRM: An obstacle-based PRM for 3D workspaces, in: P. K. Agarwal, L. E. Kavraki, M. T. Mason (eds.), *Robotics: The algorithmic perspective*, A. K. Peters, Natick, 1998, pp. 155-168.

[2] N. Amato, Y. Wu; A randomized roadmap method for path and manipulation planning, *Proc. IEEE Int. Conf. on Robotics and Automation*, 1996, pp. 113-120.

[3] J. Barraquand, L. Kavraki, J.-C. Latombe, T.-Y. Li, R. Motwani, P. Raghavan; A random sampling scheme for path planning, *Int. Journal of Robotics Research 16*, 1997, pp. 759-774.

[4] M. de Berg, M. van Krefeld, M. Overmars, O. Schwarzkopf; *Computational Geometry, Algorithms and Applications*, Springer Verlag, Berlin, 2000.

[5] G. van den Bergen; *Collision detection in interactive 3D computer animation*, PhD thesis, Eindhoven University, 1999.

[6] V. Boor, M. H. Overmars, A. F. van der Stappen; The Gaussian sampling strategy for probabilistic roadmap planners, *Proc. IEEE Int. Conf. on Robotics and Automation*, 1999, pp. 1018-1023.

[7] M. Branicky, S. LaValle, K. Olson, L. Yang; Quasi-randomized path planning, *Proc. IEEE Int. Conf. on Robotics and Automation*, 2001, pp. 1481-1487.

[8] R. Geraerts, M. H. Overmars; A comparative study of probabilistic roadmap planners, *Proc. Workshop on the Algorithmic Foundations of Robotics*, 2002, pp. 40-54.

[9] R. Geraerts, M. H. Overmars; Sampling Techniques for Probabilistic Roadmap Planners, to appear in: *Proc. Conf. on Intelligent Autonomous Systems*, 2004.

[10] D. Hsu, T. Jiang, J. Reif, Z. Sun; The Bridge Test for Sampling Narrow passages with Probabilistic Roadmap Planners, *IEEE Int. Conf. on Robotics and Automation*, 2003.

[11] D. Hsu, L. Kavraki, J.-C. Latombe, R. Motwani, S. Sorkin; On finding narrow passages with probabilistic roadmap planners, in: P. K. Agarwal, L. E. Kavraki, M. T. Mason (eds.), *Robotics: The algorithmic perspective*, A. K. Peters, Natick, 1998, pp. 141-154.

[12] L. Kavraki; *Random networks in configuration space for fast path planning*, PhD thesis, Stanford University, 1995.

[13] L. Kavraki, J.-C. Latombe; Probabilistic Roadmaps for Robot Path Planning, in: K. Gupta and A. del Pobil (eds.), *Practical Motion Planning in Robotics: Current Approaches and Future Directions*, John Wiley, pp. 3353, 1998.

[14] L. Kavraki, J.-C. Latombe; Randomized preprocessing of configuration space for fast path planning, *Proc. IEEE Int. Conf. on Robotics and Automation*, 1994, pp. 2138-2145.

[15] L. Kavraki, P. Švestka, J.-C. Latombe, M. H. Overmars; Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE Trans. on Robotics and Automation 12*, 1996, pp. 566-580.

[16] J.-C. Latombe; *Robot motion planning*, Kluwer Academic Publishers, Boston, 1991.

[17] M. H. Overmars; A random approach to motion planning, Technical report RUU-CS-92-32, Dept. Compt. Sci., Utrecht Univ., Utrecht, the Netherlands, October 1992.

[18] P. Švestka; *Robot motion planning using probabilistic roadmaps*, PhD thesis, Utrecht Univ., 1997.

[19] L. Vincent, P. Soille; Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol 13, No. 6, June 1991.

[20] S. A. Wilmarth, N. M. Amato, P. F. Stiller; MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space, *Proc. IEEE Int. Conf. on Robotics and Automation*, 1999, pp. 1024-1031.