Jur P. van den Berg
Mark H. Overmars

Institute of Information and Computing Sciences
Utrecht University, the Netherlands
berg@cs.uu.nl

Using Workspace Information as a Guide to Non-uniform Sampling in Probabilistic Roadmap Planners

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. Cell decomposition methods do not have this disadvantage, but are only applicable in low-dimensional configuration spaces. In this paper, a hybrid technique is presented that combines the strengths of both methods. It is based on a robot independent cell decomposition of the free workspace guiding the probabilistic sampling more toward the interesting regions in the configuration space. The regions of interest (narrow passages) are identified in the cell decomposition using a method we call watershed labeling. It is shown that this leads to improved performance on difficult planning problems in two- and three-dimensional workspaces.

KEY WORDS—path planning, probabilistic roadmaps, narrow-passage problem

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 Latombe 1991 for an overview), these are rarely 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 approximative methods, sacrificing completeness for speed and applicability.

A technique often used nowadays is the probabilistic roadmap (PRM) planner (Overmars 1992; Amato and Wu 1996; Barraquand et al. 1997; Švestka 1997; Glavina 1990; Kavraki and Latombe 1994; Kavraki 1995; LaValle, Branicky, and Lindemann 2004; Lindemann and LaValle 2004). 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 probabilistically complete, and is capable of solving motion planning queries in complex high-dimensional configuration spaces. It has been used in many practical situations. However, the method has trouble in finding paths through narrow passages in the environment. 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 Hsu et al. 1998).

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 (Kavraki et al. 1996; Kavraki and Latombe 1998). Later methods mostly involve sampling more densely near obstacle boundaries (Amato et al. 1998; Boor, Overmars, and van der Stappen 1999; Hsu et al. 2003), with the rationale that narrow passage regions also lie close to obstacles. Another class of methods, on the other hand, places samples far from obstacles, on the medial axis of the workspace (Wilmarth, Amato, and Stiller 1999; Holleman and Kavraki 2000; Pisula et al. 2000). The rationale of using the medial axis, besides its favorable property that it gives samples with a high clearance, is that it represents the topology of the free space in a simpler way (even though this

The International Journal of Robotics Research
Vol. 24, No. 12, December 2005, pp. 1055-1071,
DOI: 10.1177/0278364905060132

Figures 7, 10 and 12 appear in color online: http://ijr.sagepub.com

advantage is less strong in three-dimensional workspaces). The method of Wilmarth, Amato, and Stiller (1999) picks uniform random samples and retracts them to the medial axis. In Holleman and Kavraki (2000), an explicit representation of the medial axis is built, along which robot configurations are aligned to generate samples. The construction of the medial axis representation is sped up in Pisula et al. (2000) by using graphics hardware.

Despite the amount of work carried out on the narrow passage problem, a generic solution has not yet emerged. The general observation remains, however, that the way of sampling is crucial for the result (Branicky et al. 2001; Geraerts and Overmars 2004a, 2004b).

An older motion planning technique uses approximate cell decompositions of the configuration space (Latombe 1991). Mostly, an octree or a binary space partition tree is built upon the configuration space up to a certain depth. Mixed cells in the decomposition, i.e., cells that contain both free and forbidden space are further subdivided, whereas cells that are completely free or forbidden become leaves of the tree. A path can then be found through a series of adjacent free cells. It is a resolution complete method that does not suffer from the narrow passage problem, but it is only applicable to configuration spaces with few dimensions, say three or less.

In this paper, we propose a technique that combines the strengths of both the PRM method and the approximate cell decomposition method.[1] An approximate cell decomposition is built for the workspace of the robot. Since the workspace is only three-dimensional, this can easily be done. The cell decomposition is used to steer the probabilistic sampling in the overall configuration space toward the most interesting regions. The regions of interest (narrow passages) are identified in the cell decomposition. For this purpose we introduce an adapted technique from the field of image processing, which we call watershed labeling. 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 from the open regions (see Figure 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 is picked inside this region. 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.

Whereas previous sampling steering methods are mainly obstacle-based or medial axis-based strategies, our method can be regarded as a technique that uses the shape of the 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.

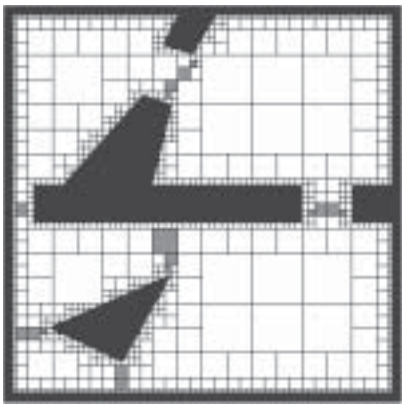


Fig. 1. A cell decomposition of the workspace of a two-dimensional example scene. The gray cells form the watersheds.

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. Hence, we will concentrate on free-flying robots in the rest of this paper.

Our method can essentially be described as a two-stage method. In a data structure phase, a robot-independent data structure is created for the environment, which can be used repeatedly in a sampling phase, in which a roadmap is created for a particular robot. Creating the data structure typically takes time in the order of one run of the sampling phase, so these costs are quickly recovered. We implemented our method, and experiments show that in comparison with previously proposed sampling strategies a considerable performance improvement is achieved on difficult planning problems.

The idea of using a robot-independent data structure is not new. In Holleman and Kavraki (2000) a similar approach is used, but, instead of using a cell decomposition as a basis for the data structure, they use the medial axis of the workspace. The advantage of the medial axis is that it increases the clearance of the robot. The disadvantage though is that during the sampling phase particular handle points on the robot must be situated by the user, which makes the approach difficult

1. An extended abstract of this paper has been published in van den Berg and Overmars (2004), and recently a similar technique has been developed based on our approach (Kurniawati and Hsu 2004).

to compare with our technique. Also, the preprocessing takes much longer in general (as indicated in Holleman and Kavraki 2000). The complexity of the medial axis directly depends on the complexity of the scene (i.e., the number of obstacle primitives). In particular, in three dimensions this may be problematic. For an (approximate) cell decomposition this relation is far less strong, if not non-existent. Further, a medial axis approach requires the use of (expensive) proximity queries, whereas simple collision checks suffice for cell decompositions. In Pisula et al. (2000) graphics hardware is used to speed up the construction of the medial axis. The running time though depends heavily on the required resolution. By using a sampling approach near the medial axis, they avoid the need for handle points. Because of the use of graphics hardware it is difficult to experimentally compare their approach to ours.

The global outline of this paper is as follows. In Section 2 we briefly recall the PRM method and in Section 3 the relation between workspace and configuration space is discussed in detail. In Section 4 we describe our global approach. The details are filled in Section 5. In Section 6 we discuss watershed labeling and in Section 7 we present experiments showing that the method indeed leads to significant improvements. Extensions to our method are discussed in Section 8. Finally, in Section 9 we discuss the work presented in this paper.

2. Probabilistic Roadmap 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 degrees of freedom of the robot. 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. $C_{free} = C \setminus C_{forb}$ denotes the set of all collision-free configurations.

PRM 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 PRMs 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 focus on a sampling strategy yielding more samples in interesting regions.

3. Relation Workspace: Configuration Space

We want to use the workspace geometry as a guide for sampling in the configuration space. This can only be done if the configuration space more or less resembles the workspace. This is in particular the case for free-flying robots in two- or three-dimensional workspaces. For articulated robots, on the other hand, the workspace barely resembles the configuration space. In this paper we concentrate on free-flying robots (see Section 8 for some remarks about other robot types) and in this section we formalize the relationship between workspace and configuration space in detail.

A configuration of the robot is described by its position and orientation. The position of the 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 (see Figure 2).

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 $C(P)$. In contrast, the point in the workspace corresponding to a configuration c is denoted by $p(c)$.

The distance $wd(p, p')$ between two points p and p' in workspace is straightforwardly defined as the Euclidean distance between them:

DEFINITION 1. $wd(p, p') = ||p - p'||$.

The workspace clearance $wcl(p)$ of a point p in workspace is defined as the distance between p and the nearest point on any workspace obstacle:

DEFINITION 2. $wcl(p) = \min_{p' \in W_{obst}} wd(p, p')$, where W_{obst} is the set containing all the workspace obstacles.

The distance $d(c, c')$ in configuration space between configurations c and c' is less easily defined. A transition between c and c' consists of both a translational part and a rotational part for free-flying robots. The rotation can be expressed in terms of an axis (going through the reference point of the

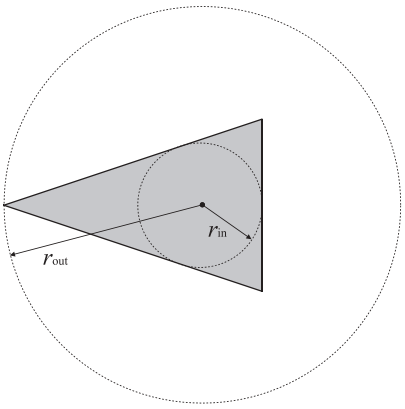


Fig. 2. r_{in} and r_{out} for a triangular robot (gray). The center of both circles is the reference point of the robot.

robot) about which the robot is rotated, and an angle. Let $\angle(c, c')$ be the angle in radians over which the robot is rotated in a transition from c to c'. Then $d(c, c')$ is defined as follows:

DEFINITION 3. $d(c, c') = wd(p(c), p(c')) + r_{out}\angle(c, c')$.

The distance $d(c, c')$ is an upper bound on the distance between any point on the robot in configuration c and the same point on the robot in configuration c'.

The configuration space clearance $cl(c)$ is defined as follows:

DEFINITION 4. $cl(c) = \min_{c' \in C_{forb}} d(c, c')$.

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

OBSERVATION 1. Let p be a point in W_{free} (the free part of the workspace). Then, $\forall c \in C(p) : cl(c) \leq wcl(p) - r_{in}$.

The above observation 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.

OBSERVATION 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 other words, the narrow passages in the configuration space can be identified in the workspace. Hence, the shape of the free workspace can very well be used to guide the sampling in the configuration space. The smaller r_{out}, the stronger the correspondence between the configuration space and the workspace. This, however, does not necessarily mean that our method performs worse for robots with large radii, as we will see in the experiments section.

4. Global Approach

To use the shape of the free workspace to guide the sampling in configuration space, the preprocessing stage of PRM is subdivided into two phases: a data structure phase and a sampling phase. In the data structure phase, a robot-independent data structure 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 using an approximate cell decomposition technique. Secondly, 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. Each of the labeled regions is assigned a weight indicating the chance with which a sample must be picked inside this region. Since narrow passages should receive relatively more samples than open regions, narrow passage regions are assigned a relatively large weight. In Sections 4.1 and 4.2 we discuss the data structure 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 other degrees of freedom are chosen uniform randomly. This sample is then treated as in the standard PRM method. In Section 4.3, the sampling phase is discussed in detail.

4.1. 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 three dimensions), trapezoidal maps and generalized Voronoi diagrams (de Berg et al. 2000). 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 focus on approximate cell decompositions. Examples are octrees and binary space partitions (Latombe 1991).

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 5. 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.

OBSERVATION 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 robot, however, a weaker statement can be made.

OBSERVATION 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.

4.2. Labeling and Weights

Approximate cell decompositions usually result in a large number of relatively small cells, especially near the boundary of the free workspace. Since we want a clear distinction between different regions of interest in the scene, for instance open regions and corridors between them, 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 and 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.

4.3. Sampling

The sampling scheme we propose on the data structure created above is simple. First, select a labeled region R according to the weight distribution. Secondly, pick a sample 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 randomly inside this cell. The other degrees of freedom are chosen uniform randomly. The resulting sample is treated as in the standard PRM method.

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

Proof. Observation 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 (Švestka 1997). □

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

THEOREM 2. 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. Observation 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 (Švestka 1997). □

5. Filling in the Details

In the above section, we posed the theoretical requirements on each stage of our hybrid approach. In this section we fill in the implementation details, such as the used approximate cell decomposition technique and the way the cells are labeled and

assigned weights. We show that these indeed fulfill the posed conditions.

5.1. Cell Decomposition

The most straightforward way to decompose the free workspace is by means of a regular voxel grid (or pixel grid in two dimensions) laid over the workspace. Each voxel is then either colliding or free, and the total collection of free voxels gives an approximation of the free workspace. This technique has been used extensively in the past (see Latombe 1991), and is used nowadays as well (see, for example, Leven and Hutchinson 2002; LaValle, Branicky, and Lindemann 2004) for motion planning.

For our purpose, the major drawback is that both open regions and narrow passages are covered by voxels of the same size, while the size of the voxels is chosen such that the narrow passages are adequately covered. This results in a total number of voxels which is too large to collision-check and process in reasonable running time.

Therefore, we use a cell-decomposition that adapts the size of the cells according to the local shape of the workspace. A well-known and obvious approximate cell decomposition for this purpose 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 and aligned in the same orientation as the initial cell covering the total scene. The cell-decomposition gives a better approximation of the free workspace if the orientation of the cells matches with the local alignment of the obstacles in workspace. In many realistic scenes we observe to some extent such an alignment. This should be exploited when choosing the orientation of the initial cell covering the entire scene.

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

OBSERVATION 5. 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 use the octree cell decomposition in this paper.

5.2. 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 the PRM method.

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 (Amato et al. 1998; Boor, Overmars, and van der Stappen 1999). Yet, experiments show that this method already works better than the basic PRM method 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 describe a watershed approach to achieve this.

6. 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 ourselves be inspired 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" (Vincent and Soille 1991) is a well-known segmentation method. It has been 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.

6.1. Watersheds in Topographical Landscapes

The watershed method was originally developed for discrete binary images (two or three dimensions). 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 can be extended to other cell decompositions as well. We use an analogy with a topographical landscape to explain the algorithm for two-dimensional 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 Figure 3 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 immersed in water. Imagine that tiny holes are pricked in the local minima of the landscape, then 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.

6.2. Watersheds for Cell Decompositions

In terms of an octree cell decomposition, each of the hierarchy levels in the octree corresponds 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 adjacent 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. Initially, all cells are "unlabeled".

Processing a hierarchy level goes as follows. For each of the cells in the level it is checked whether it has an already labeled (larger) neighboring cell (by a neighboring cell we mean a cell adjacent in the workspace). If this is the case, the cell is appended to a first-in–first-out queue (see Algorithm 1, lines 2–5). 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 1, lines 7–17).

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 1, lines 19–30). They form together a local minimum.

Fig. 3. An impression of the topographical landscape induced by the cell decomposition of the example scene (see Figure 1).

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 working of the algorithm for the example scene of Figure 1 is depicted in Figure 4. We assume that the cell decomposition is constructed simultaneously with the watershed algorithm. In Figure 4(a) we see the first hierarchy level for which there are free cells in the quadtree, which is the two-dimensional equivalent of the octree (the mixed cells are indicated light gray). Obviously, none of them has neighbor cells that are already labeled, so nothing happens in lines 2–17 of Algorithm 1. In lines 19–30 four new local minima are detected. Each of these is given a unique label (see Figure 4(b)). In the next hierarchy level (see Figure 4(b)) there are cells who have neighbors which are already labeled. These are added to the queue in lines 2–5, and are given the same label as their neighbor in lines 7–17. Also, neighboring cells in the same hierarchy that neither already have a label nor already are present in the queue are appended to the queue (the cells are labeled with 2 in the figure). Now, somewhat depending on the order in which the cells are processed, one of the cells will be labeled as a watershed separating the regions of local minima C and D. Next, in lines 19–30 a new local minimum is detected (lower left of the figure) and given a unique label.

The algorithm is given in pseudo code in Algorithm 1. A first-in–first-out queue is used in the algorithm. Four 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;

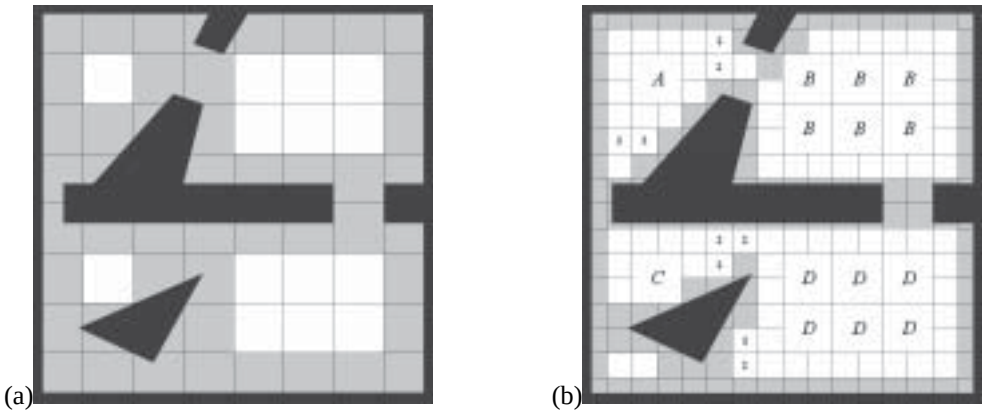


Fig. 4. The working of the watershed algorithm in the example scene of Figure 1.

- fifo_empty()—returns "true" if the queue is empty and "false" otherwise;
- fifo_present(χ)—returns "true" if χ is already in the queue and "false" otherwise. It is an $O(1)$-operation, since we set a flag at the cell when it is inserted into the queue.

Furthermore, a function $N(\chi)$ is used. It returns a set containing all free neighboring cells (i.e., cells adjacent in workspace) of χ. 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 1. 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:      χ.distance = 1
 5:      fifo_add(χ)
 6:
 7:   while not fifo_empty() do
 8:    χ ← fifo_first()
 9:    for all cells χ' ∈ N(χ) do
10:      if χ' is already labeled then
11:       if χ is unlabeled then
12:         Give χ the same label as χ'
13:       else if χ has another label than χ' then
14:         Label χ as a watershed
15:      else if χ' is unlabeled and not fifo_present(χ')
         then
16:       χ'.distance = χ.distance +1
17:       fifo_add(χ')
18:
19:   for all cells χ in the current hierarchy level do
20:    if χ is unlabeled then
21:      Give χ a new label
22:      χ.distance = 1
23:      fifo_add(χ)
24:      while not fifo_empty() do
25:       χ' ← fifo_first()
26:       for all cells χ'' ∈ N(χ') do
27:        if χ'' is unlabeled then
28:          Give χ'' the same label as χ
29:          χ''.distance = 1
30:          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.

In the algorithm, a distance value is maintained for each cell. These are not necessary for the working of the algorithm explained so far, but we will use them later on.

The construction of the watersheds is guaranteed using this algorithm. However, the exact location of the watersheds depend on the order in which the neighbors of cells are inspected. Yet, the narrow passages are identified precisely (see Figure 5(a)). Each of the regions associated with a local minimum has 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.

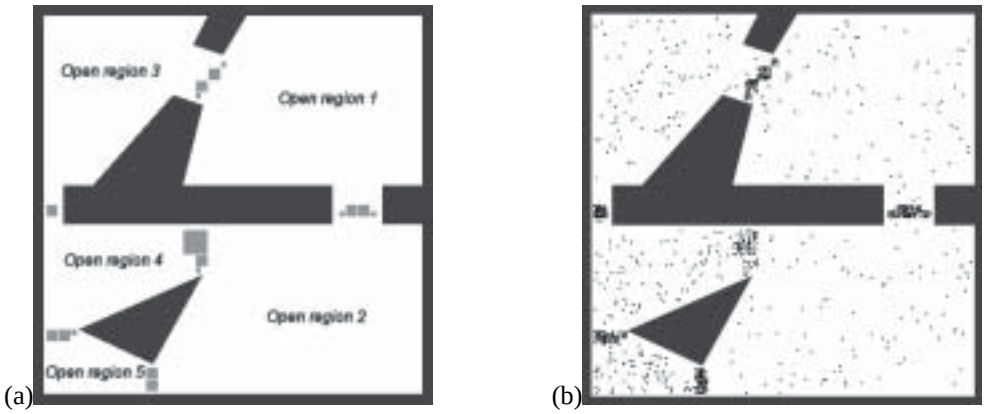


Fig. 5. (a) Labeled regions in the example scene. The gray squares form watershed regions. (b) Distribution of samples in the example scene when each labeled region is given the same weight. The sample density in open region 5 is clearly higher than the sample density in open region 2.

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 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 the original watershed algorithm, 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 A more narrow than the regions it connects, or a narrow passage should be narrower than the absolute value of B. For the cell decomposition variant of the watershed algorithm, cells of different levels already differ a factor of 2 in size. So, for the narrow passages to be detected, they should at least be twice as narrow as the regions they connect. We discuss this aspect in more depth in Section 8.2.

6.3. 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 have shown that this works quite well for most scenes (see Section 7). In Figure 5(b) an impression is given of how the samples are distributed using this method.

7. Experimental Results

7.1. Experimental Setup

Our method was integrated in our motion planning system SAMPLE (System for Advanced Motion PLanning Experiments), implemented in Visual C++ under Windows XP. All of our experiments were run on a 3.0 GHz Pentium 4 processor with 1 GB of internal memory. As basic collision-checking package we use Solid (van den Bergen 2004).

In the experiments we consider the time needed to construct a covering roadmap for the scene at issue. 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. Because randomness is involved, we averaged the running times of 100 independent runs.

Since our method uses a preprocessing step to build a data structure guiding the sampling, the actual running time is not directly comparable to traditional methods, where such a data structure step is not present. However, this data structure is robot-independent. This means that for different robots a roadmap can be constructed using the same data structure. So for each scene, this preprocessing has to be done only once. Therefore, we chose not to add the time needed to construct the data structure to the time needed to build the roadmap, but rather consider it as a part of modeling the scene into a computer model. Building a watershed labeled cell decomposition with an appropriate level of detail is in any case not expensive. We did not encounter running times over 5 s for the scenes used in our experiments. We believe that these times can be improved, but we did not concentrate on that.

7.2. Preliminary Experiments

Before running experiments with our method on a series of benchmark scenes, we first test the impact of the weighting

Fig. 6. A simple two-dimensional narrow passage scene of 100 by 100 units of length. The narrow passage has a width of three units of length and the robot (shown in the start and goal configuration) is 10 by 1 units of length.

scheme on the performance of our method, and we test how the performance of our method responds to a gradual increase in the outer radius of the robot. For these experiments, we used a simple two-dimensional narrow passage scene shown in Figure 6. The watershed labeling gives a watershed covering the small narrow passage, and two open regions on either side of the wall.

In our first experiment we vary the weight of the narrow passage according to which the samples are distributed over the labeled regions. The open regions have a weight of 1. The weight of the narrow passage is varied from 0.1 to 2.5 in this experiment. We averaged the running times to construct a graph connecting both query configurations over 100 runs. The results are shown in Figure 7(a).

The chart clearly indicates that we should not choose the narrow passage weight too small, and that for larger weights the performance is not influenced very much. In this experiment, the optimal narrow passage weight lies close to 1. Of course, the optimal weight depends on the size and the number of watersheds, but similar results were obtained for the benchmark scenes (see the next section). Therefore, we chose to fix the narrow passage weight at 1 in the rest of the experiments.

In the second experiment we explored the impact of a gradual increase in the outer radius of the robot on the performance of our method. A larger radius decorrelates the workspace and the configuration space. One would expect that this negatively affects the performance of the watershed method. Obviously, the problem becomes more difficult when the length of the robot increases. Therefore, we measured the running times to construct a graph connecting the query configurations for both the watershed method and the standard PRM method with uniform random sampling. The ratio between the running times of the watershed method and the uniform random method gives an indication of the relative performance of the watershed method. The results are shown in Figure 7(b).

Contrary to what we expected, the watershed method becomes relatively better when the radius of the robot increases. This can be explained as follows. To connect the parts of the roadmap constructed on either side of the wall, it is crucial that a sample is picked which places the robot inside the narrow passage. For the uniform random sampling method, the probability of picking such a sample decreases dramatically when the length of the robot increases. Although the narrow passages in workspace and configuration space are less correlated for larger robot radii, the watershed in the narrow passage still helps greatly in finding such a sample for the watershed method. As a result, the watershed method more easily connects the roadmap components on either side of the wall.

7.3. Benchmark Experiments

To test the watershed method more extensively, and to compare it to other sampling strategies, we experimented with our method on four different benchmark scenes: rooms, clutter, hole, and office (see Figure 8).

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, one in each of the rooms, and two watershed regions in the doors between them. It took 0.55 s to construct a watershed labeled cell decomposition.

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 "open" and watershed regions between the tetrahedra. In total, there are 296 different labeled regions. Building the data structure took 3.9 s for this scene.

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, somewhat decorrelating the workspace and configuration space. The watershed labeling took only 0.06 s. Two open regions on each side of the wall and a watershed region inside the hole were formed.

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 office to the other. 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. It took 2.0 s to build a watershed labeled cell decomposition.

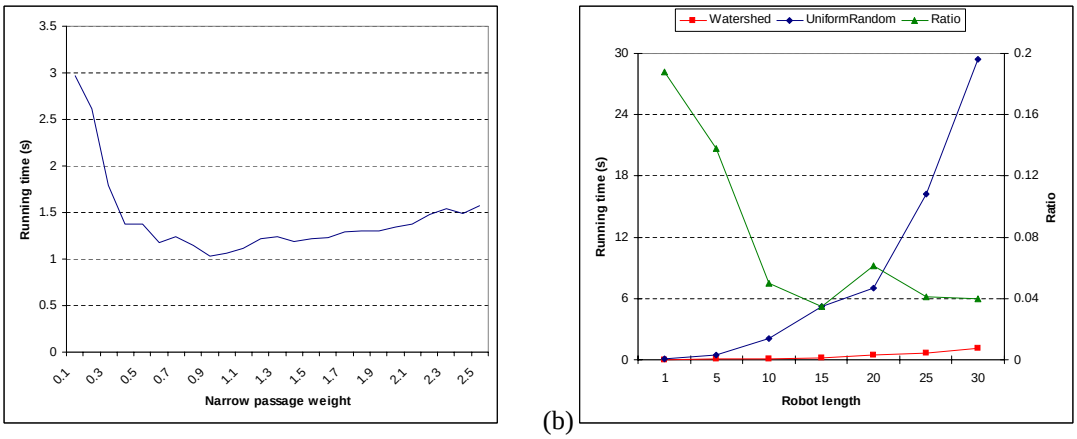


Fig. 7. (a) The running times to construct a graph connecting the query configurations (see Figure 6) for various values of the narrow passage weight. The results were averaged over 100 runs. (b) The running times for the watershed method and the uniform random method (left axis) to construct a graph connecting both query configurations for various lengths of the robot. The fraction of time the watershed method takes relative to the uniform random method is also shown (ratio, right axis).

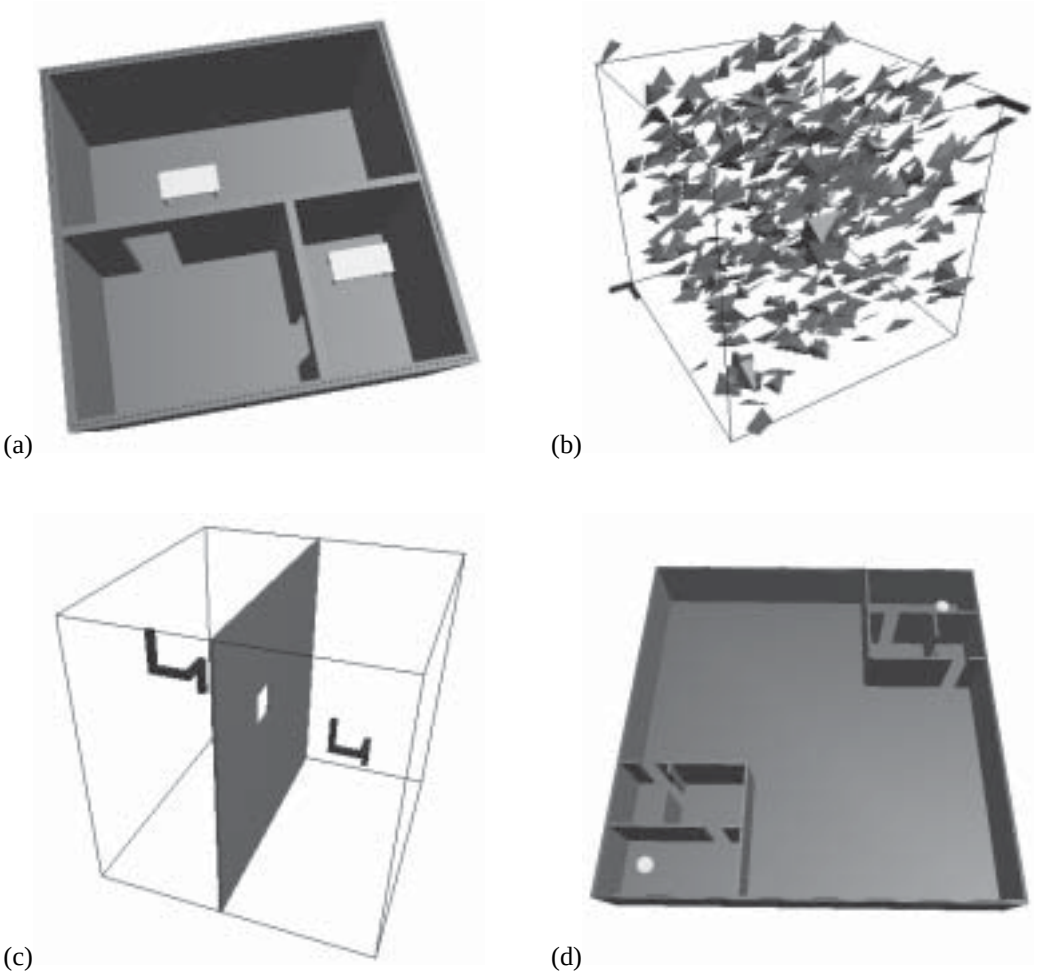


Fig. 8. The scenes used in the experiments: (a) rooms; (b) clutter; (c) hole; (d) office.

We compared our method to previously proposed techniques that aim to better capture narrow passages in the sampling distribution. In a recent publication (Geraerts and Overmars 2004b), 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. Other well-known sampling strategies we compared are Gaussian sampling (Boor, Overmars, and van der Stappen 1999), obstacle-based PRM (OBPRM; Amato et al. 1998), bridge-test sampling (Hsu et al. 2003) and medial axis PRM (Wilmarth, Amato, and Stiller 1999). We discuss them here in some more detail.

Uniform random. The uniform random sampling method straightforwardly picks its samples uniformly from the configuration space.

Gaussian. Gaussian sampling is an obstacle-based strategy that ensures the samples be distributed according to a Gaussian distribution around the configuration space obstacles. Initially, samples are picked uniform randomly, but for each sample another sample is picked at a distance corresponding to the desired variance of the Gaussian distribution. If one of the samples collides, and the other is collision-free; the collision-free sample is maintained. Other samples are rejected.

Obstacle-based PRM. In the OBPRM method, samples are picked uniform randomly, but only colliding samples are considered. They are pushed out of the obstacles by finding the first collision-free configuration in a randomly chosen direction. Samples that are initially collision-free are thrown away.

Bridge test. The bridge-test sampling method resembles the Gaussian sampling strategy in many respects. A pair of samples is chosen in the same way, but only if both samples are colliding, and the configuration midway both samples is collision-free; this configuration is considered as a sample. To keep the method probabilistically complete, one out of five samples is picked uniform randomly.

Medial axis. The medial axis PRM method uses the geometry of the workspace obstacles to retract the samples onto the medial axis of the free workspace, thereby creating samples with a high clearance. Medial axis gives a slightly higher probability of picking samples in narrow passages.

Nearest contact. Nearest contact is again an obstacle-based strategy. It is similar to the OBPRM method, but here colliding samples are pushed out of the obstacles using a penetration depth computation. Samples that are initially collision-free are rejected.

In our experiments we compared our method with all of the above sampling strategies. For each of the above methods, a number of parameters need to be set. As the scale and dimensions of all four scenes are comparable, we fixed these parameters over the various experiments, just as we did for our own method. The values of the parameters were chosen according to Geraerts and Overmars (2003). Figure 9 summarizes our results.

7.4. Results

As mentioned before, our method has two stages: a data structure phase in which a robot-independent cell decomposition is built for the workspace, and a sampling phase in which a roadmap is built for the configuration space of a particular robot. Here, we discuss the experimental results with respect to both of these.

Data structure phase. The time required to build the data structure depends on the depth of the octree, and on the amount of cells that need to be subdivided. The latter tends to be higher in cluttered environments, which is confirmed by our results (compare the times required for the clutter scene versus the hole scene). To create an appropriately detailed cell decomposition for the office scene, we needed one hierarchy level more than in the octrees of the other scenes. This explains the relatively large amount of time required for building its data structure. From the results, we see that creating the cell decomposition takes in general of the order of one run of the uniform random sampling method. Except for the clutter scene, the improved performance of our sampling method already recovers the costs on creating the data structure after two runs, when compared to uniform random sampling. This indicates that the method is superior when in a single scene multiple roadmaps must be constructed, for example, when there are different robot types.

Sampling phase. 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 other methods. We see that other advanced sampling strategies cannot really outperform the standard uniform random method. The bridge test and medial axis actually perform worse for this scene.

For the clutter scene, uniform random works best, as expected. Our method performs less well on this scene, although the performance is no worse than that of other advanced sampling methods.

For the hole scene our method again performs best, despite the complexity of the robot, which decorrelates the workspace and the configuration space. The uniform random approach completely fails here. All advanced sampling methods outperform it by far, especially medial axis PRM performs remarkably well in this scene. This is because the medial axis of this simple scene is completely formed by the hole. If we increase the complexity of the scene, for instance by creating a bounding box around it, experiments show that this advantage disappears. The performance of our method is not affected by such a change.

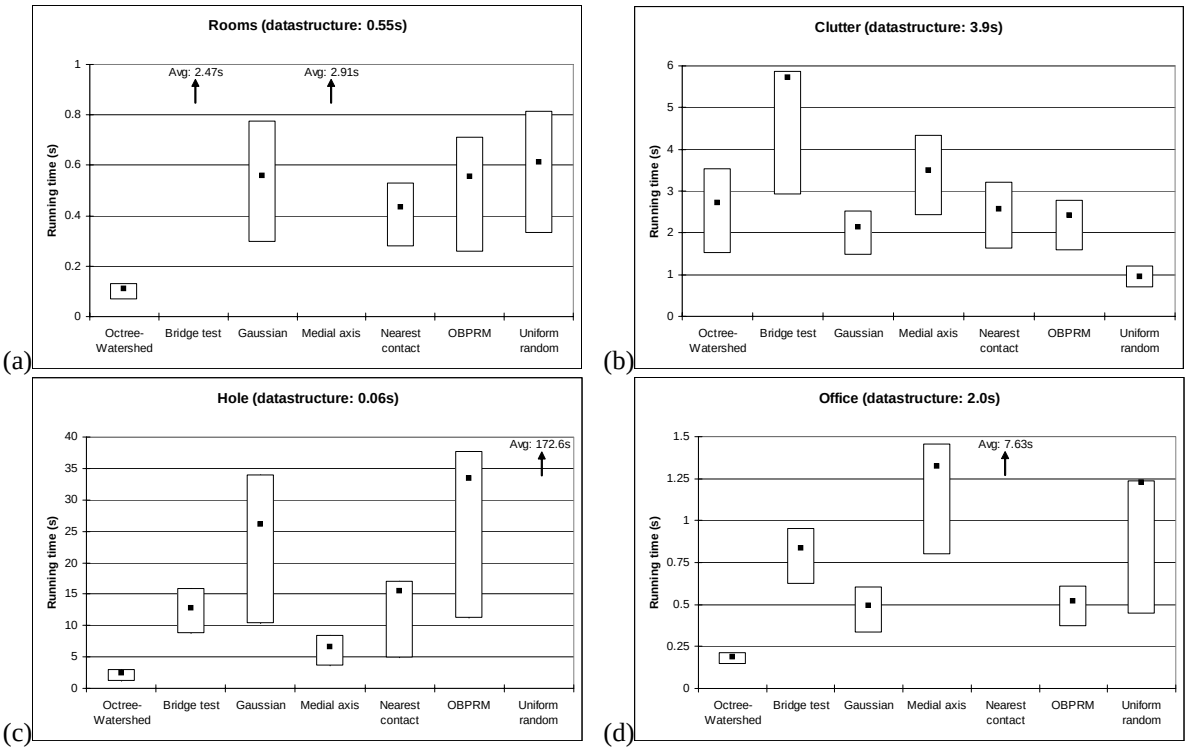


Fig. 9. Running times to create a roadmap for each of the experiments. The boxes show the area between the first and third quartiles. The square shows the average value. In some cases the results do not fit in the scale of the graph; the average running time is shown in text. In the title of each chart, the time required for building an appropriately detailed cell decomposition is given.

Also for the office scene our method works best. The nearest contact method fails here, because the obstacle boundaries in open regions are very large compared to narrow passage boundaries, and many useless samples are created there which are pushed out of the obstacles using an expensive penetration depth calculation.

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.

In general, we can say that, in narrow passage scenes, our method outperforms the previously proposed advanced sampling strategies. This is mainly due to the fact that other methods lack a way to distinguish obstacle boundaries in open regions from obstacle boundaries in narrow passages; they simply increase the sampling density near obstacles. Only the bridge-test method attempts to find samples in narrow passages. This method, however, uses a rejective strategy, which makes it hard in some scenes to actually find narrow passage samples. Our method does not have these drawbacks. It really distinguishes narrow passages from open regions, and it enables the injection of samples into narrow passages.

8. Extensions to our Method

8.1. Combination with Obstacle-based Strategies

The above results show that our method works very well for narrow passage problems. Before, mainly obstacle-based strategies were proposed, and the results show that these perform better than the standard uniform random method to tackle narrow passage problems. The nearest contact method, for instance, uses a penetration depth computation to push samples out of obstacles. This is an effective, but expensive operation, so it should only be used in those places where it is useful. In that light, the combination of our method with obstacle-based methods 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. Note that in the combined method we should also take initially non-colliding samples into account.

We briefly experimented with this combined method and the results indicate that performance of the octree–watershed method can be improved further. For example, in the hole scene, we observed an average running time of 1.97 s for the combined method to connect the query configurations, whereas the octree–watershed method and the nearest contact method individually take 2.38 and 15.42 s, respectively.

8.2. Additional Watershed Criteria

A limitation of the watershed method as discussed so far is that in some cases, in particular in scenes where the alignment of the scene with respect to the cell decomposition is poor, too many watershed regions are found. An example is given in Figure 10. We can solve this problem by re-examining the watersheds after they have been constructed. In the watershed algorithm, a cell is labeled as a watershed when it separates two regions that both contain a larger cell. In the example of Figure 10 we remove the erroneously detected watersheds when we would require that a watershed should separate regions that both contain a cell that is two sizes larger than the watershed cell. In fact, it can be proved that this prevents a false watershed to be formed as a result of a bad alignment of the scene. The re-examination of the watersheds can be carried out as a post-processing step of the watershed algorithm. If we determine a watershed to be a false positive, the watershed is deleted and the labeled regions it separates are merged.

The above fix, however, may give rise to a new problem, namely that watersheds are deleted which in fact do lie inside a narrow passage. The reason that this can happen should be sought in the cell decomposition: if the splitting planes of the larger cubes are in an unfavorable position, it can be hard to find a large free cell inside an open region. This problem can be solved by using a more advanced cell-decomposition technique, or by post-processing the octree cell decomposition. For instance, adjacent cells can be grouped to form a larger cells, or the largest fitting convex shape is found for each open region. Alternatively, we can add criteria for deleting false-positive watersheds based on the volume of the open regions the watershed is separating. Further research is required here.

8.3. Growing the Watersheds

Another limitation of the watershed method is that the watersheds are only one cell thick. For doorway-like narrow passages this suffices, but for tunnel-like passages, a single cell labeled as watershed is not likely to really increase the probability that a connection through the narrow passage is found (see Figure 11(a)). We can solve this problem by extending the watershed method to "grow" the watersheds so that they cover entire narrow passages.

To grow the watersheds such that they cover the whole narrow passage, we use the distances we maintain for each cell in the watershed algorithm. A rather simple algorithm suffices (see Algorithm 3). This algorithm calls Algorithm 2, which is provided the cell from which the watershed should grow and the label of the open region into which the watershed should grow. The key here is that the watershed is only grown to neighboring cells with a smaller distance value (see line 3 of Algorithm 2). To make sure that the watershed grows to both sides, an extra condition is incorporated in line 3 of Algorithm 2, as the cell next to the initial watershed cell may have the same distance value as the watershed cell (see Figure 11(a)). The algorithm is very fast, and gives the result as shown in Figure 11(b).

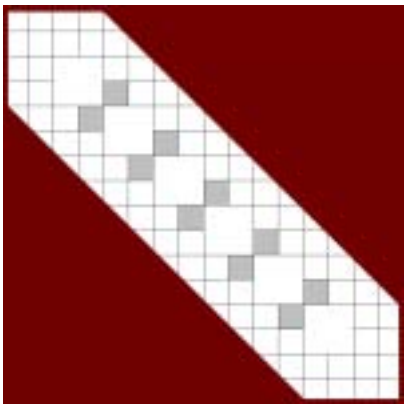


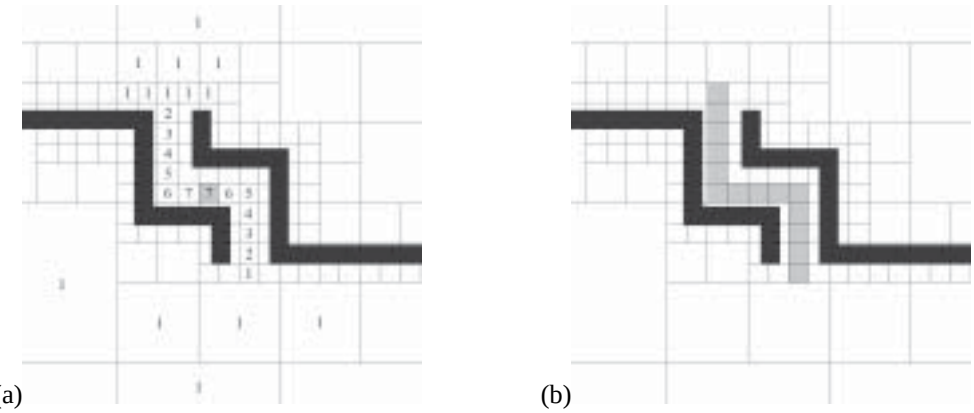
Fig. 10. A carefully chosen scene in which the watershed labeling detects too many narrow passages (gray cells).

Algorithm 2. GrowWatershedCell (χ, L)

1: **for all** cells $\chi' \in N(\chi)$ **do**
2: **if** the label of χ' equals L **then**
3: **if** χ'.distance $<$ χ.distance **or** χ is labeled as a watershed **then**
4: GROWWATERSHEDCELL(χ', L)
5: Label χ' as a watershed

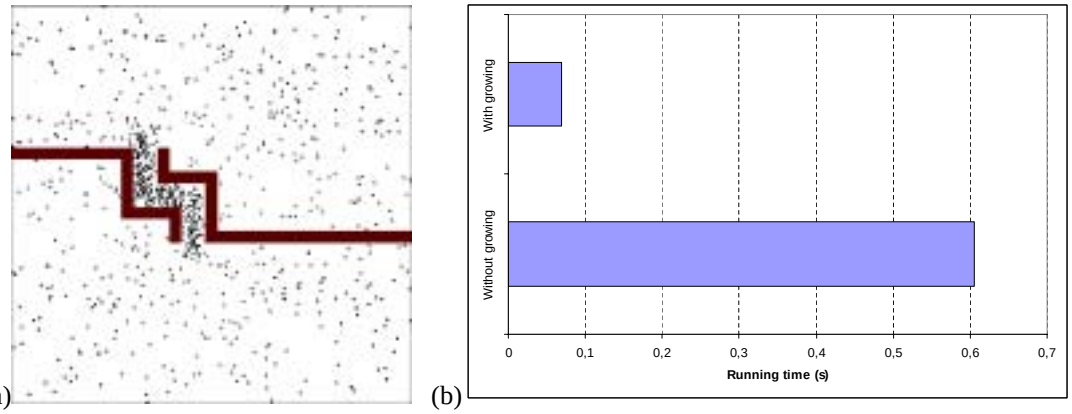
Algorithm 3. GrowWatersheds

1: **for all** cells χ in the cell decomposition **do**
2: **if** χ is labeled as a watershed cell **then**
3: $L \leftarrow$ the label of the region on the one side of the watershed
4: GROWWATERSHEDCELL(χ, L)
5: $L \leftarrow$ the label of the region on the other side of the watershed
6: GROWWATERSHEDCELL(χ, L)



(a) (b)

Fig. 11. (a) A single-cell watershed in an example scene (light gray). The numbers are the distance values associated with each cell. (b) A grown watershed (light gray).



(a) (b)

Fig. 12. (a) Distribution of samples in the narrow passage scene where each labeled region is given the same weight. (b) Difference in running time for this scene between the octree–watershed method with and without the feature of growing the watersheds.

The current algorithm is based on the observation that in a tunnel-like narrow passage the cells are equally sized; the watersheds only grow to cells of the same size. An alternative would be to allow the watersheds to grow to larger cells, up to some maximum size, as well. This could make the watersheds cover funnel-like passages that gradually change their cross-sectional radius. We chose, however, not to incorporate this into our implementation.

A simple experiment shows that this extension considerably increases the performance of the watershed method. For the scene shown in Figure 11 we use as a robot a square box that tightly fits in the narrow passage. It must find a path from one side of the tunnel passage to the other side. The results are shown in Figure 12(b). The chart clearly shows that the watershed-growing extension makes the method almost a factor of 10 faster in this scene.

8.4. Other Robot Types

Finally, we would like to make some remarks about various robot types. In this paper, we have focused on free-flying robots. We believe, however, that the presented approach may be useful for other types of robots as well (see also, for instance, Brock and Kavraki 2001). For example, for a standard articulated robot with six joints, the configuration space of the three major joints resembles the configuration space of the entire robot, so a similar approach to ours may be applied in this space. For car-like robots, the situation is almost similar to free-flying robots, and we believe our method can directly be used for this type of robots.

In general, our approach can be useful when the following ingredients are present. First, we need a clear distinction between dominating degrees of freedom and minor degrees

of freedom. Just as for free-flying robots and car-like robots, where the positional degrees of freedom are dominating the rotational degrees of freedom, the first three joints can be considered as dominant in the case of articulated robots.

Secondly, we need to be able to construct a cell decomposition in the space that guides the sampling. For free-flying robots and car-like robots we can just build a decomposition in the workspace, but for articulated robots we need the configuration space of the dominating degrees of freedom, because the correlation between workspace and the configuration space is small. Building cell decompositions in such spaces is hard, but it is possible to voxelize the space and apply the watershed method on the voxelized space. We must note, however, that the robot independence is lost in the case of articulated robots.

If both requirements are satisfied, information from the configuration space of the dominating degrees of freedom can be used for guiding the sampling in the overall configuration space.

9. Conclusions

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 proved to be a powerful technique to identify narrow passages. We have used this successfully to speed up the construction of roadmaps in difficult scenes.

A big advantage of our method is that it is able to distinguish between narrow passages and open regions, and that it enables injective sampling in difficult regions. Other advanced sampling strategies either use rejective strategies, or lack a way to distinguish open space obstacle boundaries from narrow passages. This explains the improved performance of our method compared to previous methods dealing with the narrow passage problem. The experimental results clearly show this.

A remaining issue is the choice of the parameters and variables of PRM and our own method. In this paper, we focused on the methodology and not on fine-tuning the parameters. We believe, however, that this will further improve our results.

As the watersheds form the difficult regions in the workspace, this information can further be exploited by treating watershed samples and open region samples differently. One could for example apply obstacle-based methods only for the samples chosen in the watersheds, or use different parameter settings for each region. In this paper, only the sampling is guided by the cell decomposition, but it is possible to guide other aspects of the PRM method as well. For example, the set of neighbors that is chosen for each sample can be selected based on the cell containing the sample currently under consideration. One could choose neighboring samples only in the same or adjacent labeled regions, or one could adopt the maximum neighbor distance according to the sizes of the cells or regions. Also, the cell decomposition can be used as a data structure for efficient neighbor searching. This will probably speed up our method even further, and we plan to investigate this further in the future.

Acknowledgments

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

Amato, N. and Wu, Y. 1996. A randomized roadmap method for path and manipulation planning. *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, MN, April, pp. 113–120.

Amato, N., Bayazit, O., Dale, L., Jones, C., and Vallejo, D. 1998. OBPRM: an obstacle-based PRM for 3D workspaces. *Robotics: The Algorithmic Perspective*, P. K. Agarwal, L. E. Kavraki, and M. T. Mason, editors, A. K. Peters, Wellesley, MA, pp. 155–168.

Barraquand, J., Kavraki, L., Latombe, J.-C., Li, T-Y., Motwani, R., and Raghavan, P. 1997. A random sampling scheme for path planning. *International Journal of Robotics Research* 16(6):759–774.

Brock, O. and Kavraki, L. E. 2001. Decomposition-based motion planning: a framework for real-time motion planning in high-dimensional configuration spaces. *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, Korea, May 21–26, pp. 1469–1474.

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

Boor, V., Overmars, M. H., and van der Stappen, A. F. 1999. The Gaussian sampling strategy for probabilistic roadmap planners. *Proceedings of the IEEE International Conference on Robotics and Automation*, 1999, pp. 1018–1023.

Branicky, M., LaValle, S., Olson, K., and Yang, L. 2001. Quasi-randomized path planning. *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, Korea, May 21–26, pp. 1481–1487.

Geraerts, R. and Overmars, M. H. 2004a. A comparative study of probabilistic roadmap planners. *Algorithmic Foundations of Robotics V*, J.-D. Boissonnat, J. Burdich, K. Goldberg, S. Hutchinson, editors, Springer-Verlag Berlin, pp. 43–57.

Geraerts, R. and Overmars, M. H. 2004b. Sampling techniques for probabilistic roadmap planners. *Proceedings of the Conference on Intelligent Autonomous Systems*, Amsterdam, the Netherlands, March 10–13, pp. 600–609.

Glavina, B. 1990. Solving findpath by combination of goal-directed and randomized search. *Proceedings of the IEEE International Conference on Robotics and Automation*, Cincinnati, OH, pp. 1718–1723.

Holleman, C. and Kavraki, L. E. 2000. A framework for using the workspace medial axis in prm planners. *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 24–28, pp. 1408–1413.

Hsu, D., Kavraki, L., Latombe, J.-C., Motwani, R., and Sorkin, S. 1998. On finding narrow passages with probabilistic roadmap planners. *Robotics: The Algorithmic Perspective*, P. K. Agarwal, L. E. Kavraki, and M. T. Mason, editors, A. K. Peters, Wellesley, MA, pp. 141–154.

Hsu, D., Jiang, T., Reif, J., and Sun, Z. 2003. The bridge test for sampling narrow passages with probabilistic roadmap planners. *Proceedings of the IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, September 14–19, pp. 4420–4426.

Kavraki, L. 1995. Random networks in configuration space for fast path planning. Ph.D. Thesis, Stanford University.

Kavraki, L. and Latombe, J.-C. 1994. Randomized preprocessing of configuration space for fast path planning. *Proceedings of the IEEE International Conference on Robotics and Automation*, San Diego, CA, May, pp. 2138–2145.

Kavraki, L. and Latombe, J.-C. 1998. Probabilistic roadmaps for robot path planning. *Practical Motion Planning in Robotics: Current Approaches and Future Directions*, K. Gupta and A. del Pobil, editors, Wiley, New York, pp. 33–53.

Kavraki, L., Švestka, P., Latombe, J.-C., and Overmars, M. H. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12:566–580.

Kurniawati, H. and Hsu, D. 2004. Workspace importance sampling for probabilistic roadmap planning. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, September 28–October 2, pp. 1618–1623.

Latombe, J.-C. 1991. *Robot Motion Planning*, Kluwer Academic, Boston.

LaValle, S. M., Branicky, M. S., and Lindemann, S. R. 2004. On the relationship between classical grid search and probabilistic roadmaps. *International Journal of Robotics Research* 23(7–8):673–692.

Leven, P. and Hutchinson, S. 2002. A framework for real-time path planning in changing environments. *International Journal of Robotics Research* 21(12):999–1030.

Lindemann, S. R. and LaValle, S. M. 2003. Current issues in sampling-based motion planning. *Proceedings of the International Symposium on Robotics Research*, Sienna, Italy, November.

Overmars, M. H. 1992. A random approach to motion planning. Technical Report RUU-CS-92-32, Department of Computer Science, Utrecht University, Utrecht, the Netherlands, October.

Pisula, C., Hoff, K., Lin, M. C., and Manocha, D. 2000. Randomized path planning for a rigid body based on hardware accelerated Voronoi sampling. *Proceedings of the 4th Workshop on the Algorithmic Foundation of Robotics (WAFR)*, Hanover, NH, March 16–18.

Švestka, P. 1997. Robot motion planning using probabilistic roadmaps. Ph.D. Thesis, Utrecht University.

van den Berg, J. P. and Overmars, M. H. 2004. Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners. *Proceedings of the IEEE International Conference on Robotics and Automation*, New Orleans, LA, pp. 453–460.

van den Bergen, G. 2004. *Collision Detection in Interactive 3D Environments*, Morgan Kaufmann, San Francisco.

Vincent, L. and Soille, P. 1991. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13:583–598.

Wilmarth, S. A., Amato, N. M., and Stiller, P. F. 1999. MAPRM: a probabilistic roadmap planner with sampling on the medial axis of the free space. *Proceedings of the IEEE International Conference on Robotics and Automation*, Detroit, MI, pp. 1024–1031.