An Efficient Deterministic Sequence for Sampling-based Motion Planners

Jan Rosell Maximilian Heise

Institute of Industrial and Control Engineering (IOC)
Technical University of Catalonia (UPC), Barcelona, SPAIN
email: jan.rosell@upc.edu maximilian.heise@upc.edu

Abstract

This paper presents a deterministic sequence that has the following good and useful features for sampling-based motion planners. It generates samples over a hierarchical grid structure in an incremental low-dispersion manner, allowing a uniform coverage of the Configuration Space. Due to its grid structure, neighborhood relationship between samples is easily computed, thus allowing roadmap-based planners to faster construct the roadmap. The sequence is computationally efficient and permits to locally control the degree of resolution required at each region of the Configuration Space by allowing the generation of more samples where they are most needed. The proposed sequence has been incorporated to a PRM-like planner and tested on a bend-corridor problem for different degrees of freedom.

1 Introduction

Sampling-based motion planners, like Probabilistic Roadmap Methods (PRMs) [6], or those based on the Rapidly-exploring Random Trees (RRT) [7], are giving very good results in robot motion planning problems with many degrees of freedom.

Following these random sampling methods, several approaches have been proposed that bias the sampling towards the most promising regions, thus improving the efficiency and allowing to cope with difficult path planning problems (including narrow passages). These approaches consider, for instance, a sample distribution that increases the number of samples on the border of the obstacles [3], around the medial axis of the free space [14], or around the initial and goal configurations [13]. Others propose the use of an artificial potential field to bias the sampling towards narrow passages [1], or the use of a lazy-evaluation approach that delays collision checking until it is absolutely needed [2]. Random sampling techniques have the advantage that task-specific knowledge can be incorporated to heuristically tailor the sample distribution in order to concentrate samples in critical regions.

In comparison to those approaches, deterministic sampling sequences have been proposed [4]. These deterministic sampling sequences have the advantages of classical grid search approaches, i.e. a lattice structure (that allows to easily determine the neighborhood relations) and a good uniform coverage of the Configuration Space ($\mathcal{C}$-space). For path planning purposes the uniform coverage is usually evaluated with the metric-based measure of dispersion, like the radius of the largest ball that does not contain any sample. Moreover, deterministic sampling sequences can provide an incrementally improved quality (in terms of dispersion) as the number of samples increases [10]. Deterministic sampling sequences applied to PRM-like planners are demonstrated in [8] to achieve the best asymptotic convergence rate and experimental results showed that they outperformed random sampling in nearly all motion planning problems.

As thoroughly argued in [9], the achievements of sampling-based motion planners are mainly due to their sampling-based nature, not due to the randomization (usually) used to generate the samples. Therefore, efforts should better be directed towards the study of deterministic and controllable ways of generating samples, rather than towards the proposal of heuristically guided randomization variants. In this line, this paper proposes a deterministic sampling sequence that, besides the general advantages of deterministic sequences, it is computationally very efficient and also allows to locally control the degree of resolution required at each region of the $\mathcal{C}$-space by generating more samples where they are most needed. The sequence is used in a PRM-like planner and tested on a bend-corridor problem for different degrees of freedom.

The paper is structured as follows. Section 2 presents the 2^d-tree decomposition and the code convention used to label and locate the cells. Section 3 introduces the generation of the deterministic sampling sequence that is tested

[1]This work was partially supported by the CICYT projects DPI2002-03540 and DPI2004-03104

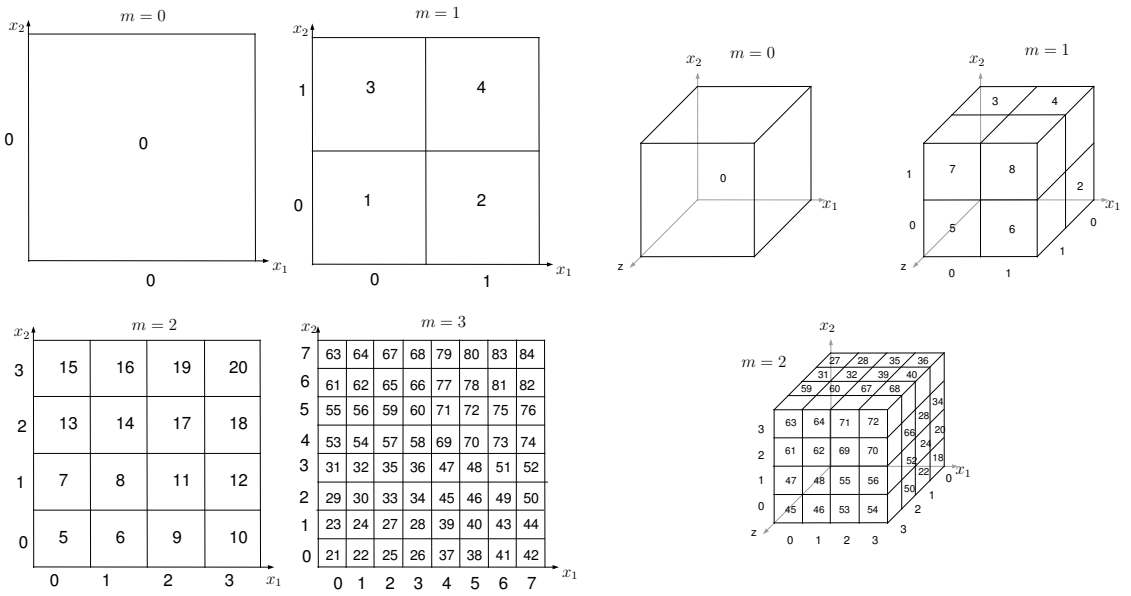

Figure 1: *Cell codes for different levels in the hierarchy in 2D and 3D $\mathcal{C}$-spaces.*

in Section 4 using a bend-corridor problem with 2, 3 and 6 d.o.f. Finally, Section 5 concludes the work.

2 $\mathcal{C}$-space decomposition

This section presents the $\mathcal{C}$-space decomposition used. Although this is not the main contribution of the paper (anyone dealing with multi-grids has coped with similar problems), it is described here since the formulation proposed allows an efficient implementation of the planner.

An initial cell, b^0, covering the entire $\mathcal{C}$-space is the tree root (b^0 is considered to have sides with unitary size). The levels in the tree are called partition levels and are enumerated such that the tree root is the partition level 0 and the maximum resolution corresponds to partition level M. Partition levels will be denoted by super-indices: a cell of a given partition level m will be called an m-cell, and denoted as b^m. The m-cells have sides of size $s_m = 1/2^m$ and form a regular grid called $\mathcal{G}_m$.

A code convention that univocally labels and locates each cell of the 2^d-tree decomposition of $\mathcal{C}$-space is introduced. Using this code convention, any subset of cells, e.g. those belonging to the subset $\mathcal{C}_{free}$ of collision-free configurations, can be managed as a list of codes, in a similar way as the *linear quadtrees* proposed in [5] for $d = 2$.

The cell codes are non-negative integers that univocally locate the cells in $\mathcal{C}$-space. The codes for a given partition level m range from C^m_{ini} to C^m_{end}, with:

$$C^m_{ini} = \frac{2^{dm} - 1}{2^d - 1} \quad (1)$$

$$C^m_{end} = 2^d C_{ini} \quad (2)$$

Since $C^m_{ini} = C^{(m-1)}_{end} + 1$, the proposed code convention uses all non-negative integers. Figure 1 shows the codes used for the cells of different partition levels for 2D and 3D $\mathcal{C}$-spaces.

2.1 Cell Codes

The obtention of the code of a cell from its location in $\mathcal{C}$-space and vice-versa is done as follows. Let:

- The index matrix $V^m_{b_k}$ be the binary $d \times m$ matrix whose rows are the binary representation of the indices $v^m_j \;\; \forall j \in 1 \dots d$ of an m-cell, b^m_k, on the regular grid $\mathcal{G}_m$:

$$V^m_{b_k} = \begin{pmatrix} v^m_1 \\ \vdots \\ v^m_j \\ \vdots \\ v^m_d \end{pmatrix} = \begin{pmatrix} a_{m1} & \dots & a_{i1} & \dots & a_{11} \\ \vdots & & \vdots & & \vdots \\ a_{mj} & \dots & a_{ij} & \dots & a_{1j} \\ \vdots & & \vdots & & \vdots \\ a_{md} & \dots & a_{id} & \dots & a_{1d} \end{pmatrix} \quad (3)$$

being a_{mj} and a_{1j} the MSB and the LSB, respectively, of the binary representation of v^m_j.

- W^m and W'^m be $d \times m$ matrices of weights, with:

$$w_{ij} = 2^{(m-j)d+i-1} \text{ for } i \in 1 \dots d \;\; j \in 1 \dots m \quad (4)$$

$$w'_{ij} = 2^{(j-1)d+i-1} \text{ for } i \in 1 \dots d \;\; j \in 1 \dots m \quad (5)$$

  Note that matrix W'^m coincides with W^m if the order of its columns is exchanged.

Then, given the indices $V^m_{b_k}$ of an m-cell b^m_k, the corresponding code $C^m_{b_k}$ is computed using the following expression:

$$C^m_{b_k} = C^m_{ini} + V^m_{b_k} \cdot W^m \quad (6)$$

where C^m_{ini} is the code of the initial cell expressed in equation (1), and the operation $A \cdot B$ represents the scalar product of matrices A and B. As an example the code of the 2-cell with indices $v^2_1 = 2$ and $v^2_2 = 1$ in the 2D $\mathcal{C}$-space of Figure 1 is obtained as:

$$V^2_{b_k} = C^2_{ini} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 4 & 1 \\ 8 & 2 \end{pmatrix} = 5 + 6 = 11 \quad (7)$$

On the other way round, given a code $C^m_{b_k}$, the indices $V^m_{b_k}$ are obtained as:

$$V^m_{b_k} = (C^m_{b_k} - C^m_{ini}) \& W^m \quad (8)$$

where the operation $a \& B$ between a scalar a and a matrix B computes the bit-AND operation between a and all the components b_{ij} of B. The components of the resulting matrix are 1 if the bit-AND operation gives a non-zero result, or 0 otherwise. As an example the indices of the 2-cell with code 11 are:

$$\begin{aligned} V^2_{b_k} &= (11-5) \& \begin{pmatrix} 4 & 1 \\ 8 & 2 \end{pmatrix} \\ &= 0110 \& \begin{pmatrix} 0100 & 0001 \\ 1000 & 0010 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{aligned} \quad (9)$$

Finally, the level m of a given cell with code k (needed to determine C^m_{ini}) is found as:

$$m = (\text{int}) \left\{ \frac{1}{d} \log_2[(2^d - 1)k + 1] \right\} \quad (10)$$

2.2 Neighbor relationship

Different levels of neighborhood can be obtained by combining the following functions:

- Manhattan distance function: Determines the codes of the cells that are at a distance Δ in a given direction v_j. It uses equation (6) with the index matrix modified adequately:

$$\text{manhattan}(C^m_{b_k}, v_j) = C^m_{ini} + \begin{pmatrix} v^m_1 \\ \vdots \\ v^m_j \pm \Delta \\ \vdots \\ v^m_d \end{pmatrix} \cdot W^m \quad (11)$$

- Parent function: Determines the code of the parent cell by using equation (6) modified as follows:

$$\text{parent}(C^m_{b_k}) = C^{m-1}_{ini} + \text{trunc}(V^m_{b_k}) \cdot W^{m-1} \quad (12)$$

  where the operation $\text{trunc}(A)$ eliminates the last column of matrix A.

- Descendant function: Determines the code of the first descendant cell by using equation (6) modified as follows:

$$\text{descendant}(C^m_{b_k}) = C^{m+1}_{ini} + \text{expand}(V^m_{b_k}) \cdot W^{m+1} \quad (13)$$

  where the operation $\text{expand}(A)$ adds a column of zeros to matrix A. The codes of the other $2^d - 1$ descendants are correlatives to this one.

3 Deterministic Sequence Generation

This section introduces the generation of the deterministic sampling sequence. The sequence relies on a predefined ordering of the 2^d descendant cells of any given parent cell.

3.1 Low-dispersion ordering of descendant cells

The position of a cell with respect to its parent cell can be defined by a binary word, i, with d bits, one for each axis. If n_j is the bit corresponding to the coordinate $x_j \;\; 1 \le j \le d$, then:

$$i = \sum_{j=1}^{d} n_j 2^{j-1} \quad (14)$$

Finding a low-dispersion ordering of the 2^d descendant cells of a parent cell is then equivalent to the finding of the sequence, L_d, of 2^d binary words such that each element of the sequence maximizes the minimum distance[1] to the previous elements of the sequence. This criterium is called

[1] The distance between two binary numbers is measured as the number of bits that differ, and is equivalent to the Manhattan distance between the cells they represent.

i	$L_2(i)$
$00 = 0$	$00 = 0$
$01 = 1$	$11 = 3$
$10 = 2$	$10 = 2$
$11 = 3$	$01 = 1$

i	$L_3(i)$
$000 = 0$	$000 = 0$
$001 = 1$	$111 = 7$
$010 = 2$	$010 = 2$
$011 = 3$	$101 = 5$
$100 = 4$	$100 = 4$
$101 = 5$	$011 = 3$
$110 = 6$	$110 = 6$
$111 = 7$	$001 = 1$

Table 1: *Ordering L_d for $d = 2$ and $d = 3$. Descendant cells are labeled from 0 to $2^d - 1$*

	Sequence L_6	Distance to Previous	Mutual distance
	000000		
S_1	111111	6	6
S_2	101010	3	3
	010101	6	3
S_3	100100	3	2
	011011	6	2
	001110	3	2
	110001	6	2
S_4	001000	4	1
	110111	6	1
	$\vdots$	$\vdots$	$\vdots$

Table 2: *Distances computed for the first elements of the ordering L_6.*

in [10] the maximization of the mutual distance, and is obtained using a digital construction method. This method uses a $d \times d$ binary matrix, T_d, that codifies the ordering and finds the sequence multiplying T_d by the binary representation of the indices of the sequence:

$$L_d(i) = T_d \;\; i = T_d \begin{pmatrix} n_1 \\ \vdots \\ n_d \end{pmatrix} \tag{15}$$

As an example Table 1 shows the ordering L_d obtained for $d = 2$ and $d = 3$ using the following matrices:

$$T_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \qquad T_3 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \tag{16}$$

Using the digital construction method it is demonstrated in [10] that:

a) The ordering obtained maximizes the mutual distance if each column of the matrix is chosen such that it is maximally distant to all the vectors spanned by the previous columns of the matrix (being the first column composed of ones).

b) Each column j (in combination with the previous columns) is used to generate a set S_j of $2^{(j-1)}$ new elements of the sequence, having all of them the same mutual distance.

In this paper we propose a matrix T_d constructed as follows. Each column $j \in 1 \ldots d$ is composed of $(j-1)$ zeros followed by a 1 (that corresponds to the diagonal). The rest of the column is filled by alternating $(j-1)$ zeros with $(j-1)$ ones until the column is completed. The resulting T_d is a lower diagonal matrix with non-zero diagonal elements and therefore is full rank. As a consequence, T_d is able to generate all the 2^d elements of the sequence.

As an example, for $d = 12$ the matrix is:

$$T_{12} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{17}$$

Using the proposed T_d, the first elements of the ordering obtained for $d = 6$ are shown in Table 2, together with the corresponding distances. Table 3 shows the mutual distances of the elements of L_d for $d = 2$ to $d = 12$. Each column of the table corresponds to the set S_j of the $2^{(j-1)}$ elements obtained by combining column j of matrix T_d with the previous ones.

The proposed convention to construct matrix T_d is general for any value of d and the sequence L_d obtained is near-optimum in terms of mutual distance. It can be appreciated in table 3 that, for any d, the mutual distances are always non-increasing and that, when there is a decrease, it is as minimum as possible. For lower values of d, alternative expressions of T_d can be found able to generate a better sequence L_d (i.e. with a slower decrease of the mutual distance), although a generalization to any d is not clear.

3.2 The sampling sequence

The sampling sequence, $s_d(k)$, is a sequence of cell codes that specifies the ordering in which the d-dimensional $\mathcal{C}$-space is to be explored. The sequence $s_d(k)$ is based on the recursive use of L_d.

Let $k \geq 0$ be the index of the sequence, m be the hierarchical level associated to k as expressed in equation (10),

d	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}
2	2	1										
3	3	1	1									
4	4	2	1	1								
5	5	2	1	1	1							
6	6	3	2	1	1	1						
7	7	3	3	1	1	1	1					
8	8	4	3	2	1	1	1	1				
9	9	4	3	3	1	1	1	1	1			
10	10	5	4	3	2	1	1	1	1	1		
11	11	5	5	3	3	1	1	1	1	1	1	
12	12	6	5	4	3	2	1	1	1	1	1	1

Table 3: *Mutual distances of the elements of L_d for d=2 to d=12, grouped in sets S_j (since all of the $2^{(j-1)}$ elements of a set have the same mutual distance).*

i	0	1	2	3	4	5	6	7	8	9
$s_2[i]$	0	1	4	3	2	5	17	13	9	8
i	10	11	12	13	14	15	16	17	18	19
$s_2[i]$	20	16	12	7	19	15	11	6	18	14

Table 4: *First 20 samples of sequence s_2.*

and T_d be the matrix that determines the cell ordering of the descendant cells as introduced in the previous section. Then:

$$s_d(k) = C^m_{ini} + (T_d V^m_{b_k}) \cdot W'^m \tag{18}$$

where $T_d V^m_{b_k}$ is the standard binary matrix multiplication between matrices T_d and $V^m_{b_k}$ and W'^m the weight matrix introduced in (5).

As an example, for $d = 2$ and $k = 6$ (with indices $v^2_1 = 1$ and $v^2_2 = 0$ on $\mathcal{G}_2$) the obtained sample is:

$$\begin{aligned} s_2(6) &= C^2_{ini} + \left[\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \right] \cdot \begin{pmatrix} 1 & 4 \\ 2 & 8 \end{pmatrix} \\ &= 5 + \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 4 \\ 2 & 8 \end{pmatrix} = 17 \end{aligned} \tag{19}$$

The first samples generated are shown in Table 4 and illustrated in Figure 2.

3.3 Adaptive behavior

When a given sampled configuration c (corresponding to the center of a cell with code K) is evaluated as a free configuration, and during the construction of a roadmap few or no free paths are found to connect c to its neighbor free samples, further sampling is required in its environment. This can be done using the following (re)sampling sequence:

$$r_d(j) = K2^{dm} + C^m_{ini} + (T_d V^m_{b_k}) \cdot W'^m \text{ with } j \geq 1 \tag{20}$$

where m is the hierarchical level associated to j. As an example Table 5 shows the first cell codes obtained by resampling cell 4 in a 2D $\mathcal{C}$-space.

i	1	2	3	4	5	6	7	8	9	10
$r_2[i]$	17	20	19	18	69	81	77	73	72	84
i	11	12	13	14	15	16	17	18	19	$\cdots$
$r_2[i]$	80	76	71	83	79	75	70	82	78	$\cdots$

Table 5: *First samples obtained by resampling cell 4 (see Figure 1 for their location in $\mathcal{C}$-space).*

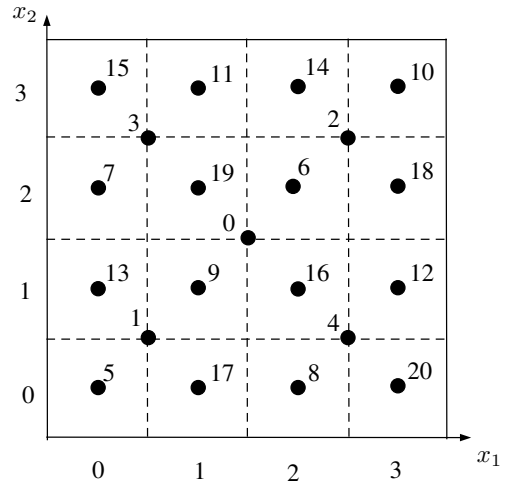

Figure 2: *Sampled configurations of a 2D $\mathcal{C}$-space located at the center of the sampled cells. The configuration labels indicate the index in the generation sequence.*

4 Evaluation

The evaluation of the proposed deterministic sequence has been carried out by implementing a PRM-like planner that uses the samples generated by the deterministic sequence. A basic PRM planner has also been implemented to compare the results. A bend-corridor problem in two, three and six dimensions has been selected as a test-bed. Figure 3 shows the bend-corridor problem for $d = 2$ and the solution path found using deterministic sampling.

First results show a good performance of the deterministic sequence in comparison to the probabilistic one. An exhaustive experimentation is currently being carried out by considering different sizes of the width corridor and different neighborhood distances and policies in the definition of neighbors.

5 Conclusions

Sampling-based path planners have proven to be the best current alternative to solve difficult path planning problems with many degrees of freedom. A crucial fac-

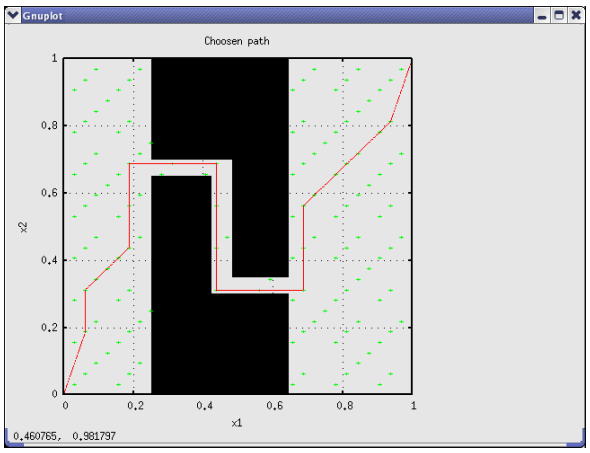

Figure 3: *The bend-corridor problem for* $d = 2$.

tor in the performance of those planners is how samples are generated. Sampling sequences should satisfy the following requirements. An uniform coverage that can be incrementally improved as the number of samples increases, a lattice structure that reduces the cost of computing neighborhood relationships, and a locally controllable degree of resolution that allows to generate more samples at the critical regions. Moreover, computationally efficient algorithms are highly desirable.

The proposed deterministic sampling sequence satisfies all these requirements and is, therefore, a good tool to be incorporated to any sampling-based motion planner. Its use in a PRM-like planner has been implemented for a bend-corridor problem with 2, 3 and 6 degrees of freedom, giving promising results. The sequence is also currently being applied for the sampling and classification of cells of a motion planner based on a combination of harmonic functions and a sampling-based scheme [12], and as the basis for the sampling in $SE(3)$ for rigid-body motion planning [11].

References

[1] D. Aarno, D. Kragic, and H. I. Christiansen. Artificial potential biased probabilistic roadmap method. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 461–466, 2004.

[2] R. Bohlin and L. Kavraki. Path planning using lazy prm. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, volume 1, pages 521–528, 2000.

[3] V. Boor, M. H. Overmars, and A. F. van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1018–1023, 1999.

[4] M. S. Branicky, S. M. LaValle, K. Olson, and L. Yang. Quasi-randomized path planning. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1481–1487, 2001.

[5] I. Gargantini. An effective way to represent quadtrees. *Communications of the ACM*, 25(12):905–910, 1982.

[6] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. K. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. volume 12, pages 566–580, August 1996.

[7] J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 995–1001, 2000.

[8] S. M. LaValle, M. S. Branicky, and S. R. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *Int. J. of Robotics Res.*, 23(7-8):673–692, 2004.

[9] S. R. Lindemann and S. LaValle. *Proc. Int. Symp. on Robotics Research*, chapter Current issues in sampling-based motion planning. Springer-Verlag, 2004.

[10] S. R. Lindemann, A. Yershova, and S. M. LaValle. Incremental grid sampling strategies in robotics. In *Proc. of the Sixth Int. Workshop on the Algorithmic Foundations of Robotics*, 2004.

[11] J. Rosell. Sampling SE(3) with a deterministic sequence for 3D rigid-body path planning. Accepted to the 2005 IEEE Int. Conf. on Robotics and Automation.

[12] J. Rosell and M. Heisse. Path planning using harmonic functions and probabilistic cell decomposition. Accepted to the 2005 IEEE Int. Conf. on Robotics and Automation.

[13] G. Sánchez and J.-C. Latombe. On delaying collision checking in PRM planning: application to multi-robot coordination. *The Int. J. Robotics Res.*, 21(1):5–26, Jan. 2002.

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