Complete coverage by mobile robots using slice decomposition based on natural landmarks

Sylvia C. Wong Bruce A. MacDonald

Department of Electrical and Computer Engineering,
University of Auckland, New Zealand
s.wong, b.macdonald *at* auckland.ac.nz

Abstract. In applications such as vacuum cleaning, painting, demining and foraging, a mobile robot must cover an unknown surface. The efficiency and completeness of coverage is improved by the construction of a map while the robot covers the surface. Existing methods generally use grid maps, which are susceptible to odometry error and may require considerable memory and computation. We propose a new "slice decomposition" ideally suited to coverage by a simple zigzag path. Cell boundaries are large, easily detectable natural landmarks. Therefore, the decomposition is robust against uncertainty in sensors. It can also handle a wider variety of environments. The proposed method has been evaluated using simulation and real robot experiments.

1 Introduction

In a coverage application, a mobile robot must visit all the reachable surface in its environment. While coverage is similar to exploration, an exploring robot moves and sweeps its long range sensors, so as to *sense* all of its environment. During a coverage application, the robot or a tool must *pass over* all floor surface.

If the environment is unknown, the robot must use a strategy that ensures it covers all the space. It must use sensors to gather information about obstacles as it moves, and it must formulate and remember some form of map, so that it may return to areas it has seen but not yet covered.

The algorithmic strategy of "divide and conquer" is a powerful technique used to solve many problems, and many mapping procedures carry out a process of space decomposition, where a complex space is repeatedly divided until simple subregions of a particular type are created. The problem at hand is then solved by applying a simpler algorithm to the simpler subregions. Exact cell decompositions [1] and occupancy grids [2] are examples of such maps. Coverage algorithms commonly use some form of space decomposition as a map, because covered areas can be stored easily by marking individual subregions.

Occupancy grids are a widely used map representation for coverage algorithms. It is straightforward to mark covered areas in a grid map. Zelinsky used the distance transform of a grid map [3]. A coverage path is formed by selecting the unvisited neighbouring cell with the highest distance transform. Unlike other coverage algorithms, a goal location must be selected. Gabriely and Rimon

incrementally subdivide the environment into disjoint grid cells, while following a spanning tree of the partial grid map [4]. A disadvantage of grid maps is the requirement for accurate localisation to create and maintain a coherent map [5]. Grid maps also suffer from exponential growth of memory usage because the resolution does not depend on the complexity of the environment [6]. Also, they do not permit efficient planning through the use of standard graph searches [6].

Exact cell decomposition divides a complex structure S into a disjoint component cells, whose union is exactly S. The boundary of a cell corresponds to a criticality of some sort. Exact cell decomposition methods are commonly used in path planning for point to point tasks. The most common example is trapezoidal decomposition [1]. It is formed by sweeping a line L across the environment, and creating a cell boundary whenever a vertex is encountered. Obstacles are limited to polygons. Therefore, each cell of a trapezoidal decomposition is either a trapezoid or a triangle. For path planning, the decomposition is first reduced to a connectivity graph representing the adjacency relation among cells [1]. The associated connectivity graph is searched to find paths between any two cells.

However, trapezoidal decomposition creates convex cells that are unnecessarily small, and therefore inefficient, for coverage purposes. Some non-convex shapes can also be covered by simple coverage patterns. For example, the two cells on each side of the obstacle in Fig. 1(a) can be merged and a simple zigzag pattern shown in Fig. 1 can still cover the combined cells. Based on merging multiple cells in trapezoidal decomposition, Choset and Pignon proposed the first exact cell decomposition specifically designed for coverage [7]; the boustrophedon decomposition, shown in Fig. 1(b), signifying the relationship between the decomposition and the zigzag. Like trapezoidal decomposition, boustrophedon decomposition is limited to environments with only polygonal objects.

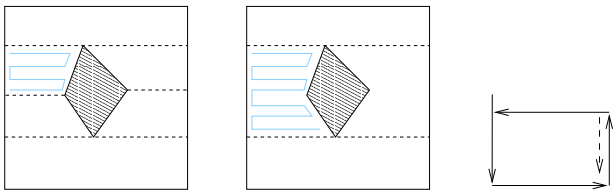


Fig. 1. (a) Trapezoidal decomposition creates cells that are unnecessarily small for coverage tasks. (b) Boustrophedon decomposition reduces the number of cells by combining multiple cells that can be covered by a zigzag. (c) The rectangular coverage pattern used in CC_R and Morse decomposition.

Butler proposed an exact cell decomposition for rectilinear environments, for his coverage algorithm CC_R [8]. Cell boundaries are formed when an obstacle boundary parallel to the sweep line is encountered. While trapezoidal and boustrophedon decompositions cannot handle obstacle surfaces parallel to the

sweep line, the criticality in CC_R is specially defined for rectilinear environments. Another difference is that CC_R is calculated online by contact sensing robots, simultaneously with the coverage process. In other words, an associated coverage algorithm is devised to use a partial cell decomposition for coverage path planning, at the same time updating the map when new information becomes available. Instead of a zigzag, CC_R uses a rectangular coverage pattern that includes retracing, shown in Fig. 1(c). The retracing is added to ensure wall following on both side boundaries, because a contact sensing robot cannot detect obstacles except when wall following. If an opening in the side boundary occurs between consecutive strips of the zigzag, the robot will miss it.

Acar *et al.* introduced Morse decomposition [9] which can handle a larger set of environments than boustrophedon decomposition and CC_R. Cell boundaries in Morse decomposition are critical points of Morse functions. Put simply, a cell boundary occurs when the sweep line encounters an obstacle whose surface normal is perpendicular to the sweep line. Morse decomposition generalises boustrophedon decomposition to include non-polygonal obstacles. However, it cannot handle surfaces parallel to the sweep line. This excludes rectilinear environments. Similarly to CC_R, Morse decomposition also has an online decomposition algorithm. However, Morse decomposition cannot use a zigzag to cover individual cells. It uses the rectangular pattern in Fig. 1(c). The wall following offered by the pattern is needed because critical points occurring on the side boundary cannot be detected even with unlimited range sensors, except when wall following [10]. This is due to the difficulty in detecting critical points of Morse functions.

This paper introduces a new exact cell decomposition for complete coverage path planning, where the decomposed regions are precisely suited to a zigzag coverage pattern, with no retracing. The length of the coverage path is greatly reduced. Cell boundaries are large scale features that have physical extension over time, and can be detected even by noisy and inaccurate sensors. Also, our algorithm works on a larger variety of environments, including both rectilinear and non-rectilinear ones. Obstacles can be polygonal, or curved. Lastly, the cell decomposition can be constructed online, in an unknown environment, while the robot covers the space [11].

Section 2.1 explains the slices and segments created by a sweep line. Section 2.2 defines the criticality for cell boundaries. Section 2.3 presents the slice decomposition algorithm. Section 2.4 discusses the effects of step size and sweep direction. Section 3 presents results and section 4 discusses the work.

2 Slice Decomposition

2.1 Slice and segments

A slice decomposition is created by sweeping a line from the top of an environment to the bottom. There are two types of region — obstacle and free space. At any time, the sweep line intersects a number of free space and obstacle regions determined by the topology of the environment and position of the sweep line.

We call the arrangement of regions intersected by the sweep line a *slice* and the regions within *segments*.

Fig. 2(a) shows an obstacle with the sweep line at two different positions. The slices created are shown on the right; at position 1 the slice contains one free space segment, an obstacle segment, and then another free space segment. The slice at position 2 has three free space segments and two obstacle segments.

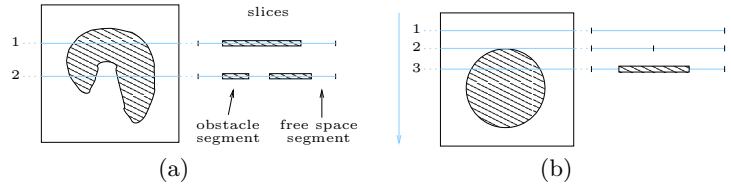


Fig. 2. (a) The arrangement of segments in slices made by the sweep line changes as it sweeps through the environment. (b) The number of segments present in a slice changes as the sweep line enters an obstacle.

In Fig. 2(b), at position 1, the slice contains only one free space segment. Obstacle segments begin to emerge at position 2, where the sweep line first intersects with the object.

The sweep line can be viewed as a ray passing through the segments on a slice. The ray intersection test [12] shows that every time an intersection is made, the line is in a different type of region. This guarantees that each segment is bounded by two intersection points, and also implies that the sweep line always has an even number of intersections on the slices, since the ray always starts and ends in the obstacle region outside the boundary.

2.2 Criticality

Two slices S_a and S_b are consecutive if they are from sweep line positions one time step apart. If the sweep line moves by a distance δx for each time step, and the slices S_a and S_b are from positions x_a and x_b respectively, then slice S_a and slice S_b are consecutive slices if and only if $\mid x_a - x_b \mid = \delta x$.

Cell boundaries occur when there is an abrupt change in the topology between segments in consecutive slices. There are two situations where this can happen:

1. A segment in the previous slice is split by the emergence of a new segment.
   - An obstacle segment emerges within a free space segment, as in Fig. 3(a).
   - A free space segment emerges within an obstacle segment, as in Fig. 3(b).
2. A segment from the previous slice disappears in the current slice.
   - An obstacle segment disappears, as in Fig. 3(c).
   - A free space segment disappears, as in Fig. 3(d).

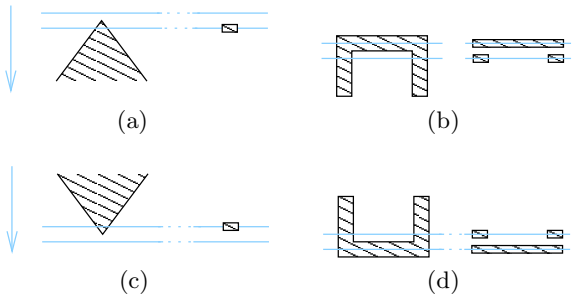


Fig. 3. (a), (b) One segment splits into multiple segments. (c), (d) Multiple segments merge into a single segment.

2.3 Decomposition Algorithm

The slice decomposition is formed by maintaining a list D of active obstacle and free space cells with segments present on the slices created by the sweep line as it sweeps through the environment, summarised in Algorithm 1. The history of list D, ie all the cells that have appeared in D, forms the decomposition. The sweep stops to process and update list D whenever a criticality occurs.

Algorithm 1 Offline Slice Decomposition

1: $c \in$ {free space cell, obstacle cell}
2: **for all** time t **do**
3: Move sweep line downwards by δx
4: $D_{t-1} = (\ldots, c_{i-2}, c_{i-1}, c_i, c_{i+1}, c_{i+2}, \ldots)$
5: **for all** segments in D_{t-1} **do**
6: **if** emergence inside c_i **then**
7: $(c_i) \leftarrow (c_{e-1}, c_e, c_{e+1})$
8: $D_t = (\ldots, c_{i-2}, c_{i-1}, c_{e-1}, c_e, c_{e+1}, c_{i+1}, c_{i+2}, \ldots)$
9: **if** c_i disappears **then**
10: $(c_{i-1}, c_i, c_{i+1}) \leftarrow (c_d)$
11: $D_t = (\ldots, c_{i-2}, c_d, c_{i+2}, \ldots)$

The algorithm has two loops, one for moving the sweep line from top to bottom (line 2), the other for inspecting segments in the previous and the current slice for topology changes (line 5). At line 1 are specified all cells that are either free space cells or obstacle cells. Within the first loop, line 3 shows that the sweep line is moved by δx for each time step. Line 4 gives the format of the list D at the previous time step D_{t-1}. Lines 6 and 9 within the inner loop correspond to the two cases of criticality. For segment emergence (line 6), the segment that is split into two halves is replaced by three separate segments (line 7). The three segments belong to new cells and are therefore given new cell IDs, c_{e-1}, c_e, c_{e+1}. These new cell IDs identifying this slice contain a cell boundary. Line 8 shows

the list D_t after the changes. The updates for segment disappearance are shown in lines 9 to 11. The cell that contains the disappeared segment, along with its two neighbours, are replaced in D by a single new cell (line 10). Line 11 shows the list D_t after the changes.

In the example is in Fig. 4, f_n are free space cells and o_n are obstacle cells. Initially, the sweep line intersects only the first free space cell f_1, giving just that one space cell, $D_t = (f_1)$. At the first event, an obstacle segment emerges and the first cell f_1 is split. The decomposition D_t then changes to contain three cells – a free space cell, an obstacle cell and another free space cell, $D_t = (f_2, o_1, f_3)$. Then obstacle cell o_1 is split when a free space cell emerges. The decomposition D_t changes to contain five cells, $(f_2, o_2, f_4, o_3, f_3)$. Next D_t changes to three cells, (f_5, o_3, f_3), as the left side bulge is passed. Finally the decomposition D_t contains only one free space cell f_6 when the sweep line exits the obstacle.

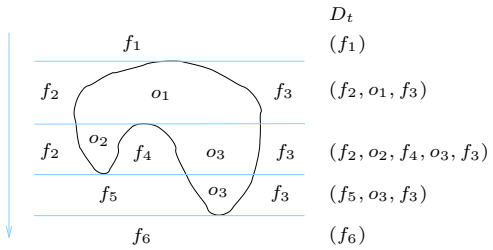


Fig. 4. An example of slice decomposition.

The algorithm tracks both free space and obstacle cells, although only the free space cells are of interest, since mobile robots cannot move inside obstacles.

2.4 Effect of Step size and sweep direction

Since slice decomposition uses a discrete line sweep process, the step size between consecutive slices therefore affects the decomposition yield for a given environment. In practice the step size is determined by the width of the robot, to ensure no space is left uncovered in consecutive sweeps. If the step size is reduced to be infinitesimally small, $\delta x \to 0$, then the sweeping process becomes a continuous sweep, like other exact cell decompositions. However, slice decomposition also works for step sizes larger than infinitesimal.

To capture all cells in a particular environment, the maximum step size has to be smaller than the height of the smallest cell

$$\delta x \leq \min h(c_i) \tag{1}$$

δx is the step size of the line sweep and $h(c_i)$ is the height of the i-th cell. Equation 1 guarantees that all cells will be present in at least one slice.

Fig. 5 illustrates the effect of varying the step size, on the decomposition created. When the steps are small, all cells in the environment are captured. For example, in Fig. 5(a), the step size is small enough to guarantee a sweep line to pass through the small cell between the two lobes at the top of the obstacle. When the step size is increased to the height of the smallest cell, ie $\delta x = \min h(c_i)$, the second sweep position in Fig. 5(a) just barely touches the cell. If the step size is further increased, the smallest cell may be missed entirely, as is the case in Fig. 5(c).

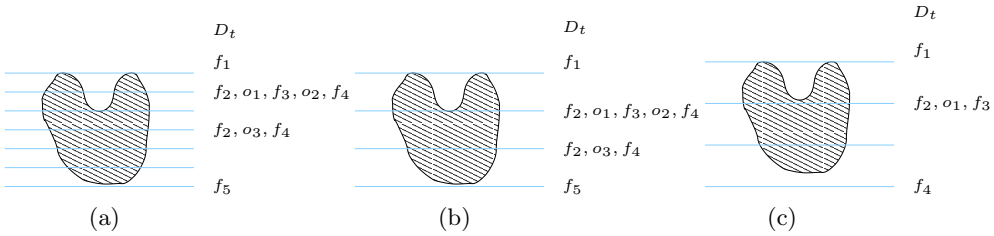


Fig. 5. Effect of step size on decomposition produced. All sweep lines are assumed to be slightly above the obstacle surface they are touching. The list of cells on the right shows where changes (criticalities) occur. (a) $\delta x = \frac{1}{2} \times \min h(c_i)$, (b) $\delta x = \min h(c_i)$, (c) $\delta x > \min h(c_i)$.

When equation 1 is satisfied, the decompositions created are independent of differences in the step size. Compare the slice decomposition in Fig. 5(a) and 5(b). Although the cells are discovered at different positions, the overall transitions of the list D are the same.

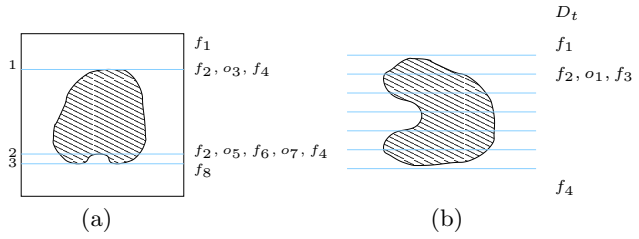


Fig. 6. (a) Forward and reverse sweep yield the same slice decomposition. (b) Rotation changes slice decomposition.

The decomposition created is the same whether the sweeping is in the forward (top to bottom) or the reverse (bottom to top) direction. The decomposition is dependent only on the position of the sweep lines, as illustrated in Fig. 6(a). It shows the same sweep line positions as Fig. 5(a), but the obstacle is upside down.

The topology changes in the list D_t are essentially the same in both figures. The only change is to the numbering of cells.

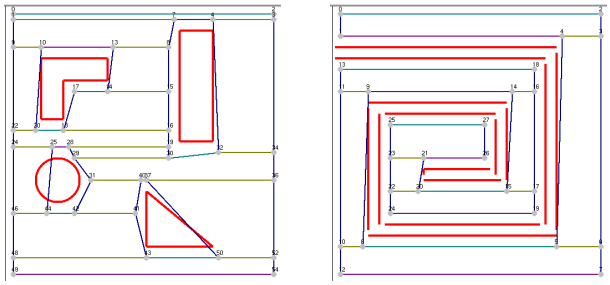
However, if the environment is rotated, the decomposition will be different. Equation 1 guarantees the same decomposition being created only for a particular sweep angle. Fig. 6(b) shows the same obstacle as in Fig. 5, but rotated 90°. It can be seen that the decomposition will be different from that given in Fig. 5 no matter how small the step size is. This is not a shortcoming of a discrete sweep algorithm because continuous sweep based exact cell decomposition, such as trapezoidal decomposition, is also affected by rotational transforms.

3 Results

Slice decomposition was evaluated both in simulation and with a Khepera robot. In all the experiments, the environment is unknown to the robot and the slice decomposition is created online during the coverage process [11].

Fig. 7(a) shows a "normal environment" and slice decomposition. The simulated robot creates a topological map that embeds the decomposition. The horizontal edges of the topological map correspond to the cell boundaries. The environment is divided into 12 free space cells. Some vertical edges cross over the obstacles, because they are simply drawn as straight lines linking their nodes.

Fig. 7(b) shows a more unusual arrangement of obstacles. The free space in the spiral is divided into 9 free space cells.



(a) (b)

Fig. 7. Slice decomposition for (a) a normal environment, (b) a spiral environment.

Slice decomposition was also implemented and tested on the 53mm diameter Khepera robot [11]. It has 8 infra-red sensors which can detect objects up to 30 to 40mm away, and optical wheel encoders for dead reckoning. Fig. 8(a) shows the area covered and Fig. 8(b) the path the robot took in one of the experiments. The cell boundaries of the slice decomposition are shown in Fig. 8(c).

The step size Δx is set to the diameter of the robot, since we want to cover all the surface between consecutive strips of the zigzag. Since the robot is smaller than the free space cells, the step size Δx is always smaller than the height of

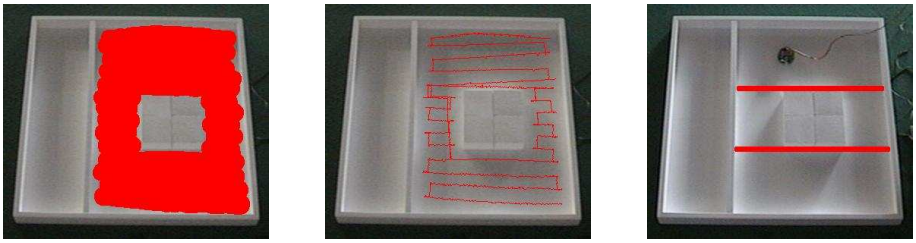


Fig. 8. Coverage with a Khepera robot (a) area covered (b) path taken (c) slice decomposition created

the smallest cell; equation (1) is satisfied and all features in the environment are captured. If the robot is larger than some of the cells, then it cannot enter and cover these cells. The slice decomposition created by such a robot will therefore not have a representation of these cells.

4 Discussion

Criticalities in exact cell decompositions are usually defined as small features, such as vertices in trapezoidal decomposition [13] and critical points in Morse decomposition [9]. In comparison, criticality in slice decomposition is defined using large features, segments. For example, obstacle segments are detected as proximity to obstacles along the sweep line [11]. These large features have physical attributes that are detectable over time. Spurious sensor errors are filtered out through averaging. As a result, the detection becomes robust against noisy and inaccurate sensing [14].

Trapezoidal decomposition forms regions more frequently than slice decomposition, by dividing the space as the sweep line crosses every vertex. While the larger regions formed by slice decomposition may not be convex, the regions are still covered by a simple zigzag algorithm, since the non–convex sides of the space are perpendicular to the zigs and zags.

The concept of non-zero step sizes is incorporated in slice decomposition. If the robot moves in a zigzag path to cover individual cells in the decomposition, then the long strips in the zigzag are the sweep lines. The distance between strips in the zigzag path becomes the step size in the slice decomposition.

Since mobile robots cannot move inside obstacles, some free space cells must be swept in the reverse direction, for example in the L-shaped obstacle of Fig. 7(a).

Slice decomposition can handle a larger variety of environments. Boustrophedon decomposition can only handle polygonal obstacles. CC_R can only handle rectilinear environments. Morse decomposition is more general and can handle obstacles with smooth surfaces, but is only defined for non-rectilinear environments because boundaries parallel to the sweep line are degenerate cases for Morse functions. In comparison, slice decomposition is defined on changes in the

topology of slices. It can handle any environment with polygonal and smooth-surfaced objects, including rectilinear ones, for example that shown in Fig. 7(b).

5 Conclusion

This paper presents a new exact cell decomposition for coverage. Slice decomposition uses changes in topology to decompose an environment, where each cell intersects with the sweep line twice as it passes over. Cells formed can be covered by a zigzag. Our work uses large features for defining cell boundaries and can detect boundaries robustly. It also can cover a wider variety of environments. The decomposition is tested with simulation and real robot experiments.

References

1. Latombe, J.C.: Robot Motion Planning. Kluwer (1991)
2. Elfes, A.: Sonar-based real-world mapping and navigation. IEEE Journal of Robotics and Automation **RA-3** (1987) 249–265
3. Zelinsky, A.: A mobile robot exploration algorithm. IEEE Transactions on Robotics and Automation **8** (1992) 707–717
4. Gabriely, Y., Rimon, E.: Spiral-STC: An on-line coverage algorithm of grid environments by a mobile robot. In: Proceedings of the IEEE International Conference on Robotics and Automation, Washington, DC (2002) 954–960
5. Castellanos, J.A., Tardós, J.D., Schmidt, G.: Building a global map of the environment of a mobile robot: The importance of correlations. In: Proceedings IEEE International Conference on Robotics and Automation (ICRA). Volume 2. (1997) 1053–1059
6. Thrun, S.: Learning metric-topological maps for indoor mobile robot navigation. Artificial Intelligence **99** (1998) 21–71
7. Choset, H., Pignon, P.: Coverage path planning: The boustrophedon decomposition. In: Proceedings of the International Conference on Field and Service Robotics, Canberra, Australia (1997)
8. Butler, Z.J., Rizzi, A.A., Hollis, R.L.: Contact sensor-based coverage of rectilinear environments. In: Proceedings IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics. (1999) 266–271
9. Acar, E.U., Choset, H., Rizzi, A.A., Atkar, P.N., Hull, D.: Morse decompositions for coverage tasks. International Journal of Robotics Research **21** (2002) 331–344
10. Acar, E.U., Choset, H.: Sensor-based coverage of unknown environments: Incremental construction of morse decompositions. International Journal of Robotics Research **21** (2002) 345–366
11. Wong, S.C., MacDonald, B.A.: A topological coverage algorithm for mobile robots. In: Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Volume 4., Las Vegas, Nevada (2003) 1685–1689
12. Foley, J.D., et al.: Computer graphics : principles and practice. Second edn. Addison-Wesley (1990)
13. Chazelle, B.: Approximation and decomposition of shapes. In Schwartz, J.T., Yap, C.K., eds.: Algorithmic and Geometric Aspects of Robotics. Lawrence Erlbaum Associates (1987) 145–185
14. Mataric, M.J.: Integration of representation into goal-driven behavior-based robots. IEEE Transactions on Robotics and Automation **8** (1992) 304–312