Shortest Path Planning in Discretized Workspaces Using Dominance Relation

Sungtaeg Jun and Kang G. Shin, *Senior Member, IEEE*

***Abstract*—Various forms of shortest path planning (SPP) have been studied by numerous researchers. 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 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 rectilinear visibility in 3D or higher space. Unlike the case of 2D space where the shape of a partition is a rectangle, the shape of a partition in 3D or higher space is arbitrary. However, we can prove the existence of dominance relations between the partitioned regions. This relation is then utilized to efficiently solve the SPP problem in 3D or higher space.**

***Key Words*—Find path problem, L_1 and L_∞ metrics, path planning, rectilinear visibility, shortest-path planning, visibility graph.**

I. Introduction

One of the main problems in achieving automatic task scheduling is automatic path planning (APP) in the presence of obstacles in the workspace. The applications of APP are quite diverse, the most notable being automatic path generation for mechanical manipulators [10], [12] and/or autonomous vehicles [17]. Another important application is automatic channel routing in VLSI design [13] and computer networks [4]. In particular, the APP of mechanical manipulators, known as robot motion planning, is a challenging task due to the large number of degrees of freedom (DOF's) involved. Another factor to be considered for robot path planning is the various costs associated with a path, such as the length and safety of the path. In this paper, we develop a new method for dealing with these costs in 3D or higher workspaces.

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 [3], [11], clearance from the obstacles [14], and a combination of distance and clearance [16]. In particular, the APP that minimizes the traveling distance is called the *shortest path planning* (SPP) problem while the other APP's are usually called the *find path* (FP) problem.

The main difficulty in solving the SPP problem are the various shapes of moving objects. The specific solution for a specific shape usually does not apply to other shapes. To remedy this problem, the configuration space approach (CSA) was proposed in [11], [2]. 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. As a result of the development of CSA, many issues associated with the FP can be resolved with the solution techniques developed for SPP.

The most popular 2D 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 contains 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 [6]. Though the VG is very useful in 2D, it is very difficult to use for 3D or higher dimensional problems. Others [1], [15], [8] approached this problem by representing the free space in different ways. These algorithms either have exponential complexity or are heuristic.

In this paper, we introduce the L_1 *visibility* between two points in a digitized workspace, based on which the workspace will be partitioned. Previous attempts using L_1 metric either are limited to 2D problems [17] or restrict obstacles to be a certain type [9]. By partitioning the workspace, we only have to search each partitioned region (of cells) instead of dealing with each individual cell. Furthermore, we can derive dominance relations between the partitions of the workspace and utilize these relations to reduce the number of search steps required.

The paper is organized as follows. Section II states the SPP formally. In Section III, we define the concept of L_1 visibility and demonstrate how it partitions the workspace. In Section III-A, the properties of a partitioned workspace are examined. Section III-B presents a graph representation of the workspace based on which an SPP solution algorithm is derived. Section IV presents an example and the simulation

Manuscript received September 1, 1989; revised August 9, 1990. This work was supported by the U.S. Airforce Office of Scientific Research under Contract F33615-85-C-5105 and by the National Science Foundation under Grant DMC-8721492. A portion of this paper was presented at the 1990 IEEE International Conference on Robotics and Automation.

The authors are with the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109-2122.

IEEE Log Number 9144346.

results. The paper ends with concluding remarks in Section V.

II. Problem Statement

When considering 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: 1) an infinite number of paths exist for each given origin–destination pair, and 2) 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.[1] Such a 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 $l \times m \times n$ identical cells. According to the CSA [11], the object to be moved can be shrunk to a point by enlarging all of the obstacles in the workspace. In what follows, cells are represented as $o, p, q, \ldots$ if their locations need not be specified. However, if their locations need to be specified, they are represented as $o_{ijk}, p_{lmn}, q_{abc}, \ldots$. When a sequence of cells need to be specified, these cells are called $v_1, v_2, v_3, \ldots$. Informally, the goal of the 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.

The most commonly used cost is path length. In a Euclidean space E^d, the L_p distance between two points $x = (x_1, x_2, \ldots, x_d)$ and $y = (y_1, y_2, \ldots, y_d)$ is defined as

$$d_p(x, y) = (|x_1 - y_1|^p + |x_2 - y_2|^p + \cdots + |x_d - y_d|^p)^{1/p}, \quad \text{where } 1 \le p < \infty$$

$$d_\infty(x, y) = \max(|x_1 - y_1|, |x_2 - y_2|, \ldots, |x_d - y_d|).$$

Though there exist an infinite number of L_p metrics, only three of them have significance for path planning: L_1, L_2, and L_∞. The advantage of using the L_2 metric over the L_1 or the L_∞ metrics is its ability to describe the object's traversal distance. On the other hand, using the L_1 and L_∞ metrics can improve the safety of the generated path. It should be noted that the travel distance is less meaningful in joint space while safety (i.e., the margin of tracking error) is more important in the joint space, as compared to Cartesian space.

We will limit our discussion to the L_1 metric, since, as shown in [9], any problem in the L_1-metric space can be transformed into an equivalent problem in the L_∞-metric space with a simple transformation of the coordinate system. In this paper, the *cost* of a path P, denoted by $C(P)$, is defined as the length of P measured in the L_1 metric. Furthermore, since all the cells are identical, the L_1 distance between the centers of any two physically adjacent cells are identical and will be treated as *unit distance*. For the purpose of this paper, two cells are said to be *neighbors* if they are physically adjacent.

The path planning problem can now be stated formally as follows: For any two *given* points x and y and a set $\mathcal{O}$ of cells that are occupied by obstacles, we want to *find* a sequence, $P = v_1 v_2 \ldots v_n$, of neighboring cells such that

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

while *minimizing* n.

III. Partitioning the Workspace

In this section, we will define the L_1 visibility between two cells and introduce an equivalence relation that partitions the workspace.

Definition 1: In an 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 $P = v_0 v_1 \ldots v_{d_1(v,w)}$ 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.

Definition 2: For any two adjacent cells, a set $N = \{n_x^+, n_x^-, n_y^+, n_y^-, n_z^+, n_z^-\}$ is called the set of *neighbor operators* if

$$n_x^+(v_{ijk}) = v_{lmn} \Rightarrow l = i + 1, m = j, n = k$$

$$n_x^-(v_{ijk}) = v_{lmn} \Rightarrow l = i - 1, m = j, n = k$$

$$\vdots$$

$$n_z^-(v_{ijk}) = v_{lmn} \Rightarrow l = i, m = j, n = k - 1$$

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[2] 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

$$P = v_0 v_1 \ldots v_{d_1(v,w)}$$

where

$$v_0 = v, \; v_{d_1(v,w)} = w, \; v_i = n(v_{i-1}) \quad \text{for some } n \in O$$

and O is called the *generating operator set* of P. The dual of O, as denoted by O^*, is the generating operator set of the reverse sequence of P, i.e., $v_{d_1(v,w)} v_{d_1(v,w)-1} \cdots v_0$.

In other words, we can say that two cells are visible from each other when there exists a path without running back and forth between the two cells. Using the above definition of visibility, the *dominance* relation between two sets of cells are defined as follows.

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

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

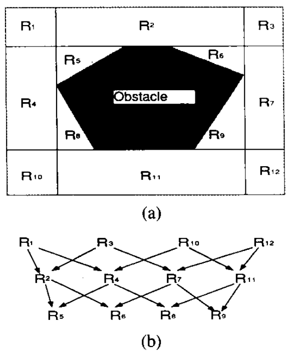


Fig. 1. Partitioning of the workspace into regions and the dominance graph.

Definition 4: For any two cells u and v in a workspace W, u is said to dominate v, as 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, as denoted by $A >_s B$, iff for any $v \in B, \exists u \in A$ such that $u >_c v$.

Based on the dominance relation between cells, the *equality* 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 $v >_c u$.

Notice that the relation $\sim_c$ is an equivalence relation,[3] 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 $S(v)$, the set of all the cells that are not visible from u. Such a set will henceforth be referred to as a *region*. Any two cells that belong to the same region are visible from the same set of cells. Similarly, when a cell is dominating another cell, the dominating cell is visible from all the cells that are visible from the dominated cell. Fig. 1 shows an 2D example of a partitioned workspace. In Fig. 1(a), any cell in R_1 is visible from any other cell in the entire workspace except for those cells in R_9. Similarly, no cell in R_2 is visible from any other cell in $R_8 \cup R_9 \cup R_{11}$. The dominance relation between the regions are shown as a graph in Fig. 1(b).

There are several ways of obtaining regions. In the case of the 2D space, the border of the regions is formed by projecting the edges of the obstacles along the x and the 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 the total numbers of cells visible from two equivalent cells are 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 below.) 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 visible from v (not visible from v). Initially, all the indicator is set to 0. Starting with a cell v of distance 0 from v (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 distances $2, 3, \ldots, K$ from v, where K is the maximum possible distance between any two cells in the workspace. In most cases, K is three times the resolution of each axis. 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.

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

```
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 $w \in D_i(v)$
          $I(w) \leftarrow \max_{u \in D_{i-1}(v) \cap D_1(w)} I(u)$
    end
          $N(v) \leftarrow \sum_{w \in W} I(w)$
  end
end {P1}
```

The output N of **P1** 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 cells next to each other is different, then the two cells belong to different regions. It is possible that two cells have the same number of visible cells even though they belong to two separate regions. To handle this problem, it is necessary to use the following post-processing. First, the output of **P1** is stored into a file and sorted in accordance with the N value and location of each cell. The sorted data will be read in and grouped together to form regions. Upon reading in the sorted data, all the cells with the same number of visible cells form a list. Initially, a region is assumed to contain only the first cell of the list. Then, all the cells physically adjacent to the region are deleted from the list and then added to the region. It is necessary to go through the entire list again until no more cells are added to the region. This procedure can be quite time consuming if only one cell is added during each iteration. That is, the worst case occurs when the last cell of the list is added to the region during each iteration. However, this does not happen as cell locations are also sorted. Similarly, the next region is extracted from the remainder of the list, and the procedure continues until the list becomes empty. Once all the cells with the same number of visible cells are grouped to form regions, the next data are read in and the same procedure applied.

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 (see Fig. 2). Such undetected boundaries can be easily recovered by extending some of the

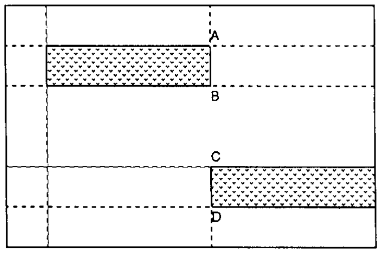


Fig. 2. An example of undetected boundaries.

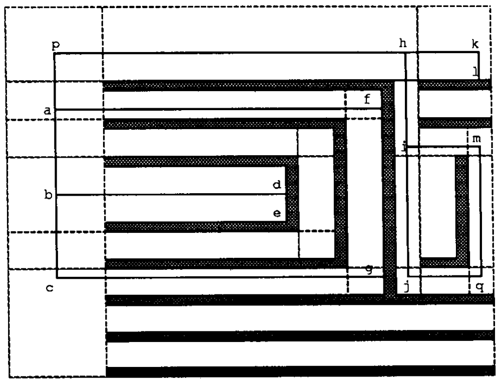


Fig. 3. An instance of an FOB search.

detected boundaries. In Fig. 2, BC can be recovered later by extending either AB or CD.

By partitioning the workspace based on the relation $\sim_c$, the path-planning problem can be divided into the following two subproblems.

Subproblem 1 (Inter-Region): Find a sequence, $P = R_0 R_1 R_2 \ldots R_k R_d$, of regions such that R_0 is the region that contains the origin and R_d the region that contains the destination.

Subproblem 2 (Intra-Region): Find a path to traverse within each region in the sequence found from Subproblem 1.

A. Properties of Partitioned Regions

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

Definition 6: If a depth-first search can always find the shortest path between two nodes without backtracking, it is said to be *free of backtracking* (FOB).

It should be noted that a search can be either FOB or independent of the search direction. Fig. 3 shows one instance of FOB search. Let us consider some of the decision points during the search of path between p and q. Without any *a priori* knowledge, the search would start from p. The search may proceed toward 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 the breadth-first search [14], [5] or a best-first search [7], [6]. 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.

If the search started from q, there are still two directions to choose from: one toward j and the other toward m. However, in the case, both the subsequent search toward m and that toward j find the shortest path between p and q because the absence of backtracking guarantees the success of the depth-first search for a shortest path, thus resulting in the computational complexity $O(n)$. With a discretization resolution of 100×100 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.

If the search between certain two nodes is known *a priori* to be FOB, search efficiency can be improved greatly. In many cases, 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 an FOB search.

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.

Proof: Let $D_k(v)$ be a set such that $D_k(v) = \{u \mid d_1(u, v) = k\}$, and w be a cell such that $w \in D_1(v) \cap D_{d_1(u,v)-1}(u)$. If no such w exists, v is not visible from u. However, this is impossible because v is always visible from itself, and thus, v should be visible from u by the definition of dominance. Thus, there always exists at least one cell, say w_1, such that $w_1 \in D_1(v) \cap D_{d_1(u,v)-1}(v)$. Since w_1 is visible from v, it is also visible from u by the definition of dominance. That is, there always exists a cell w_2 such that

$$w_2 \in D_2(v) \cap D_1(w_1) \cap D_{d_1(u,v)-2}(u).$$

Similarly, for any w_{i-1}, there always exists a $w_i \in D_1(w_{i-1})$ such that $w_i \in D_i(v) \cap D_{d_1(u,v)-i}(v)$ for $i = 2, 3, \ldots, d_i(u, v)$. ■

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 subproblem *Intra-Region* can be solved trivially, i.e., the shortest path between two cells in the same region can be constructed by a depth-first search. Furthermore, the shortest path between any two cells with a dominance relation can be constructed by a depth-first search without backtracking.

We now want to show how to determine the dominance relation between regions and how to construct a path between regions with a dominance relation. 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[4] of a free cell u such that $v \sim_c w$. Then $u \sim_c v$.

Proof: Since $v, w \in D_1(u)$, there are four possible cases to consider: $v = n_x^+(u)$, $w = n_y^+(u)$; $v = n_x^+(u)$, $w =$

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

$n_y^-(u)$; $v = n_x^-(u)$, $w = n_y^+(u)$; and $v = n_x^-(u)$, $w = n_y^-(u)$. Since one can prove the theorem in a similar way for all of these four cases, it is sufficient to consider only one of them; we have chosen the first case. That is, we want to show both $v >_c u$ and $u >_c v$ when $v = n_x^+(u)$ and $w = n_y^+(u)$.

Consider the case of $v >_c u$ first. Let r be any cell visible from u; then there exists at least a sequence $uu_1u_2 \ldots u_{d_1(u,r)-1}r$, of free cells by the definition of visibility. Since $u_1 \in D_1(u)$, u_1 should be the neighbor of u in the positive/negative direction of the x axis or the y axis.

- When $u_1 = n_x^+(u)$, r is visible from v because $u_1 = v$.
- When $u_1 = n_y^+(u)$, r is visible from w because $u_1 = w$.
- When $u_1 = n_x^-(u)$, r is visible from v because there exists a sequence $vuu_1u_2 \ldots u_{d_1(u,r)-1}r$.
- When $u_1 = n_y^-(u)$, r is visible from w because there exists a sequence $wuu_1u_2 \ldots u_{d_1(u,r)-1}r$.

Therefore, any cell visible from u is also visible from either v or w. Thus, $v >_c u$, i.e., $v, w >_c u$.

Now we want to show $u >_c v$. Let p be any cell visible from both v and w. Then there exists at least a sequence $P_v = vv_1v_2 \ldots v_{d_1(v,p)-1}p$ of free cells. By the definition of visibility, there should be four possible generating operator sets O of P_v.

- When $O = \{n_x^+, n_y^+\}$, $uvv_1v_2 \ldots v_{d_1(v,p)-1}p$ will be the shortest path from u to p because $v = n_x^+(u)$. Hence, p is also visible from u.
- When $O = \{n_x^+, n_y^-\}$, $uvv_1v_2 \ldots v_{d_1(v,p)-1}p$ will be the shortest path from u to p because $v = n_x^+(u)$. Thus, p is also visible from u.
- When $O = \{n_x^-, n_y^+\}$, there exists at least one sequence $ww_1w_2 \ldots w_{d_1(w,p)-1}p$ of free cells between w and p because p is visible from w. Then, p can be reached from u via the sequence $uww_1w_2 \ldots w_{d_1(w,p)-1}p$ because $w = n_y^+(u)$.
- When $O = \{n_x^-, n_y^-\}$, if P_v consists of the cells obtained using the operator n_x^-, then $v_1 = u$ (because there is a unique $n_x^-(v) = u$). Else let v_k be the first cell such that $n_x^-(v_{k-1})$. Since v_{k-1} is visible from w, there exists a sequence $P_w = ww_1w_2 \ldots w_kv_k$ of free cells that can be obtained using the operator n_y^-. Since $n_y^-(w) = w_1 = u$, and thus $uw_2 \ldots w_kv_kv_{k+1} \ldots v_{d_1(v,p)-1}p$ is a valid sequence for visibility, i.e., p is visible from u.

Therefore, $u >_c v$. ■

Theorem 1 states the important fact that the shape of a region in 2D is always rectangular.

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 $u \sim_c v$. 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.

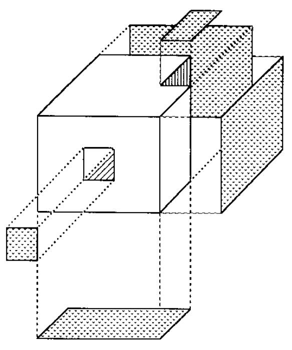


Fig. 4. Typical shape of a region in 3D.

Corollary 3 provides valuable information on the whereabouts of 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 a 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 in the 2D case, Corollary 4 implies a region with an arbitrary shape (see Fig. 4) in the 3D case. 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[5] orthogonal neighbors belong to. Due to the irregular shape of the 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. However, 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_{lmn} be any two cells such that $v_{ijk} \sim_c v_{lmn}$. For any cell v_{opq} such that $\min(i,l) \le o \le \max(i,l)$, $\min(j,m) \le p \le \max(j,m)$, $\min(k,n) \le q \le \max(k,n)$, $v_{opq} \Re v_{ijk}$ implies $v_{ijk} >_c v_{opq}$.

Proof: For any cell u such that $u \Re v_{opq}$, there exists a path $P_1 = uw_1w_2 \ldots w_{d_1(u,v_{opq})-1}v_{opq}$ generated by a set of orthogonal neighbor operators. Since $(\forall u\ u \Re v_{opq} \Rightarrow u \Re v_{ijk}) \Rightarrow v_{ijk} >_c v_{opq}$, let us suppose $u \Re v_{ijk}$. Let w_i be the cell such that $w_i \Re v_{ijk}$, and let n be a neighbor operator such that $n(w_{i+1}) = w_i$ for some cell w_{i+1}. Since $v_{opq} \Re v_{ijk}$, there exists a generating set O_1 of operators for a shortest path P_1 from v_{opq} to v_{ijk}. Then $\{n\} \cup O_1$ is not a set of orthogonal neighbors because $u \Re v_{ijk}$. Since $v_{ijk} \sim_c v_{lmn}$ and $v_{opq} \Re v_{ijk}$, there exists a generating set O_2 for a shortest path P_2 from v_{opq} to v_{lmn}. Then, $\{n\} \cup O_2$ should be an orthogonal set of neighbor operators because O_1 is the

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

dual of O_2. This implies that $w_i \Re v_{lmn}$, which contradicts the fact that $v_{ijk} \sim_c v_{lmn}$. Thus, $u \Re v_{ijk}$ and $v_{ijk} >_c v_{opq}$. ■

The above lemma implies that when v_{opq} is not visible from v_{ijk} and v_{lmn}, it is completely isolated from the rectangloid formed by v_{ijk} and v_{lmn}. This is because all other cells within such a rectangloid are dominated by v_{ijk} and v_{lmn}, and thus, the cells which are not visible from v_{ijk} and v_{lmn} are also not visible from all other cells in the rectangloid. 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 between 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 the 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 then easily be checked by comparing its location with these two points or with ROD. These two points will henceforth be called the *range* of the region R and can be denoted by $r_{\min}(R)$ and $r_{\max}(R)$. Using this 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

$$r_{\min}(R_1) \le r_{\min}(R_2), \quad r_{\max}(R_2) \le r_{\max}(R_1), \quad R_1 \ne R_2.$$

B. Workspace Representation

A workspace can be represented as a graph showing dominance relations among its regions.

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 $\exists$ 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): 1) it is difficult to check the dominance relation between all pairs of regions due to the large number of possible combinations, and 2) 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 $R_1 \rhd R_2$, $R_1 \ne R_2$ and if and only if there is no $R \in V$ such that $R_1 \rhd R$ and $R \rhd R_2$.

Notice that an 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. 3. Though R_0 dominates all other regions in the workspace, it will not be shown in the MDG. That is, many of the dominance relations may be lost in the MDG. Using the MDG will make the computation somewhat

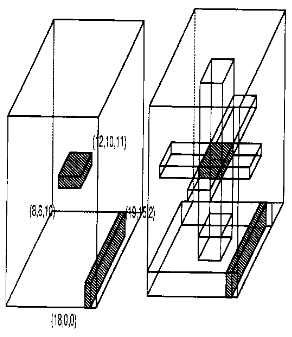


Fig. 5. Generation of 3D regions.

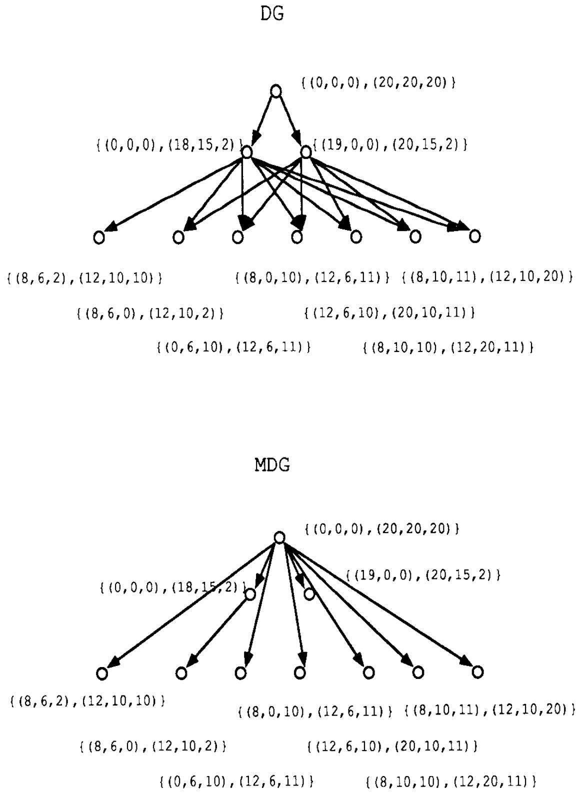


Fig. 6. DG and MDG.

inefficient as compared to DG (due to lack of full knowledge of dominance relations), but will not degrade the quality (length) of the path obtained. Fig. 5 shows an example workspace with two obstacles and its partitioned version. The partitioned workspace is converted into a DG and an MDG as shown in Fig. 6. Notice that most of the dominance relations in the DG are also given by the MDG except for those of the two regions $\{(0, 0, 0), (18, 15, 2)\}$ and $\{(19, 0, 0), (20, 15, 2)\}$. This is due to the limited range of the two regions.

We have shown that a shortest path can be found via a depth-first search when a dominance relation exists between a

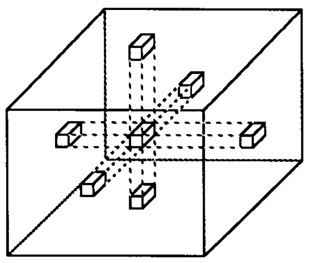


Fig. 7. Projections of a search point.

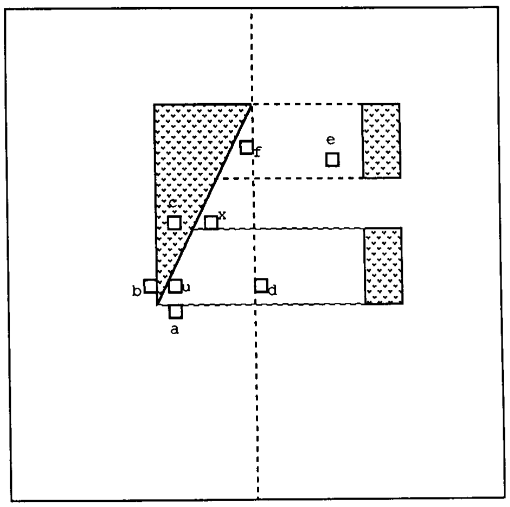


Fig. 8. Regions separated by an obstacle outside their ROD's.

given origin and destination pair. In some cases, however, no dominance relation may exist between the origin and the destination. Consider the problem of finding a shortest path between two cells u and v such that $u \not>_c v$ and $v \not>_c u$. 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 four projections in 2D and six projections in 3D because the shape of a region (and an ROD) is rectangular (see Fig. 7). 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 the 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. 8 shows an example of Case 3. Among four projections a, b, c, and d of a cell u, b and c are occupied by an obstacle. Unlike with $R(b)$, $R(u)$ shares a border with $R(c)$ and the shortest path may have to pass through that particular border. In such a case, 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 the 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 through d. On the other hand, the shortest path between u and f should pass through the common border of $R(u)$ and $R(c)$. This 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 interfere with the path between u and f and can thus be ignored. In other words, the 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 are obtained. For each member of T, check whether it is occupied by an 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 cells yet to be examined, and the best call is added to U, 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 $Best := u$, $P(u, Best) =$ 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.
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 $P(u, w) :=$ 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) $P(u, v) :=$ concat$(P(u, Best), P(Best, v))$.
```

The computational complexity from Step 2 to Step 4 is

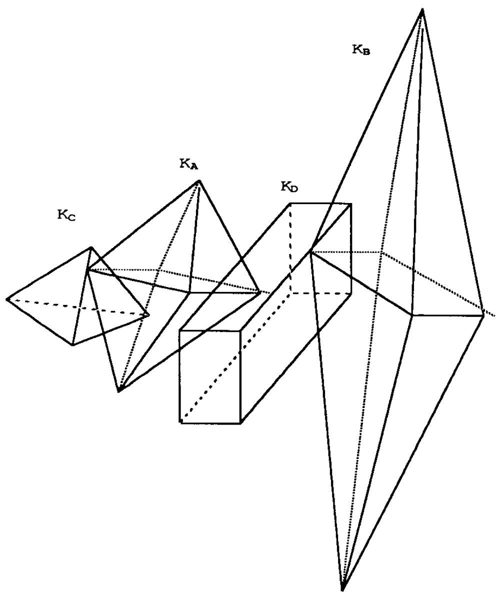


Fig. 9. Various shapes of obstacles.

$O(n)$ 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., $m \ll n^3$). Since ROD's, rather than individual regions, are searched, search efficiency is also improved.

IV. Effects of Obstacle Shapes

In this section, we will discuss various shapes of obstacles and their effect on the fragmentation of the workspace and present simulation results. In general, a rectilinear surface of an obstacle partitions a workspace into large regions while a diagonal surface of the obstacle generates many small regions, i.e., fragmentation. In particular, the worst-case fragmentation occurs when there are several obstacle surfaces such that $Ax + By + Cz + D = 0$ for $|A| = |B| = |C| = 1$. An example obstacle of this type can be seen in Fig. 9, in which there are four different shapes of obstacles: 1) diamond shape, K_A and K_B; 2) pyramid shape, K_C; and 3) rectilinear shape, K_D.

Upon placing these obstacles in the middle of a workspace discretized into a $20 \times 20 \times 20$ grid, we have obtained the results in Table I. The obstacle K_A causes the most fragmentation as there is no rectilinear edge, and slopes of the surfaces are arranged to be very close to 1. K_B has a shape similar to K_A except that both the top and bottom points are stretched by six times. The proposed partitioning, performed on a Sun 3/280,[6] supports the intuition that the obstacle containing K_A exhibits the worst fragmentation. Though the number of regions generally increases with the number of obstacles, this number actually decreased with the existence of rectilinear obstacle K_D. The difference between K_A and K_B can be explained by the fact that digitizing a workspace has the same effect as approximating it with rectangular walls. Though K_B is much larger than K_A, many of K_B's surfaces are close to being rectilinear surfaces as the slope of the original surface has much larger and/or smaller than 45° with respect to the reference axis.

TABLE I
Statistics of Workspace with Various Shapes of Obstacles

Obstacle	K_A	K_B	$K_A + K_C$	$K_A + K_D$
Total Number of Regions	331	218	523	157
Size of the Largest Region	1657	780	1657	1141
Size of the Smallest Region	1	1	1	1
Maximum Level of Dominance	2	3	3	3
Number of 1-Cell Regions	96	52	247	27
Average Number Region Searched	3.3	2.7	4.2	2.4

Using the partitioned workspace data, paths between 1000 randomly selected origin-destination pairs have been constructed. All cells are assumed to have an identical probability to become a destination or an origin. On the average, the number of regions searched is approximately 1% of the total number of regions. The fragmentation has little effect on the search, since most of the small regions rarely becomes a destination and/or an origin. During the search, most small regions are bypassed as we search large regions first when there are several regions with the same estimated distance to the destination. The average search time is almost negligible (less than 1 s), once the workspace is partitioned. This number would increase sharply if all regions are assumed to have the same probability of becoming an origin or destination. However, it is impractical to assume that two regions consisting of one cell and 1000 cells, respectively, have the same probability of becoming an origin/destination.

V. Conclusion

In this paper, we presented a new method of partitioning the workspace using L_1 visibility. It was shown that the optimal path with respect to 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. Though the number of regions can be fairly high with diagonal surfaces, it was shown that only 1% of the regions are searched before constructing a path. Using MDG in place of DG has little effect on the search as there is not enough room for improvement in search efficiency. It should be noted that using MDG will not degrade the quality of path, i.e., the length of a generated path. In fact, partitioning a workspace may be the source of efficiency as MDG only contains partial information on dominance relations. Though the workspace with polyhedral obstacles are regarded as a

[6]The Sun workstation is running under SunOS Release 3.5. Each example has taken approximately 3000–4000 s of user time (150–170 s of system time) for the partitioning algorithm.

more general solution, many workspace configurations are obtained in digitized form, and our algorithm provides a very efficient solution in such an environment.

One area needing improvement is the algorithm to obtain the region. The current algorithm is somewhat inefficient with $O(n^6)$ complexity. However, in many applications, the workspace remains static and one needs to calculate the regions only once.

This paper focused on the SPP in 3D. Unlike other approaches, our method does not depend on any particular geometry of the workspace and obstacles. 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.

References

[1] R. A. Brooks, "Planning collision-free motions for pick-and-place operations," *Int. J. Robotics Res.*, vol. 2, pp. 19-44, 1983.

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

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

[4] 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.

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

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

[7] S. Kambhampati and L. S. Davis, "Multiresolution path planning for mobile robots," *IEEE J. Robotics Automat.*, vol. RA-2, pp. 135-145, Sept. 1986.

[8] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE Int. Conf. Robotics Automat.*, Mar. 1985, pp. 500-505.

[9] D. T. Lee and C. K. Wong, "Voronoi diagrams in L_1-(L_∞-) metrics with 2-dimensional storage applications," *SIAM J. Comput.*, vol. 9, pp. 200-211, 1980.

[10] T. Lozano-Perez, "Automatic planning of manipulator transfer movements," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-11, pp. 681-698, 1981.

[11] ——, "Spatial planning: A configuration space approach," *IEEE Trans. Comput.*, vol. 32-2, pp. 108-120, Feb. 1983.

[12] J. Luh and C. Lin, "Approximate joint trajectories for control of industrial robots along cartesian paths," *IEEE Trans. Syst., Man Cybern.*, vol. SMC-14, pp. 444-450, 1984.

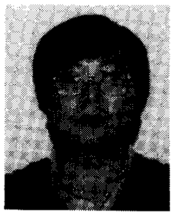
[13] A. Mirzaian, "Channel routing in VLSI," in *Proc. ACM Symp. Theory Comput.*, 1984, pp. 101-107.

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

[15] B. Schachter, "Decomposition of polygons into convex sets," *IEEE Trans. Comput.*, vol. C-27, no. 11, pp. 1078-1082, Nov. 1978.

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

[17] C. Thorpe, "Path relaxation: Path planning for a mobile robot," in *Proc. Nat. Conf. Artificial Intell.*, Aug. 1984.



Sungtaeg Jun received the B.E. degree in 1977 from Seoul National University, Seoul, Korea, and the M.S. degree in 1983 from the University of Detroit, Detroit, MI. He is currently working toward the Ph.D. degree in the Department of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor. His current research interests include CAD/CAM, computational geometry, and computer graphics.

Kang G. Shin (S'75-M'78-SM'83) received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1970 and the M.S. and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, NY, in 1976 and 1978, respectively.

From 1978 to 1982, he was on the faculty of the Rensselaer Polytechnic Institute, Troy, NY. He joined The University of Michigan, Ann Abor, in 1982, where he is currently a Professor of Electrical Engineering and Computer Science and chairs the Computer Science and Engineering Division. He has held visiting positions at the U.S. Airforce Flight Dynamics Laboratory; AT&T Bell Laboratories; the Computer Science Division within the Department of Electrical Engineering and Computer Science at the University of California, Berkeley; and the International Computer Science Institute, Berkeley. In 1985, he founded the Real-Time Computing Laboratory, where he and his colleagues are currently building a 19-node hexagonal mesh multicomputer, called HARTS, to validate various architectures and analytic results in the area of distributed real-time computing. He has authored/coauthored over 180 technical papers (more than 70 in archival journals) in the areas of fault-tolerant computing, distributed real-time computing, computer architecture, and robotics and automation.

Dr. Shin received the Outstanding IEEE Transactions on Automatic Control Paper Award for a paper on robot trajectory planning in 1987. In 1989, he received the Research Excellence Award from The University of Michigan. He served as the Program Chairman of the 1986 IEEE Real-Time Systems Symposium (RTSS), the General Chairman of the 1987 RTSS and the Guest Editor of the IEEE Transactions on Computers August 1987 special issue on real-time systems. He currently chairs the IEEE Technical Committee on Real-Time Systems, is a Distinguished Visitor of the IEEE Computer Society, and is an Area Editor of the *International Journal of Time-Critical Computing Systems*.