SHORTEST PATH PLANNING WITH DOMINANCE RELATION[1]

Sungtaeg Jun and Kang G. Shin

Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109–2122

ABSTRACT : Although SPP is generally regarded as a solved problem in 2D space, few of the existing 2D solutions can be applied to 3D. Since many of the real world applications are based on 3D or higher space, this deficiency severely limits the applicability of 2D solutions. In this paper, we present a new method of partitioning the workspace using the *rectilinear visibility* in 3D or higher space. Unlike the case of 2D space where the shape of a partition in 2D is a rectangle, the shape of a partition in 3D or higher space is arbitrary. In spite of the arbitrary shape of partitioned regions, we first prove that there exist dominance relations between regions. This relation is then utilized to efficiently solve the SPP problem in 3D or higher space.

1 INTRODUCTION

One of the main problems in achieving automatic task scheduling is Automatic Path Planning (APP) in the presence of obstacles in the workspace. Applications of APP are quite diverse, the most notable being automatic path generation for mechanical manipulators [9, 7] and/or autonomous vehicles [13]. Another important application is automatic channel routing in VLSI design [10] and computer networks [3]. In particular, the APP of mechanical manipulators, known as robot motion planning, is quite a challenging task due to their large number of degree of freedom (DOF). Another factor to be considered is the various costs and constraints associated with a path, such as length, safety, and time of completion. In this paper, we develop a new technique for managing these costs and constraints.

APP usually deals with an object to be moved and a workspace cluttered with obstacles. The goal of an automatic path planner is to find a path for the object from an origin to a destination without colliding with any of the obstacles while optimizing a certain criterion function. Some of the most widely used criteria are traveling distance [2, 8], clearance from the obstacles [11], and combination of distance and clearance [12]. In particular, the APP that minimizes the traveling distance is called the *Shortest Path Planning* (SPP) Problem while the other APPs are usually called the *Find Path* (FP) Problem.

One problem in solving the SPPP is the various shapes of moving objects. The specific solution for a specific shape usually does not apply to other shape. To remedy this problem, the Configuration Space Approach (CSA) was proposed in [8, 1]. In the CSA, the origin and the destination are represented as configuration vectors, not as Cartesian positions. Thus, a moving object is represented as a point along with the forbidden configurations due to constraints, i.e., extended obstacles. With the development of CSA, many issues associated with FPP can be resolved with the solution techniques developed for SPP.

The most popular solution to the SPP hinges on the visibility graph (VG). The VG method is based on the premise that when two points in a plane are not visible from each other, the shortest path always contain one or more vertices of the obstacle in the plane. Following this premise, the workspace is transformed into a graph in which the distances between all pairs of mutually visible vertices are precalculated. The optimal solution can then be obtained using Dijkstra's graph search algorithm [5]. Though the VG is very useful in 2D, it is very difficult to use in higher dimensional problems.

In this paper, we shall introduce the L_1 *visibility* between two points in a digitized workspace, based on which we can derive dominance relations between certain partitions of the workspace. These dominance relations show some useful properties that can be utilized to solve the SPP.

The paper is organized as follows. Section 2 states the SPP formally. In Section 3 we define L_1 visiblity and demonstrate how it partitions the workspace. In Section 3.1, the properties of a partitioned workspace are examined. Section 3.2 presents a graph representation of the workspace based on which an SPP solution algorithm is derived. Section 4 presents an example and simulation results. The paper concludes with Section 5.

2 PROBLEM STATEMENT

Consider the problem of moving an object in a workspace cluttered with obstacles. We want to find a path, or determine a set of points, for the object to traverse from a starting point (origin) to an end point (destination) without colliding with any obstacle in the workspace. There are two sources of difficulty associated with this problem: (i) an infinite number of paths exist for each given origin–destination pair, and (ii) it is in general difficult to represent obstacles of arbitrary shape in the workspace. One way of circumventing these sources of difficulty is to divide the workspace into a finite number of cells[2]. Such division not only reduces the infinite number of possible paths to a finite number of paths, but also allows each obstacle to be represented by the set of cells it occupies.

Let the workspace be divided into $\ell \times m \times n$ identical cells. According to the CSA [8], the object to be moved can be shrunk to a point by growing obstacles. In what follows, cells are represented as $o, p, q, \ldots$ when their locations need not be specified, as $o_{ijk}, p_{\ell mn}, q_{abc}, \ldots$ when their locations need to be specified, and as $v_1, v_2, v_3, \ldots$ when a sequence of cells needs to be specified. Informally, the goal of a path planner is to find a path formed by a sequence of neighboring free (unoccupied) cells from the origin to the destination while minimizing a certain path cost.

[1] The work described in this paper was supported in part by the U.S. Airforce Office of Scientific Research under contract No. F33615–85–C–5105 and the National Science Foundation under grant DMC–8721492.

[2] A cell is a square in 2D and a cube in 3D.

CH2876-1/90/0000/0138$01.00 © 1990 IEEE

The *cost* of a path P, denoted by $C(P)$, is defined to be the length of P measured in L_1–metric. Two cells are said to be *neighbors* if they are physically adjacent. Since all the cells are identical, the L_1–distance between the centers of any two neighboring cells are identical and will be treated as *unit distance*. The path planning problem can now be stated formally as follows:
For given two points, x and y, and a set, $\mathcal{O}$, of cells that are occupied by obstacles, find a sequence, $P = v_1 v_2 \dots v_n$, of neighboring cells such that

$$x \in v_1,\ y \in v_n,\ d_1(v_i, v_{i+1}) = 1, v_i \notin \mathcal{O} \text{ for } 1 \le i \le n-1,$$

while minimizing n.

3 PARTITIONING THE WORKSPACE

It is necessary to define the following terms for clarity of presentation.

Definition 1 In L_1–metric space, a cell v is said to be *visible* from a cell w, denoted by $v \Re w$, if there exists a sequence $P = v_0 v_1 \dots v_{d_1(v,w)}$ of free cells such that $v_0 = v$, $v_{d_1(v,w)} = w$, $d_1(v_{i-1}, v_i) = 1$, and $d_1(v_i, v_{d_1(v,w)}) = d_1(v,w) - i$, $1 \le i \le d_1(v,w)$, where $d_1(u,v)$ is the L_1–distance between u and v. Otherwise, v is said to be *not visible* from w, denoted by $v \not\Re w$.

Definition 2 A set, $N = \{n_x^+, n_x^-, n_y^+, n_y^-, n_z^+, n_z^-\}$, is called the set of *neighbor operators* if

$$\begin{aligned} n_x^+(v_{ijk}) = v_{\ell mn} &\implies \ell = i+1,\ m = j,\ n = k \\ n_x^-(v_{ijk}) = v_{\ell mn} &\implies \ell = i-1,\ m = j,\ n = k \\ &\vdots \\ n_z^-(v_{ijk}) = v_{\ell mn} &\implies \ell = i,\ m = j,\ n = k-1. \end{aligned}$$

and a set, $O \subset N$, is called the *orthogonal* set of neighbor operators if they always generate the neighbors in orthogonal directions of a cell, e.g., $\{n_x^+, n_y^-, n_z^+\}$.

Definition 3 For any two free[3] cells, v and w, in the workspace, v is said to be *visible* from w if there exists an orthogonal set, O, of neighbor operators such that $P = v_0 v_1 \dots v_{d_1(v,w)}$ where $v_0 = v$, $v_{d_1(v,w)} = w$, $v_i = n(v_{i-1})$ for some $n \in O$, and O is called the *generating operator set* of P. The dual of O, denoted by O^*, is the generating operator set of the reverse sequence of P, i.e., $v_{d_1(v,w)} v_{d_1(v,w)-1} \dots v_0$.

Using the above definition of visibility, the *dominance* relation between cells and that between two sets of cells are defined as follows.

Definition 4 For any two cells u and v in a workspace W, u is said to dominate v, denoted by $u >_c v$, iff $w \Re v \rightarrow w \Re u, \ \forall\ w \in W$. Similarly, for any two sets, A and B, of cells in W, A is said to dominate B, denoted by $A >_s B$, iff for any $v \in B$, $\exists u \in A$ such that $u >_c v$.

Having defined the dominance relation between cells, the *equal* relation is defined as follows.

Definition 5 For any two cells u and v, u is said to be *equal* to v, denoted by $u \sim_c v$, iff $u >_c v$ and $v >_c u$.

[3] A cell is said to be *free* if it does not intersect any obstacle.

[4] It is reflexive, symmetric, and transitive.

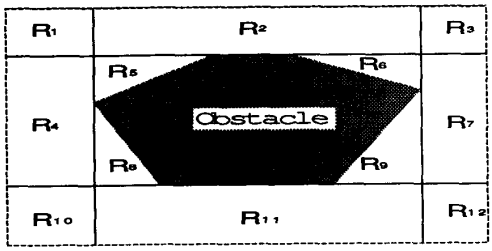


Figure 1: Partitioning of the workspace into regions.

Notice that the relation $\sim_c$ is an equivalence relation[4] and its central importance is that it induces a partition of the workspace. That is, the relation $\sim_c$ divides the workspace into several sets of cells such that $u \in R(v) \rightarrow S(u) = S(v)$, where $R(v)$ is a partition containing v and $S(u)$ the set of all the cells that are not visible from u. Such a set will henceforth be referred to as a *region*. In the 2D example of Fig. 1, any cell in R_1 is visible from any other cell in the entire workspace except for those in R_9. Similarly, no cell in R_2 is visible from any cell in $R_8 \bigcup R_9 \bigcup R_{11}$.

There are several ways of obtaining regions. In case of 2D space, a border of the regions is formed by projecting the edges of obstacles along x and y directions, as shown in Fig. 1. The following procedure, **P1**, is used to determine the total number of cells visible from a given cell. Since total numbers of cells visible from two equivalent cells are the same, **P1** can be used to partition the workspace under the assumption that no two neighboring regions have the same number of visible cells. (This assumption can be relaxed trivially as we shall see shortly.) Informally, the procedure works as follows. All the cells visible from a given cell, v, of the workspace are obtained and expressed with an indicator vector I, i.e., $I(x) = 1$ (0) if a cell x is free (occupied by an obstacle). Starting with a cell v of distance 0 (i.e., itself), one can determine all the visible cells of distance 1 from v. Using a recursion, one can then calculate all the cells visible from v of distance $2, 3, \dots K$, where K is the maximum possible distance between any two cells in the workspace. After determining all the cells visible from a given cell v, the total number, $N(v)$, of cells visible from v is obtained from the vector I.

Procedure **P1**
For every cell v in the workspace W
  if v is a free cell
  begin
    initialize $I(w) \leftarrow 0$ for all $w \in W$
    $I(v) \leftarrow 1$
    for $i = 1$ **to** K
    begin
      Generate $D_i(v)$ which is the set of cells of distance i from v.
      for every $w \in D_i(v)$

$$I(w) \leftarrow \max_{u \in D_{i-1}(v) \bigcap D_1(w)} I(v)$$

    end

$$N(v) \leftarrow \sum_{w \in W} I(w)$$

  end
end{**P1**}

The output of **P1**, N, is a matrix that contains the total number of cells visible from each cell v. According to **P1**, if the number

of cells visible from any two neighboring cells is different, then the two cells belong to different regions. Another notable fact is that all the boundaries of a region are perpendicular to one or more principal axes. Therefore, the vertices, edges, and surfaces of regions can be determined from N as follows:

1. For any vertex, $\frac{\partial^3 N}{\partial x \partial y \partial z} \neq 0$.
2. For any edge parallel with z-axis, $\frac{\partial^2 N}{\partial x \partial y} \neq 0$. Similarly, edges parallel with x-axis or y-axis can be obtained.
3. For the surfaces that are perpendicular to x-axis, $\frac{\partial N}{\partial x} \neq 0$. Similarly, other surfaces can be determined.

P1 cannot detect the boundary between two adjacent regions when the total number of visible cells for the two regions happens to be the same. Such undetected boundaries can be easily recovered by extending some of detected boundaries.

By partitioning the workspace based on the relation $\sim_c$, the path planning problem can be divided into two subproblems: (i) Path planning between two cells in the same region, and (ii) path planning between two cells in different regions.

3.1 Properties of Partitioned Regions

Before describing the properties of a partitioned region, it is necessary to define the following term.

Definition 6 Search between two nodes is said to be *Free Of Backtracking* (FOB) if depth-first search can always find the shortest path between them without backtracking.

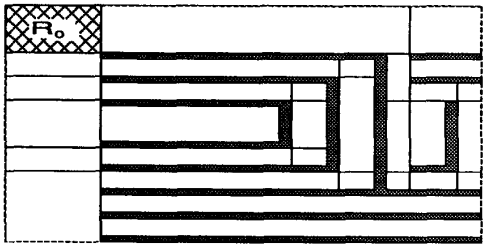
If we know *a priori* the search between certain two nodes to be FOB, search efficiency can be improved greatly. However, it is very difficult, if not impossible, to know this before the actual search takes place. The following lemma provides one useful instance of FOB.

Lemma 1 (Random-Path) For any two cells u and v such that $u >_c v$, the shortest path between u and v is FOB if the search started from v.

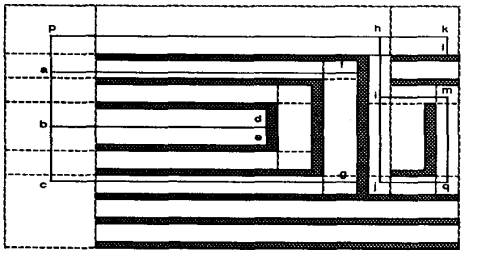
Due to limited space, the proofs of the lemmas and theorems are omitted in this paper.

Corollary 1 For any u and v such that $u \sim_c v$, the shortest path between them is FOB regardless of the search direction used.

According to Corollary 1, the shortest path between two cells in the same region can always be constructed by depth-first search. Furthermore, the shortest path between any two cells with dominance relation can be constructed by depth-first search without backtracking. In Fig. 2a, any cell in R_0 dominates all other cells in the workspace. Consider the construction of a path from a cell $p \in R_0$ to a cell $q \notin R_0$. Fig. 2b shows some of decision points during the search. Without knowing $p \sim_c q$, the search would start from p. The search may proceed towards a or h. If a is chosen, any subsequent search will end up with e or g and then fail. Even if h is chosen, the subsequent search may fail by choosing k instead of i as the next point. To remedy this problem, many algorithms are based on breadth-first search [11, 4] or best-first search [6, 5]. Hence, the computational complexity of these algorithms is $O(n^2)$ for 2D and $O(n^3)$ for 3D, where n is the number of decision points.



(a)



(b)

Figure 2: Dominance relation between the regions and its effect on the search.

By contrast, if the search had started from q, there are still two directions to choose from: one towards j and the other towards m. However, the search proceeding towards m subsequently finds the shortest path between p and q; so does the search proceeding towards j. The absence of backtracking guarantees the success of depth-first search for a shortest paath, thus resulting in computational complexity $O(n)$. With the discretization resolution of 100 $\times$ 100 for 2D (100 $\times$ 100 $\times$ 100 for 3D), use of the dominance relation is shown to improve the search efficiency by a factor of 2 for 2D (4 for 3D) when the time taken to decide among several available directions is not considered. To determine the dominance relation between regions, it is first necessary to understand the shape of each region.

Theorem 1 (2D case) Let v and w be two orthogonal neighbors[5] of a free cell u such that $v \sim_c w$. Then, $u \sim_c v$.

Corollary 2 (3D case) Let v, w, and x be three orthogonal neighbors of a free cell, u, such that $v \sim_c w$ and $v \sim_c x$. Then $u \sim_c v$.

Corollary 3 In 2D space, there exists a rectangle that contains all the connected cells in the same region but no cells from other regions.

Corollary 4 In 3D space, there exists a rectangloid that contains all the cells in the same region and may also contain other embedded rectangloids.

Corollary 3 provides valuable information on the whereabout of

[5] An orthogonal neighbor of a node is the neighbor obtained as a result of applying an orthogonal operator to the node.

the neighboring regions in 2D. It should be noted that neighboring regions of a region are always found alongside its edges. Furthermore, the shortest path between two cells in neighboring regions passes through the projection of one of the two cells to an edge between the two regions. Any other path that does not pass through the projection will have the same or longer length. Note that there are four edges in a rectangle, and thus, there are at most four projections for each region as some of its edges may be occupied by obstacles (see R_5 in Fig. 1).

Unlike the 2D case, Corollary 4 implies a region in 3D to have an arbitrary shape. This is due to the fact that one orthogonal neighbor of a cell may belong to a region different from the one that the other two [6] orthogonal neighbors belong to. Due to this irregular shape of region, it is very difficult to represent a region in 3D. One method of representing a 3D object is to enumerate its vertices, edges, and surfaces. This representation method is not attractive because of the difficulty in determining whether or not a cell belongs to a certain region. Moreover, enumerating all the members associated with a region could be costly due to the existence of a large number of cells in the region. The following lemma provides an important property of such an irregular region.

Lemma 2 Let v_{ijk} and $v_{\ell mn}$ be any two cells such that $v_{ijk} \sim_c v_{\ell mn}$. For any cell v_{opq} such that $min(i,\ell) \leq o \leq max(i,\ell)$, $min(j,m) \leq p \leq max(j,m)$, $min(k,n) \leq q \leq max(k,n)$, $v_{opq} \Re v_{ijk}$ implies $v_{ijk} >_c v_{opq}$.

The above lemma implies that when v_{opq} is not visible from both v_{ijk} and $v_{\ell mn}$, it is completely isolated from the rectangloid formed by v_{ijk} and $v_{\ell mn}$. This is because all other cells within such a rectangloid are dominated by v_{ijk} and $v_{\ell mn}$, and thus, those cells not visible from both v_{ijk} and $v_{\ell mn}$ are not visible from all other cells in the rectangloid either. In other words, any cell v that is visible from both v_{ijk} and v_{opq} is located outside the rectangloid. Therefore, the shortest path berween v_{ijk} and v_{opq} should contain at least one cell outside the rectangloid.

Corollary 5 For the smallest rectangloid containing a given cell u and for all the cells v such that $v \sim_c u$, $u \Re w \Rightarrow u >_c w$ for all cells w inside this rectangloid. Such a rectangloid is called the *Rectangloid of Dominance* (ROD) of u.

Since the shape of region and/or ROD is a rectangloid, it is sufficient to represent the member cells in the region with the two extreme points $(x_{min}, y_{min}, z_{min})$ and $(x_{max}, y_{max}, z_{max})$. Whether a cell belongs to a region or not can easily be checked by comparing its location with these two points of the region or ROD. These two points will henceforth be called the *range* of region R and denoted by $r_{min}(R)$ and $r_{max}(R)$. Using the range, the *cover* relation is defined as follows.

Definition 7 For any two regions R_1 and R_2, R_1 is said to *cover* R_2, denoted by $R_1 \rhd R_2$, when

$$r_{min}(R_1) \leq r_{min}(R_2),\ r_{max}(R_2) \leq r_{max}(R_1), R_1 \neq R_2.$$

3.2 Workspace Representation

Dominance relations among regions can be represented as a graph and so can the workspace.

[6] There are at most three orthogonal neighbors of a cell in 3D

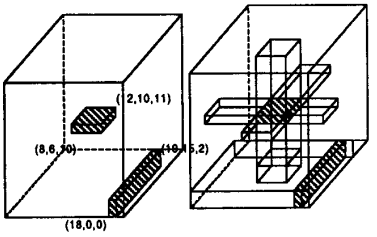


Figure 3: Generation of 3D regions.

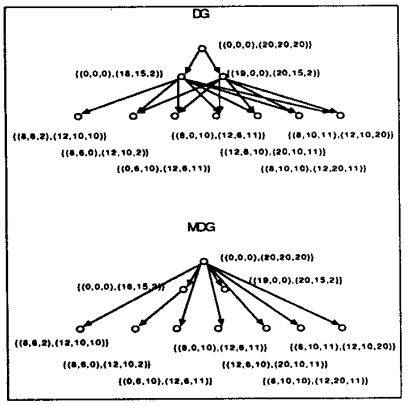


Figure 4: DG and MDG.

Definition 8 The workspace is represented as a digraph, $G = (V, E)$, where V is the set of regions and E is the set of edges such that there exists an edge e from $R_1 \in V$ to $R_2 \in V$ if and only if $R_1 \rhd R_2$.

There are two sources of difficulty to obtain the dominance graph (DG): (i) it is difficult to check the dominance relation between all pairs of regions due to the large number of possible combinations, and (ii) it is difficult to describe a 3D region due to its irregular shape.

To circumvent these difficulties, a modified dominance graph (MDG) is defined as follows.

Definition 9 The MDG is a digraph, $MDG = (V, E')$, where V is the set of regions and E' is the set of edges such that $\exists$ an edge e from $R_1 \in V$ to $R_2 \in V$ if and only if $R_1 \rhd R_2$, $R_1 \neq R_2$, and there is no $R \in V$ such that $R_1 \rhd R$ and $R \rhd R_2$.

Notice that a MDG contains partial information on the dominance relation for a given workspace. Especially, $E' = \emptyset$ for 2D as shown in Corollary 3. A similar example can also be found in Fig. 2. Though R_0 dominates all other regions in the workspace, it will not be shown in the MDG. Though using the MDG will make the computation somewhat inefficient, it will not degrade the quality of the path obtained. Fig. 3 shows an example workspace with two obstacles and the same workspace after partition. Then, the workspace is converted into DG and MDG as shown in Fig. 4. Notice that most of the dominance relations in DG are shown in MDG except those of two region $\{(0,0,0),(18,15,2)\}$ and $\{(19,0,0),(20,15,2)\}$. This is due to the limited range of those two regions.

We have shown that the shortest path can be found with depth-first search when a dominance relation exists between the origin and destination. In some cases, however, no dominance relation may exist between a given pair of origin and destination. Consider the problem of finding a shortest path between two cells u and v such that $u \not\succ_c v$ and $v \not\succ_c u$. Let $R(u)$ be a region containing the cell u. The shortest path between u and v should contain at least a cell from one of the regions next to $R(u)$. Such regions will henceforth be called *bordering regions* of $R(u)$. The closest cell that belongs to a bordering region of $R(u)$ can be found by projecting u to its borders. There exist at most 4 projections in 2D and 6 projections in 3D because the shape of region (and ROD) is rectangular. Suppose the shortest path $P(u, v)$ between u and v passes through $R(w)$, one of $R(u)$'s bordering regions where w is the projection of u. Then one of the following cases is true.

1. w is visible from u.
2. w is not visible from u.
3. w is occupied by an obstacle.

In Case 1, $P(u, v)$ can be obtained by concatenating $P(u, w)$ and $P(w, v)$ in L_1-metric. Note that $P(u, w)$ is a straight-line segment between u and w; otherwise, w cannot be visible from u. Case 2 cannot be true since no cell in $R(w)$ can be visible from u and $R(w)$ is not next to $R(u)$.

Fig. 5 shows an example of Case 3. Among 4 projections a, b, c, and d of a cell u, b and c are occupied by an obstacle. Unlike b, we may have to find a replacement cell $x \in R(c)$ for c that is closest to u but not inside the obstacle. Finding such a cell may be difficult as, in many cases, such a cell is not unique in 3D. It should be noted that we need a replacement for c only when shortest path should pass through the border of $R(u)$ and $R(c)$. For example, it is not necessary to find a replacement for c when the destination is e as the shortest path can pass d. On the contrary, the shortest path between u and f should pass through the common border between $R(u)$ and $R(c)$. That is the case when $R(u)$, $R(c)$, and $R(f)$ are separated by obstacles that are located completely outside $R(u) \cup R(c) \cup R(f)$. Such obstacles do not intefere with the path between u and f, and can thus be ignored. In other words, construction of $P(u, f)$ is FOB when starting from u.

The following algorithm constructs a shortest path between u and v for the general case. Informally, after initialization, the algorithm examines the MDG to see whether there exists any dominance relation between the current cell (initially, the origin) and the destination. If there is, the algorithm constructs the path using depth-first search. Otherwise, a set T of projections of the current cell is obtained. For each member of T, check whether it is occupied by obstacle or not. If it is occupied, check whether construction of a path between the destination and the current cell is FOB or not. If so, the algorithm stops after constructing the path. Otherwise, that projection is deleted from T, the remaining members of T are added to S, the set of examined cells. Then, we choose the most attractive cell (the closest cell to the destination) in S as the current cell and the procedure repeats itself until path construction is completed or S becomes empty. What we said above can be summarized in algorithm form as follows.

1. Let $Best := u$, $P(u, Best) = \text{nil}$, $S := \emptyset$, $T := \emptyset$ and $U := \emptyset$.
2. If $Best >_c v$ then construct $P(Best, v)$ and go to Step 8.
3. If $v >_c Best$ then construct $P(Best, v)$ and go to Step 8.

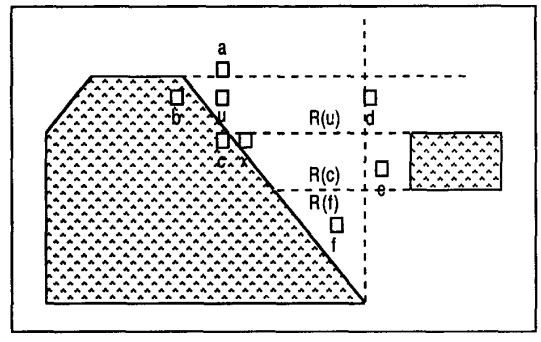


Figure 5: Regions separated by an obstacle outside their RODs.

4. $T := \{$ Projections of $Best \} - S$. For every $w \in T$,
   If w is occupied by an obstacle then try to construct $P(Best, v)$
   using depth-first search.
   If path construction is successful then go to Step 8.
   Else $T := T - \{w\}$.
   else $P(u, w) := \text{concat}(P(u, |current\_cell), P(Best, w))$.
5. $S := S \cup T$, $S := S - \{Best\}$ and $U := U \cup \{Best\}$.
6. Let $Best$ be such that $\min_{Best \in S}(\text{length}(P(u, Best)) + d_1(Best, v))$.
7. Go to Step 2.
8. $P(u, v) := \text{concat}(P(u, Best), P(Best, v))$.

The computational complexity from Step 2 to Step 4 is $O(n)$ where n is the resolution of the workspace, i.e., the number of cells in each axis. As the algorithm stops when either path is found or S is empty, the maximum number of iterations from Step 2 to Step 5 occurs when S is empty. That is, the maximum number of iterations is identical to the total number of regions, m, and the overall complexity becomes $O(mn)$. Since the total number of regions can be as high as the total number of cells (i.e., $m = O(n^3)$), the overall complexity can be as high as $O(n^4)$. This overall complexity is deceiving as the total number of regions is much smaller than the total number of cells (i.e., $m << n^3$). Since RODs, rather than individual regions, are searched, search efficiency is also improved.

4 AN EXAMPLE WORKSPACE

In this section, we consider an example workspace cluttered with obstacles with various shapes as shown in Fig. 6. Specifically, the effects of various orientations of an obstacle on workspace partitioning are described and a typical path in such an environment is also constructed.

The workspace is digitized as a $32 \times 32 \times 32$ grid. The resulting MDG has statistics as shown in Table 1. Region sizes vary from one to several thousand cells. Most one-cell regions are due to the pyramid shape obstacle K_2. Diagonal edges in K_2 usually divide the workspace into small regions. Due to these one-cell regions, the median region size is 2 while the mean region size is 71.6.

According to our simulation based on 1,000 randomly selected origin-destination pairs, the average number of regions searched is less than three. For 63.6% of the cases considered, there exists a dominance relation between the region containing the origin and that

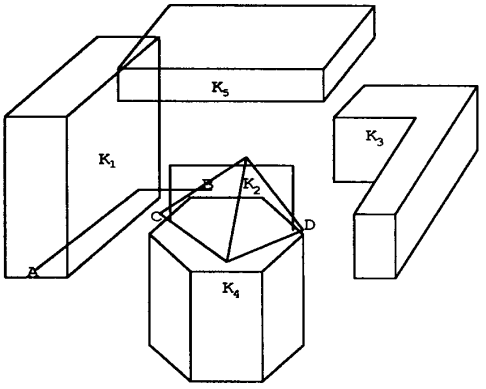


Figure 6: A sample workspace with various obstacles.

containing the destination, thus requiring no region to be searched at all. This is mainly due to the fact that the largest region dominates all the other regions and requires no regions to be examined, and on the average, either the origin or the destination lies in the largest region for approximately 50% of the time. Furthermore, only one region needs to be searched for 28.2% of the time. This implies that we need to search less than two regions for 91.8% of the time.

The fragmentation has little effect on path construction, i.e., the probability of the destination or origin falling in a one–cell region is less than 0.01. Even in the case when the origin falls in a one-cell region, the number of regions to be searched is very small if the destination belongs to a larger region by changing the search direction. For the source-destination pair (A, B) in Fig. 6, the total number of regions examined is zero since there exists a dominance relation between A and B, even though B falls in a one–cell region. The worst-case occurs when both the origin and destination fall in one–cell regions and are placed on the opposite side of K_2, shown as C and D in Fig. 6, respectively. In such a case, the total number of regions searched can be as high as 200. However, the probability of such a case occurring is less than 0.0001.

5 CONCLUSION

In this paper, we presented a new method of partitioning the workspace using L_1-visibility. It was shown that the optimal path w.r.t. L_1-metric between two partitioned regions can be obtained easily if a dominance relation exists between them. When no such relation exists between the origin and destination, we have presented an $O(mn)$ algorithm. Our path planner is shown to find an optimal solution for the digitized workspace. Though the workspace with polyhedral obstacles are regarded as a more general solution, many workspace configurations are obtained in digitized form and our algorithm provides a very efficient solution in such an environment.

This paper mainly addressed the SPPP in 3D. Unlike other approaches, our method does not depend on any particular geometry. Since each region is represented with two extreme points or inequality predicates, our algorithm can be extended to $k \geq 4$ D space without much difficulty.

Total number of cells	32,768
Total number of cells occupied by obstacles	3,550
Total number of free cells	29,218
Total number of regions	408
Size of the largest region	8,857
Size of the smallest region	1
Median region size	2
Mean region size	71.6
Avg. number of regions examined	2.23

Table 1: Statistics of regions.

References

[1] R. A. Brooks and T. Lozano-Perez, "A subdivision algorithm in c-space for findpath with rotation," *IEEE Trans. System, Man, and Cybernetics*, vol. 15, pp. 224–233, Mar. 1985.

[2] L. P. Chew, "Planning the shortest path for a disc in $o(n^2 \log n)$ time," *ACM Proc. Symp. on Computational Geometry*, pp. 214–220, Jun. 1987.

[3] J. Davis and S. Tufekci, "A decomposition algorithm for locating a shortest path between two nodes in a network," *Networks*, vol. 12, pp. 161–172, 1982.

[4] P. DeRezende, D. Lee, and Y. Wu, "Rectilinear shortest paths with rectangular barriers," *ACM Proc. Symp. Computational Geometry*, pp. 204–213, 1985.

[5] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Math.*, vol. 1, pp. 269–271, 1959.

[6] S. Kambhampati and L. S. Davis, "Multiresolution path planning for mobile robots," *IEEE Journal of Robotics and Automation*, vol. RA-2, pp. 135–145, Sep. 1986.

[7] T. Lozano-Perez, "Automatic planning of manipulator transfer movements," *IEEE Trans. System, Man, and Cybernetics*, vol. 11-10, pp. 681–698, 1981.

[8] T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Trans. Comp.*, vol. 32-2, pp. 108–120, Feb. 1983.

[9] J. Luh and C. Lin, "Approximate joint trajectories for control of industrial robots along cartesian paths," *IEEE Trans. System, Man, and Cybernetics*, vol. 14-3, pp. 444–450, 1984.

[10] A. Mirzaian, "Channel routing in VLSI," *ACM Symp. Theory of Computing*, pp. 101–107, 1984.

[11] C. O'Dunlaing and C. K. Yap, "The voronoi diagram method of motion-planning: I. the case of a disc.," *J. Algorithm*, vol. 6, pp. 104–111, 1985.

[12] S. H. Suh and K. G. Shin, "Robot path planning with a weighed distance-safety," *Proc. 26-th Conf. on Decision and Control*, pp. 634–641, 1987.

[13] C. Thorpe, "Path relaxation: Path planning for a mobile robot," *Proc. of the National Conference on Artificial Intelligence*, Aug. 1984.