An Adaptative Deterministc Sequence for Sampling-based Motion Planners

Jan Rosell and Pedro Iñíguez

IOC-DT-P-2004-12
Setembre 2004

An Adaptive Deterministic Sequence for Sampling-based Motion Planners

Jan Rosell and Pedro Iñiguez

Abstract

This paper presents a deterministic sequence with good and useful features for sampling-based motion planners. On the one hand, the proposed sequence is able to generate samples over a hierarchical grid structure of the $\mathcal{C}$-space in an incremental low-dispersion manner. On the other hand it allows to locally control the degree of resolution required at each region of the $\mathcal{C}$-space by disabling the generation of more samples where they are not needed. Therefore, the proposed sequence combines the strength of deterministic sequences (good uniformity coverage), with that of random sequences (adaptive behavior).

Index Terms

Deterministic sampling, Path planning.

I. Introduction

Sampling-based motion planners, like Probabilistic Roadmap Methods (PRMs) [8], or those based on the Rapidly-exploring Random Trees (RRT) [9], 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 [15], or around the initial and goal configurations [14]. 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 $\mathcal{C}$-space. For path planning purposes the uniform coverage is usually evaluated with the metric-based measure of dispersion (loosely, 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 [11]. Deterministic sampling sequences applied to PRM-like planners are demonstrated in [10] 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 [5], 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 based on a 2^d-tree decomposition of a d-dimensional $\mathcal{C}$-space with an adequate code convention that labels and locates each cell of the tree. The proposed sequence has the advantages of deterministic sequences and also allows to locally control the degree of resolution required at each region of the $\mathcal{C}$-space, by disabling the generation of more samples where they are not needed (and thus concentrating samples in critical regions).

The paper is structured as follows. Section II presents the 2^d-tree decomposition and the code convention used to label and locate the cells. Section III introduces the generation of the deterministic sampling sequence and shows its main features and potential use in path planning problems. Finally, Section IV concludes the work.

II. $\mathcal{C}$-space decomposition

The following 2^d-tree decomposition of a d-dimensional $\mathcal{C}$-space is considered [12]. 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. Partition levels will be denoted by super-indices: a cell of a

Manuscript submitted to IEEE-TR June 23, 2004.
This work was partially supported by the CICYT projects DPI2001-2202 and DPI2002-03540
Jan Rosell is with the Institute of Industrial and Control Engineering at the Technical University of Catalonia, Barcelona, Spain
Pedro Iñiguez is with the Dept. of Electronics, Electrics and Automatic Engineering at the Rovira i Virgili University, Tarragona, Spain

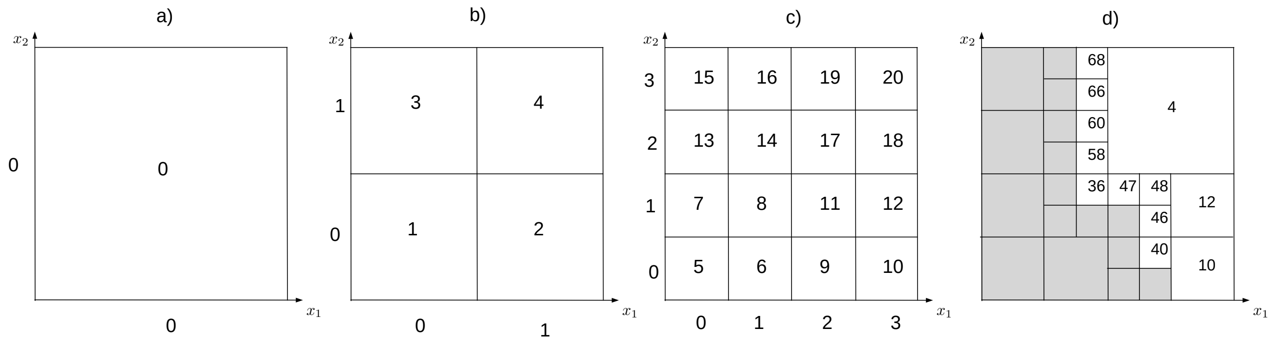

Fig. 1. Cell codes for different levels in the hierarchy in a 2D $\mathcal{C}$-space: a) level 0, b) level 1, c) level 2; and d) An example of the codes of a subset of cells of a quadtree that correspond to $\mathcal{C}_{free}$.

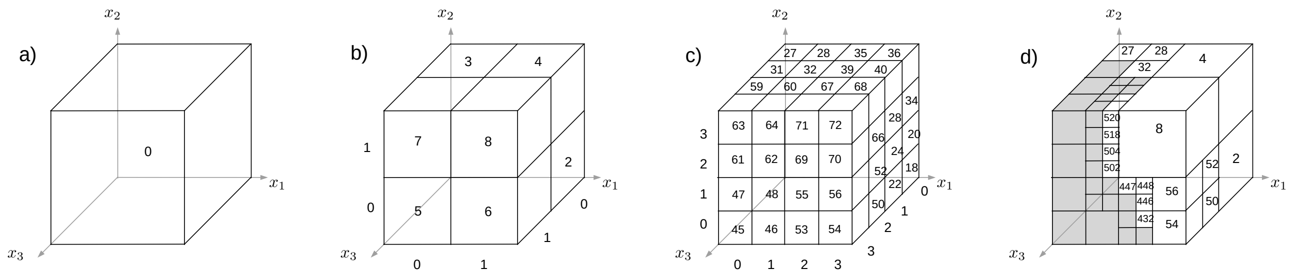

Fig. 2. Cell codes for different levels in the hierarchy in a 3D $\mathcal{C}$-space: a) level 0, b) level 1, c) level 2; and d) An example of the codes of a subset of cells of an octree that correspond to $\mathcal{C}_{free}$.

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 [6] 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:

$$C^m_{ini} = \frac{2^{dm} - 1}{2^d - 1} \tag{1}$$

$$C^m_{end} = 2^d C_{ini} \tag{2}$$

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

A. 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:

- $V^m_{b_k} = (v^m_1, \ldots, v^m_d)$ be the indices of an m-cell, b^m_k, on the regular grid $\mathcal{G}_m$. The values of $v^m_j \ \ \forall j \in 1 \ldots d$ range from 0 to $(2^m - 1)$, e.g. cell code 9 in Figure 1c has indices $(2, 0)$ on grid $\mathcal{G}_2$.
- $(a_{mj} \ldots a_{1j})_2$ be the binary representation[1] of v^m_j, i.e.:

$$v^m_j = (a_{mj} \ldots a_{1j})_2 = \sum_{i=1}^{m} a_{ij}\ 2^{(i-1)} \qquad \forall j \in 1 \ldots d \tag{3}$$

- The conversion table $T(b^m_k)$ of an m-cell of a d-dimensional $\mathcal{C}$-space be the binary $m \times d$ matrix whose columns are the binary representation of $v^m_j \ \ \forall j \in 1 \ldots d$:

[1]When necessary, subindices will be used to show the base in which a number is represented, e.g. binary numbers will be represented with subindex 2.

Code	v_d^m	$\dots$	v_j^m	$\dots$	v_1^m
r_m	a_{md}	$\dots$	a_{mj}	$\dots$	a_{m1}
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
r_i	a_{id}	$\dots$	a_{ij}	$\dots$	a_{i1}
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
r_1	a_{1d}	$\dots$	a_{1j}	$\dots$	a_{11}

Then, given the indices $V_{b_k}^m$ of an m-cell b_k^m, the corresponding code $C_{b_k}^m$ is computed using the following expression:

$$C_{b_k}^m = C_{ini}^m + \sum_{i=1}^{m} r_i 2^{d(i-1)} \tag{4}$$

where C_{ini}^m is the code of the initial cell expressed in equation (1), and $r_i \quad \forall i \in 1 \dots m$ are the rows of $T(b_k^m)$, i.e. $r_i = (a_{id} \dots a_{i1})_2$.

Note that from equation (4), it can be seen that the values $r_i \ \forall i \in 1 \dots m$ are the digits of the representation in base 2^d of $(C_{b_k}^m - C_{ini}^m)$. Taking into account this fact, given the code $C_{b_k}^m$ of a cell b_k^m, the corresponding indices $V_{b_k}^m$ on the grid $\mathcal{G}_m$ can be retrieved from the columns of the conversion table built with the binary representation of the digits of $(C_{b_k}^m - C_{ini}^m)_{2^d}$ as rows.

As an example consider the 2-cell with code 11 in Figure 1c. Its indices are $(v_1^2, v_2^2) = (2, 1) = (10_2, 01_2)$. Then, the conversion table is:

code 11	v_2^2	v_1^2
r_2	0	1
r_1	1	0

Using the values $r_2 = 01_2 = 1$ and $r_1 = 10_2 = 2$, the code can be retrieved as:

$$C_{b_k}^2 = C_{ini}^2 + \sum_{i=1}^{2} r_i 2^{2(i-1)} = 5 + 2 \times 1 + 1 \times 4 = 11 \tag{5}$$

On the other way round, $C_{b_k}^2 - C_{ini}^2 = 11 - 5 = 6 = 12_4$, and therefore its digits are $r_2 = 1 = 01_2$ and $r_1 = 2 = 10_2$. These values correspond to the rows of the conversion table and therefore allow to retrieve the indices of the cell from the corresponding columns.

B. Offspring codes

The descendants of a cell b_k^m are obtained by adding rows to the conversion table $T(b_k^m)$ and letting the bits of the new rows have any value. For instance, the conversion tables of the descendant cells of $C_{b_k}^m = 11$ are:

code 45	v_2^3	v_1^3
r_3	0	1
r_2	1	0
r_1	0	0

code 46	v_2^3	v_1^3
r_3	0	1
r_2	1	0
r_1	0	1

code 47	v_2^3	v_1^3
r_3	0	1
r_2	1	0
r_1	1	0

code 48	v_2^3	v_1^3
r_3	0	1
r_2	1	0
r_1	1	1

Codes of neighboring cells can also easily be computed with the aid of the conversion table.

C. Hierarchical level

The level m of a given cell with code K is found as:

$$m = (\text{int}) \left\{ \frac{1}{d} \log_2[(2^d - 1)K + 1] \right\} \tag{6}$$

For instance, the hierarchical level of a cell with code 14 in Figure 1c is:

$$m = (\text{int}) \left\{ \frac{1}{2} \log_2[(2^2 - 1)14 + 1] \right\} = (\text{int})2.7171 = 2 \tag{7}$$

III. Deterministic Sequence Generation

This section introduces the generation of the sampling sequence that is based on the 2^d-tree code convention. The sequence relies on a predefined ordering of the 2^d descendant cells of any given parent cell.

A. Low-dispersion ordering of descendant cells

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

$$w = \sum_{j=1}^{d} b_j 2^{j-1} \tag{8}$$

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 distance[2] to the previous elements of the sequence.

The set, W_d, of all the 2^d binary words with d bits can be classified in subsets taking into account the number of ones of each word. There are d subsets, W_d^n, of $\binom{d}{n}$ elements each one, being n the number of ones of the words of the subset. Let first consider an ordering of W_d based on this classification. Let $R_d^n(i)$ and $R_d(i)$ be the proposed ordering of W_d^n and W_d, respectively.

The ordering $R_d^n(i)$ can be described by the following recursive expression:

$$R_d^n(i) = \begin{cases} 2^{d-1} + R_{d-1}^{n-1}(i) & \text{if} \quad 0 \leq i < \binom{d-1}{n-1} \\ R_{d-1}^{n}(i - \binom{d-1}{n-1}) & \text{else} \end{cases} \tag{9}$$

with the following particular cases:

$$R_d^n(i) = \begin{cases} 2^d - 1 & \text{if} \quad n = d \\ 0 & \text{if} \quad n = 0 \end{cases} \tag{10}$$

Then, the ordering $R_d(i)$ is:

$$R_d(i) = R_d^n(i-a) \;\; \text{for} \quad a \leq i < a + \binom{d}{n} \tag{11}$$

with:

$$a = \sum_{j=n+1}^{d} \binom{d}{j} \tag{12}$$

As an example consider the following tables corresponding to $R_2(i)$, $R_3(i)$ and $R_4(i)$:

i	$R_2(i)$	
0	$11_2 = 3$	$R_2^2(i)$
1	$10_2 = 2$	$R_2^1(i)$
2	$01_2 = 1$	
3	$00_2 = 0$	$R_2^0(i)$

i	$R_3(i)$	
0	$111_2 = 0$	$R_3^3(i)$
1	$110_2 = 7$	$R_3^2(i)$
2	$101_2 = 5$	
3	$011_2 = 3$	
4	$100_2 = 4$	$R_3^1(i)$
5	$010_2 = 2$	
6	$001_2 = 1$	
7	$000_2 = 0$	$R_3^0(i)$

i	$R_4(i)$	
0	$1111_2 = 15$	$R_4^4(i)$
1	$1110_2 = 14$	$R_4^3(i)$
2	$1101_2 = 13$	
3	$1011_2 = 11$	
4	$0111_2 = 7$	
5	$1100_2 = 12$	$R_4^2(i)$
6	$1010_2 = 10$	
7	$1001_2 = 9$	
8	$0110_2 = 6$	
9	$0101_2 = 5$	
10	$0011_2 = 3$	
11	$1000_2 = 8$	$R_4^1(i)$
12	$0100_2 = 4$	
13	$0010_2 = 2$	
14	$0001_2 = 1$	
15	$0000_2 = 0$	$R_4^0(i)$

Finally, the sequence L_d is found by successively selecting elements of R_d in such a way that the even elements of the sequence are the opposite of the previous odd ones (i.e. by changing ones by zeros and vice-versa), and the subsets W_d^n are properly alternated.

$$L_d(i) = R_d(j_d(i)) \tag{13}$$

with

$$\begin{aligned} j_d(i) &= (2^d - 1)\frac{1-(-1)^i}{2} \\ &+ (-1)^i \left\{ \frac{(\text{int})\frac{i}{2}(-1)^{(\text{int})\frac{i}{2}}}{2} + (2^{d-1} - \frac{1}{2})\left(\frac{1-(-1)^{(\text{int})\frac{i}{2}}}{2}\right) \right\} \end{aligned}$$

[2]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.

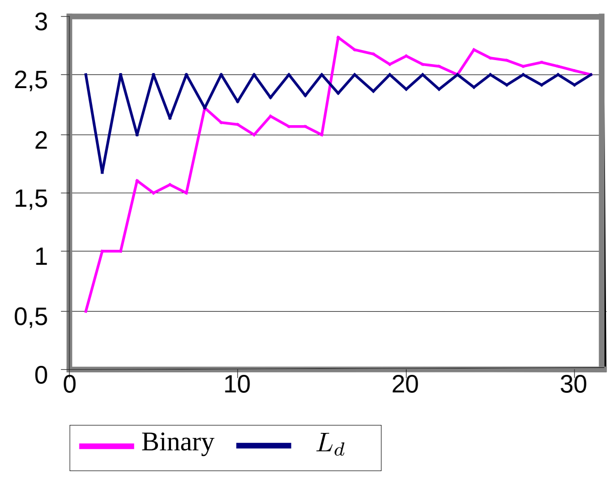

Fig. 3. Evolution of the mean distance of each new added cell of a sequence, for the binary sequences and the L_d sequence, when $d = 5$.

For instance, $j_2(i) = \{0, 3, 1, 2\}$, $j_3(i) = \{0, 7, 3, 4, 1, 6, 2, 5\}$, and $j_4(i) = \{0, 15, 7, 8, 1, 14, 6, 9, 2, 13, 5, 10, 3, 12, 4, 11\}$. The ordering $j_d(i)$ chosen is a simple yet efficient one. Alternative ways of $j_d(i)$ should be considered.

As an example consider the following tables corresponding to $L_2(i)$, $L_3(i)$ and $L_4(i)$:

i	$L_2(i)$
0	$11_2 = 3$
1	$00_2 = 0$
2	$10_2 = 2$
3	$01_2 = 1$

i	$L_3(i)$
0	$111_2 = 0$
1	$000_2 = 7$
2	$011_2 = 3$
3	$100_2 = 4$
4	$110_2 = 6$
5	$001_2 = 1$
6	$101_2 = 5$
7	$010_2 = 2$

i	$L_4(i)$
0	$1111_2 = 15$
1	$0000_2 = 0$
2	$1001_2 = 9$
3	$0110_2 = 6$
4	$1110_2 = 14$
5	$0001_2 = 1$
6	$1010_2 = 10$
7	$0101_2 = 5$
8	$1101_2 = 13$
9	$0010_2 = 2$
10	$1100_2 = 12$
11	$0011_2 = 3$
12	$1011_2 = 11$
13	$0100_2 = 4$
14	$0111_2 = 7$
15	$1000_2 = 8$

The mean Manhattan distance of one cell to all other brother cells (including the zero distance to itself) is:

$$\text{dist} = \frac{1}{2^d} \sum_{j=0}^{d} \binom{d}{j} \times j = \frac{d}{2} \tag{14}$$

Then, when a cell is added to the sequence, it is expected that its mean distance to all the previous cells of the sequence converges to $\frac{d}{2}$. A better sample dispersion is obtained if the variance around the mean value is as small as possible and if the convergence is as fast as possible. Figure 3 compares the evolution of this mean distance for L_d with respect to the binary sequence, $B(i) = i$, when $d = 5$.

B. 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 L_d and results in an incremental low-dispersion covering of the entire $\mathcal{C}$-space.

Let $k \geq 0$ be the index of the sequence, m be the hierarchical level associated to k as expressed in equation (6), r_i be the digits of $(k - C^m_{ini})_{2^d}$, and $L_d(\cdot)$ be the cell ordering of the descendant cells as expressed in equation (13). Then:

$$s_d(k) = C^m_{ini} + \sum_{i=1}^{m} L_d(r_i) 2^{d(m-i-1)} \tag{15}$$

Note that $L_d(r_i)$ are the digits, in the reverse order, of the representation in base 2^d of $(s_d(k) - C^m_{ini})$. As an example the last column of the following table shows the values of $s_2(k)$ for the first 20 cell samples generated on a 2D $\mathcal{C}$-space. Columns are computed from left to right and represent the steps to compute equation (15). Columns 3, 4 and 5 are written in base 4.

k	m	$(r_m \dots r_1) =$ $(k - C^m_{ini})_{2^2}$	$L_2(r_i)$	digit reversal	decimal	$+C^m_{ini}$
0	0	0	0	0	0	0
1	1	0	3	3	3	4
2	1	1	0	0	0	1
3	1	2	2	2	2	3
4	1	3	1	1	1	2
5	2	00	33	33	15	20
6	2	01	30	03	3	8
7	2	02	32	23	11	16
8	2	03	31	13	7	12
9	2	10	03	30	12	17
10	2	11	00	00	0	5
11	2	12	02	20	8	13
12	2	13	01	10	4	9
13	2	20	23	32	14	19
14	2	21	20	02	2	7
15	2	22	22	22	10	15
16	2	23	21	12	6	11
17	2	30	13	31	13	18
18	2	31	10	01	1	6
19	2	32	12	21	9	14
20	2	33	11	11	5	10

The coordinates of the configurations corresponding to the center of the sampled cells can easily be obtained from the indices of the cells on the corresponding grids. Let $X^m_{b_k} = (x^m_1, \dots, x^m_d)$ be the coordinates of the center of an m-cell, b^m_k, with indices $(v^m_1, \dots, v^m_d)$. Then:

$$x^m_j = v^m_j s_m + \frac{s_m}{2} \qquad \forall j \in 1 \dots d \tag{16}$$

For instance, the cells with codes 2 and 12 have, respectively, the indices $(1, 0)$ on grid $\mathcal{G}_1$ and $(3, 1)$ on grid $\mathcal{G}_2$. Since $s_1 = 0.5$ and $s_2 = 0.25$, then the resulting coordinates are $(0.75, 0.25)$ and $(0.875, 0.375)$.

C. *Backward sequence*

The backward sequence $s^{-1}_d(C^m_{b_k})$ determines for a given code $C^m_{b_k}$, which is its order in the sampling sequence. It relies on L^{-1}_d, which gives (using the tabulated sequence L_d) the index i that corresponds to a given value $L_d(i)$. Then:

$$s^{-1}_d(C^m_{b_k}) = C^m_{ini} + \sum_{i=1}^{m} L^{-1}_d(r_i) 2^{d(m-i-1)} \tag{17}$$

where in this case r_i are the digits of $(C^m_{b_k} - C^m_{ini})_{2^d}$.

D. *Adaptive behavior*

When a given sampled configuration is generated for path planning purposes (e.g. the center of a cell with code $C^m_{b_k}$), that configuration is evaluated using a collision checker. Then, some environment knowledge (for instance distance information or the number of neighbor configurations belonging to $\mathcal{C}_{free}$) can be used to determine that no further samples are required within that cell. In this case, the backward sequence is used to compute the indices of the sampling sequence that correspond to the descendant cells of $C^m_{b_k}$. These indices are set as disabled indices. Then, when $s_d(k)$ resumes, no samples are generated for these disabled indices.

For instance, if in a 2D $\mathcal{C}$-space the generation of samples is to be disabled on the region covered by cell 4, the codes of its descendant cells are first computed, using the procedure detailed in subsection II-B:

$$\text{Descendant Cells} = \{17, 18, 19, 20, 69, 70, 71, 72, \dots\} \tag{18}$$

Then, those values are applied to the backward sequence to generate the following sequence indices:

$$\text{Disabled Indices} = \{5, 13, 9, 17, \dots\} \tag{19}$$

When $s_d(k)$ resumes, the samples corresponding to those disabled indices are skipped. Figure 4 shows the sampled configurations of a 2D $\mathcal{C}$-space generated using $s_2(k)$ with $0 \le k \le 18$ and $k \ne 5 \ne 9 \ne 13 \ne 17$.

Later on, perhaps due to a change in the environment, there may be the need to locally generate samples on a previously disabled cell $C^m_{b_k}$. Those samples can be generated by applying the sampling sequence of equation (15), considering level m as the initial level 0 and using the following initial cell codes:

$$C^0_{ini} = C^m_{b_k} \tag{20}$$

$$C^{m+1}_{ini} = (2^d C^m_{ini}) + 1 \tag{21}$$

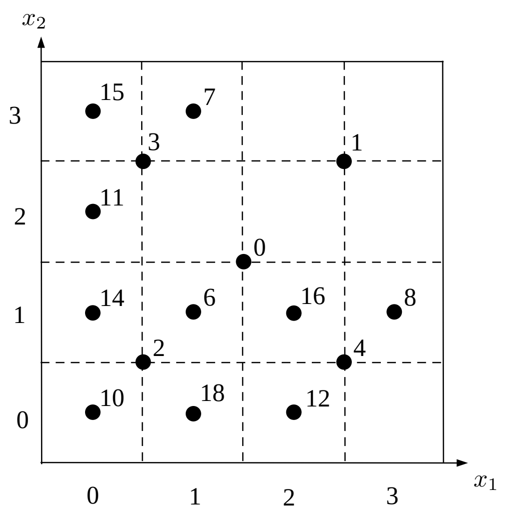

Fig. 4. Sampled configurations of a 2D $\mathcal{C}$-space located at the center of the corresponding sampled cells of the quadtree. The labels indicate the index in the generation sequence (samples of partition level $m \geq 2$ at the top right region have been disabled).

IV. 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 factor 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.

The proposed deterministic sampling sequence satisfy all these three requirements and is, therefore, a good tool to be incorporated to any sampling-based motion planner. Its use in a probabilistic harmonic-function based path planner [7] is giving promising results [13]. Future work include an extensive empirical testing of the proposed sampling sequence on several PRM-like planners, and a comprehensive comparison with other sampling strategies.

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. E. 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. La Valle, 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] P. Dario and editors R. Chatila, editors. *Proc. Eighth Int'l Symp. on Robotics Research*, chapter Current issues in sampling-based motion planning. Springer-Verlag, 2004 to appear.

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

[7] P. Iiguez and J. Rosell. Probabilistic harmonic-function-based method for robot motion planning. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 382–387, 2003.

[8] 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.

[9] 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.

[10] S. M. LaValle, M. S. Branicky, and S. R. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *Int. J. of Robotics Research*, to appear 2004.

[11] S. R. Lindemann and S. M. LaValle. Incremental low-discrepancy lattice methods for motion planning. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 2920–2927, 2004.

[12] J. Rosell and P. Iñiguez. A hierarchical and dynamic method to compute harmonic functions for constrained motion planning. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2334–2340, 2002.

[13] J. Rosell and P. Iñiguez. Extending the use of harmonic functions using probabilistic cell sampling and classification. Technical Report IOC-DT-P-2004-25, UPC, 2004.

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

[15] 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.