Translational Motion Planning for a Convex Polyhedron in a 3D Polyhedral World Using an Efficient and New Roadmap

A.Dattasharma*and S.S. Keerthi†
Department of Computer Science and Automation
Indian Institute of Science, Bangalore 560012, India

Abstract

In this paper, we consider the problem of moving a convex polyhedral object among convex polyhedral obstacles which have pairwise disjoint interiors in three dimensional Euclidean space. We use an augmented Voronoi diagram, which is complete when the free space is bounded, and prove that the size of this diagram is is $O(n^2)$ when the size of the moving object and the number of obstacles are assumed to be constant and n is the total number of faces on the obstacles. We also give an efficient and easy to implement algorithm to construct the roadmap.

1. Introduction. Collision Avoidance Path Planning (CAPP) is a widely studied problem in Robotics. In a general CAPP problem, a set of rigid fixed objects in R^d alongwith their orientations, positions and geometries is supplied, and the objective is to find a feasible path for another rigid body M (called the moving object or robot) from a starting point s to a final point f, or report that no such path exists. A particular class of methods suggested to tackle the problem is called the Roadmap methods [1,2,3,4,5,6]. Several of these approaches tackle only simplified problems in lower dimensions [1,4]. Some methods are for general dimensions and objects but are very difficult to implement [3]. Other methods proposed suffer from difficulties like incompleteness [2], weak deformation retract [5] or absence of an algorithm for construction of the roadmap [6]. Thus the area of roadmaps in three or higher dimensions remains relatively unexplored.

In this paper, we will discuss an algorithm for translational CAPP problem in three dimensional Euclidean space for a convex, polyhedral moving object and convex, polyhedral obstacles with nonempty interiors. Also, the obstacles are assumed to be pairwise interior disjoint and pairwise vertex disjoint. The purely translational motion planning problem is important in assembly sequence planning and when contact motions are important [7]. We construct our roadmap in two stages. First, we define a *generalized Voronoi diagram* which we call the *skeleton*. This skeleton may not be complete. In the second stage, we show that it is possible to make this skeleton complete by adding a small number of artificial edges. In the process, we attempt to answer an open question regarding the size of Voronoi diagrams in 3D for nonpoint convex polyhedra [8]. We prove that, the size of such a diagram (under the distance measure we use) is $O(n^2Q^2l^2)$ where n is the number of 2-faces on the obstacles, Q is the number of obstacles and l is the size of the moving object. Thus when Q and l are assumed to be constant the data complexity is $O(n^2)$. We prove that our roadmap is complete, and give an efficient and easy to implement algorithm to construct the roadmap. The complexity of the algorithm is $O((n+Ql)e+Q^3)$ where e is the number of edges in the roadmap. It is of interest to note that since we are working with polyhedral obstacles, a similar algorithm can construct the Voronoi diagram of a point set under the box or sup metric.

2. Preliminaries. The following notations will be used. O_i denotes the i-th obstacle. M denotes the moving object. $\mathcal{F}$ is the set $R^3 \setminus \bigcup O_i$. We call $\mathcal{F}$ the free space. $\tilde{\mathcal{F}}$ is the set of all points in R^3 where M can rest with respect to a reference point v_{ref} belonging to the interior of M without intersecting any O_i. We call $\tilde{\mathcal{F}}$ the feasible free space. For a set A, $bd(A)$ denotes the boundary of A, $int(A)$ denotes the interior of A and $relint(A)$ the relative interior of A. $\overline{xy}$ will denote the straight line segment joining the two points x and y.

Definition 2.1 A set S is said to be *polyhedral* if S can be written as a finite union of convex polyhedra.

Definition 2.2 Suppose X_i, $i = 1,\ldots,n$ are polyhedral sets. Let E_i be the set of all open 1-faces of X_i and V_i the set of all 0-faces of X_i. Then we call the set $S = \bigcup\{E_i \bigcup V_i\}$ the *skeleton* of $\{X_i\}$.

To construct the generalized Voronoi diagram, we

*e-mail : abhi@csa.iisc.ernet.in

†e-mail : ssk@csa.iisc.ernet.in

use a distance measure taken from [4].

Definition 2.3 Consider M, $v_{ref} \in int(M)$, and $x \in R^3$. Then the *M-distance of A from x* is defined as

$$d(x;A) = \inf\{\lambda : (x + \lambda M) \bigcap A \neq \phi, \lambda \geq 0\}$$

If $x \in A$, $d(x;A) = 0$. For convenience, we write $d(x;O_i)$ as $d_i(x)$.

This distance measure has several advantages over the ordinary l_2 metric. With this distance measure, it is easy to trim off the infeasible parts of the roadmap. Next, it takes into consideration the shape of M, and thus the concept of "most safe" path is meaningful. Also, it ensures that the roadmap has only straight line segments.

Definition 2.4 Let O_i be an obstacle. Then the cell associated with O_i, C_i is the set

$$\{x \in R^3 : d_i(x) \leq d_j(x)\ \forall j \neq i, j \in 1, \ldots, Q\}$$

where Q is the total number of obstacles.It can be shown that each cell is polyhedral.

To avoid unnecessary complications in the algorithm and the proofs, we make two generic assumptions [9]. One is called *independence* [10], which says that every connected set of points where the expanded object maintains exactly k touches is a $4-k$ dimensional manifold for $1 \leq k \leq 4$, and is empty if $k > 5$. The other is that for every $x \in R^3 \setminus O_i$, the set $(x + d_i(x)M) \bigcap O_i$ is a singleton.

3. Basic Properties. We list a few basic results which are used in later sections.

Proposition 3.1 Let A be a convex set. Then the distance function $d(.;A) : R^3 \rightarrow R$ is convex.

Proof Consider $x_1 \in R^3, x_2 \in R^3$. Let $d(x_1;A) = \lambda_1$ and $d(x_2;A) = \lambda_2$. Therefore there exist $m_1 \in M, a_1 \in A$ such that $x_1 + \lambda_1 m_1 = a_1$. Similarly there exist $m_2 \in M, a_2 \in A$ such that $x_2 + \lambda_2 m_2 = a_2$. Now consider any $0 \leq \mu \leq 1$. Then $\mu(x_1+\lambda_1 m_1)+(1-\mu)(x_2+\lambda_2 m_2) = \mu a_1+(1-\mu)a_2$. Since A is convex, the right hand side of the above equation belongs to A. Call it a_3. Expanding the left hand side, we get $(\mu x_1 + (1-\mu)x_2) + (\mu\lambda_1 m_1 + (1-\mu)\lambda_2 m_2) = a_3$. The first term of the left hand side is a point on $\overline{x_1 x_2}$. Call it x_3. Now define $\lambda = \mu\lambda_1 + (1-\mu)\lambda_2$. Then the second term of the left hand side is $\lambda(\frac{\mu\lambda_1}{\lambda} m_1 + \frac{(1-\mu)\lambda_2}{\lambda} m_2)$. But the coefficients of m_1 and m_2 within the parantheses sum to 1, $0 \leq \mu \leq 1$, and $\lambda \geq 0$, thus producing a convex combination. By convexity of M, this is a point in M. Call it m_3. Therefore the above shows that for any $x_3 \in \overline{x_1 x_2}$ there exist $m_3 \in M, a_3 \in A$ such that $x_3 + \lambda m_3 = a_3$. Therefore $d(x_3;A) \leq \lambda = \mu\lambda_1 + (1-\mu)\lambda_2 = \mu d(x_1;A) + (1-\mu)d(x_2;A)$. ∎

Note that by the above proposition, $d(.;A)$ is continuous.[11]

Consider two obstacles O_i and O_j. Let x be a point in R^3 such that $d_i(x) = d_j(x)$, i.e. if v_{ref} is placed on x and M is expanded then it touches both O_i and O_j simultaneously. Consider these touches. Each of these touches can be described by the element of the obstacle (open face, open edge or vertex) being touched and the element of M(open face, open edge or vertex) touching it. Let us define the *type of touch T at x* as these two touch descriptions.

Consider only two obstacles O_i and O_j, and ignore others. Then the following three propositions hold.

Proposition 3.2 Consider a type of touch T. Then

$$C(T) = \text{closure}\{x \in R^3 : d_i(x) = d_j(x) \text{ and type of touch at } x \text{ is } T\}$$

is convex.

Proof Given T, it is easy to see that $C(T)$ can be empty. So suppose $C(T) \neq \phi$. Given T, the set $C(T)$ can be described as a set of linear equalities and inequalities, the equalities arising because of the touch constraints defined by T and the inequalities arising to take into account the faces(edges,vertices) of O_i and O_j not in T. This shows that a nonempty $C(T)$ is convex. ∎

It can be shown that $\bigcup_T C(T)$ consists of polygons.

Proposition 3.3 Consider $C(T)$ as above. Let $x_1 \in C(T)$, $x_2 \in C(T)$, $x_1 \neq x_2$. Then $d_i(x)$ varies affinely over $\overline{x_1 x_2}$.

Proof As T is fixed, $x+d_i(x)M$ touches the same face or edge or vertex of $O_i \forall x \in \overline{x_1 x_2}$. For any x_1, x_2 if we consider $x = \alpha x_1 + (1-\alpha)x_2, \alpha \in [0,1]$ only the equality maintaining the touch corresponding to T is necessary and so $d_i(x) = \alpha d_i(x_1) + (1-\alpha)d_i(x_2)$. ∎

Proposition 3.4 Consider $D = \bigcup_T C(T)$, $C(T)$ as above. Then the number of polygons in D is $O(n_i n_j l^2)$ where n_i, n_j, l are, respectively, the number of closed 2-faces on O_i, O_j and M. Also, the size of vertex set and edge set of D are each $O(n_i n_j l^2)$ and these bounds are optimal.

Proof Between O_i and O_j, we can find $n_i n_j$ combinations of 2-faces. Also, for a touch T to be maintained we need two vertices or edges or faces from M, which can be chosen in $O(l^2)$ ways. Thus the total size of the face set is $O(n_i n_j l^2)$. The example in fig 4.1 shows that it is possible to have $n_i n_j$ combinations of obstacle faces for each object vertex. A similar example can be constructed to show that $O(l^2)$ combinations of object vertices are possible for each pair of obstacle faces. Combining these two figures properly, we get the required bound [12]. Also, by convexity of O_i and O_j, the size of the edge set and vertex set of O_i and O_j are also $O(n_i)$ and $O(n_j)$ respectively. Using these results, it is easy to show that the size of vertex set and edge set of D are also $O(n_i n_j l^2)$. ■

4. Data Size Complexity. Proposition 3.4 addresses the issue of data complexity when only two obstacles are present. However, in a typical case the number of obstacles Q is greater than 2. Using the propositions 3.1–3.4, one can prove the following result.

Theorem 4.1 Suppose there are Q obstacles, and suppose n is defined as $n = \sum_i n_i$. Then the size of each of the vertex set, edge set and the face set of the polygons generated by O_i's is $O(n^2Q^2l^2)$.

Proof For lack of space, we leave out some of the exact details of the proof. However, we give all the major points in an intuitive way.

We will start with the edge complexity, and from there we derive the bounds for the vertex set and the face set.

By independence, an edge e is generated by a three touch. As for every point on the polygons generated by O_i's at least two obstacles have the same M-distance, the three touches must involve at least two obstacles. Also, since there are only three touches, they can come from at most three obstacles. Thus there are only two different cases to consider.

Case 1. 3 touches involving 3 obstacles : Consider the obstacles O_i and O_j. The number of edges generated by these two is $O(n_i n_j l^2)$. Suppose we introduce a new obstacle O_k. Let us call the polygons generated by O_i and O_j as P_{ij}. Then new edges can be generated by intersection of (P_{ij} and P_{ik}) and (P_{ij} and P_{jk}).

Using propositions 3.1 and 3.3 one can establish that an edge of P_{ij} can be broken into at most two edges because of O_k, i.e. an edge e can generate at most one new edge. Also, new edges can be formed by intersection of polygons in P_{ij} and P_{ik}. Note that by independence, a polygon in P_{ij} and a polygon in P_{ik} can't have a common relative interior(because that implies that the set of points where three obstacles are equidistant is a 2D manifold). Thus intersections of polygons of P_{ij} and P_{ik} are transversal. Next observe the following. Suppose two polygons P_1 and P_2 intersect in such a way such that the intersection lies in $relint(P_1)$. Then P_2 is truncated at this intersection and can't intersect any other polygon. Also, the existing part of P_2 can't contribute another edge as affine variation of M-distance in $relint(P_2)$ implies such a case is impossible. Thus if a polygon P_1 intersects r other polygons where $r > 2$ then at least $r - 2$ of these intersections lie in $relint(P_1)$ and are truncated. Then it is easy to see that the total number of extra edges that can be created by polygon intersections of P_{ij} and P_{ik} is at most $O(n_i n_j l^2 + n_i n_k l^2)$. Thus we have, for $P_{ij} \bigcap P_{ik}$, total number of edges as $O(n_i n_j l^2 + n_i n_k l^2)$. Similar argumnets hold for P_{ik}.

Considering $P_{ij} \bigcap P_{jk}$, we find, by similar reasoning, at most $O(n_i n_j l^2 + n_j n_k l^2)$ edges. Together they imply at most $O((n_i n_j + n_j n_k + n_i n_k)l^2)$ edges.

Considring all possible triples (i, j, k), the size of 3 obstacle edge set is $O(n^2Ql^2)$.

Case 2. 3 touches involving 2 obstacles : Using a reasoning similar to case 1 we can show that the 2 obstacle edge set size is $O(n^2l^2)$.

Considering the two cases together and observing that each edge can generate one extra edge by each remaining obstacle, we have the total edge complexity as $O(n^2Ql^2)O(Q) = O(n^2Q^2l^2)$.

Since every edge has at most two vertices and each vertex belongs to the closure of an edge, the size of vertex set is equal to the size of the edge set. Also, every edge is created because of three touches; so locally around each edge we can find at most three polygons(each polygon is created by a two touch). Therefore the size of face set also is the same as the edge set. ■

Suppose P is a polygon in $D = \bigcup_{i,j,T} C(T)$. Let $\tilde{P} = \{x \in P : d_i(x) \geq 1 \forall i\}$. We call one connected component of $\tilde{P}$ with connected relative interior a feasible polygon. Then the following holds.

Theorem 4.2 Suppose there are Q obstacles. Then the size of each of the vertex set, edge set and the face set of the feasible polygons generated by O_i's is $O(n^2Q^2l^2)$.

Proof The following result is easy to establish. Suppose P is a polygon in D. Then set $\tilde{P}$ is either ϕ, or whole of P or is such that $\tilde{P}$ and $P \setminus \tilde{P}$ are seperated by a straight line $a \cdot x = c$. i.e. $\forall y \in \tilde{P}$ $a \cdot y \geq c$ and $\forall z \in P \setminus \tilde{P}$ $a \cdot z < c$. Thus each polygon P in D has at most one straight line L dividing it to generate $\tilde{P}$. L can intersect each edge at most once, thus adding at most e edges where e is the number of edges for P. Similarly, number of vertices can increase by at most e, and number of polygons by at most e. ■

In a recent paper[13], de Berg et.al. showed that for any two given points p and q which are path connectible, there exists a piecewise linear path between p and q which is of size $O(Q^4)$, although they do not give an algorithm. This is an interesting topological result since the complexity does not involve n and l; in fact this result holds for general convex obstacles and object. In comparison to this result, theorem 4.2 should be viewed from a different perspective because of the following two reasons: (i) theorem 4.2 specifically applies to piecewise linear paths with Voronoi properties; (ii) the Voronoi diagram of theorem 4.2 is a global structure using which it is easy to determine a path between any two path connectible points.

Now we define the basic component of our roadmap.

Definition 4.1 The *skeleton* is the skeleton of cells $\{C_i\}$, $i = 1, \ldots, Q$.

Note that by theorem 4.1, the skeleton size is $O(n^2Q^2l^2)$.

5. Algorithm for construction of skeleton of $\{C_i\}$ in $\mathcal{F}$. Here we give an informal description of the algorithm for construction of the skeleton of $\{C_i\}$ in $\mathcal{F}$. Full details can be found in [12].We will assume that $\mathcal{F}$ is bounded.

The M-distance between an obstacle O_i (described as $O_ix \leq a_i$) and the moving object M (described as $Py \leq q$) from a point p can be found by solving the linear program(LP) : min λ such that $P(x-p) \leq \lambda q, O_ix \leq a_i, \lambda \geq 0$, where x is the M-closest point to p on $O_i, \lambda = d_i(p)$. The complexity of solving this LP is linear in number of constraints[14], i.e. $O(n_i + l)$.

We start by finding an initial starting point. Consider two obstacles O_i and O_j. Then minimization of λ subject to $P(x-p) \leq \lambda q, P(y-p) \leq \lambda q, O_ix \leq a_i, O_jy \leq a_j, \lambda \geq 0$ (where x and y are the M-closest points to p on O_i and O_j respectively, p is the position of v_{ref}, and $\lambda = d_i(p)$) gives an equidistant point p from O_i and O_j if one exists. Then find the M-distances from p to $O_k \forall k = 1, \ldots, Q, k \neq i, j$. If every $d_k(p) \geq \lambda$, then p is a feasible starting point. Else choose a new pair (i, j) and repeat. Note that this process terminates because we have only finite number of obstacles and when the free space is bounded, one such pair will eventually generate one valid p.

Next we check if this point lies on an edge or a vertex(the check is done easily beacuse for every edge point three touches are active and for every vertex four touches are active, and while finding $d_k(p)$ we can easily check for this). If so, we are already on the skeleton. If not, then p is on an open face of $bd(C_i)$. Then we move along a random direction on the face and using propositions 3.1 and 3.3 formulate another LP to find a point on the skeleton along the chosen direction. Because $\mathcal{F}$ is bounded, any random direction is sufficient.

Once a point on the skeleton is reached, we start constructing the skeleton as follows. The LP which took us to the skeleton also gives information about the complete type of touch description at this point. Using this information, it is possible to find analytic expression(s) of the edge(s) which contain(s) this point. To find the delimiters of the edge(s)(vertices), we formulate LPs again. Thus, we construct successive vertices, and continue constructing the skeleton component polygon by polygon. Note that the word polygon is used in a loose sense to identify a polygonal boundary in a skeleton component.Thus we get complete information about a connected component of the skeleton.

The above procedure suffers from one serious defect. Consider fig 5.1. Suppose this configuration is floating inside another box. If we choose the dimensions of the boxes properly, then P_2 is a polygon wholly generated by the three boxes B_1,B_2 and B_3 and P_1 is a polygon wholly generated by B_1, B_2 and the outside box. There is a connected component of the skeleton which contains P_1 and another component which contains P_2; and these two are disjoint although both belong to the same connected component of free space. Thus the above algorithm must identify each connected component of the skeleton. Also, to make sure that in one connected component of $\mathcal{F}$ the skeleton is connected, it must have a way of joining such disconnected skeleton components. The following definition is useful in this respect.

Definition 5.1 A polygon P is said to be contained in another polygon P_1 if $P \subset relint(P_1)$.

In fig 5.1, P_1 contains P_2.

The following result can be established. Consider a triple $(i, j, k), i, j, k \in \{1, \ldots, Q\}, i \neq j \neq k$. Then there exists only one connected skeleton component where O_i, O_j, O_k contributes a point. Also, suppose a polygon P is contained and the obstacles associated with its edges are O_i, O_j and O_k, with only O_k active in its interior. Then the triple (i, j, k) cannot form a polygon which contains another polygon T such that the edges of T involves O_i, O_j and O_l with O_l active in its interior.

We also prove the very important result that *if a single obstacle has two disconnected skeleton components associated with it in one connected component of $\mathcal{F}$ then one of them has a polygon contained in a polygon of the other.* Using these results, we modify the algorithm as follows. For every polygon constructed, we check if it is a contained one. Suppose this polygon is P and it is a contained one. Suppose the obstacles active on the edges of P are O_i, O_j and O_k, with O_k being active in the interior of P, and x is a point on $bd(P)$. Note that this means we already know that O_i, O_j are involved in the container polygon. We choose a random direction d in the plane of P. In this direction we find the first point where an obstacle $O_l, \forall l \neq i,j,k$ has the same M-distance as O_i(and thus O_j). These points can be found by solving a series of LPs. Also, since the container polygon may involve only O_i and O_j, we solve another LP to find out the first point where only the pair O_i and O_j contribute three touches. All these points are candidate points for the container polygon as for each of these points three touches are active. However, all of them may not be feasible. We sort these points according to their distances from x in increasing order. Now we scan the list from left to right. Each point has a triple (i,j,l) associated with it. If this triple is such that l is one of i or j then this is the container polygon. If this triple is not encountered before, we start constructing this skeleton component and in the process identify whether this is the container polygon. If this triple is encountered before and generated a contained polygon then we skip to the next point, else this is the container polygon. In the process, all such contained polygons can be connected using artificial wholly feasible straight line edges, and the last of them can be connected to the container polygon. The check if a triple is encountered or not can be done in constant time by maintaining an array of all triples where each triple is assigned a pre-fixed place, and storing information about whether this triple created a contained polygon and if so, whether already joined. Thus we get complete contained-container information, and we join the contained polygons to the container using artificial straight line edges, which are wholly feasible if the endpoints are feasible. The final result is a connected skeleton. These artificial edges do not change the data size of the output as for every contained polygon we need only one extra edge. Also, the process of constructing the complete skeleton over all connected components of free space is finite by virtue of the following proposition which can be proved using finite induction over the number of obstacle faces.

Proposition 5.1 If $\mathcal{F}$ has two disconnected components $\mathcal{F}_1$ and $\mathcal{F}_2$ then there exist faces f_1 of some O_i and f_2 of some O_j such that f_1 contributes a point to $\mathcal{F}_1$ but not to $\mathcal{F}_2$, and f_2 contributes a point to $\mathcal{F}_2$ and not to $\mathcal{F}_1$.

Thus checking whether each face of each obstacle has contributed a point to a skeleton component suffices as a test for termination.

Theorem 5.2 The complexity of the algorithm is $O((n+Ql)e+Q^3)$, where e is the number of edges in the skeleton. In the worst case, e is $O(n^2Q^2l^2)$ and the complexity is $O(n^3Q^2l^2 + n^2Q^3l^3)$.

The skeleton, as formed by the algorithm is enough if we are interested only in the Voronoi diagram of the polyhedra. However, for motion planning we need to consider only a subset of this diagram, namely the set $R = \{x : x \in \text{skeleton}, d_i(x) \geq 1\ \forall i = 1,\ldots,Q\}$. The next result concerns that.

Theorem 5.3 The roadmap R can be constructed in worst case $O(n^3Q^2l^2 + n^2Q^3l^3)$ time.

6. Motion Planning. In the previous section, we sketched the algorithm to construct the roadmap. That this roadmap is useful for motion planning is established by the following completeness theorem.

Theorem 6.1 Suppose the feasible free space $\tilde{\mathcal{F}}$ is bounded. Then for every connected component of $\tilde{\mathcal{F}}$, the roadmap R is connected.

Proof For a cell C_i, consider the set $\{x \in C_i : d_i(x) \geq 1\}$. We call this the feasible set of cell C_i. Note that this set may be ϕ, one connected component or union of several connected components. We call each of these connected components a *feasible cell.*

Let $\overline{C}$ be a connected component of $\tilde{\mathcal{F}}$. Also suppose the proposition is not true for $\overline{C}$. Then there exist disconnected components of R in $\overline{C}$. Since number of feasible cells is finite, there exist only finite such disconnected components. Let these be $R_1, R_2, \ldots, R_r$.

Let the feasible cells associated with R_i be $F_{i_1}, \ldots, F_{i_k}$. Since $\tilde{\mathcal{F}}$ is bounded, every feasible cell boundary consists of closed polygons. Thus no feasible cell of R_i can intersect a feasible cell of $R_j, j \neq i$ because if they do they must intersect at the boundary and so we either have an intersection at the skeleton or a containment and in both cases the algorithm generates a connected skeleton.

Now consider $p \in R_1, q \in R_2$. Since $R_1, R_2 \in \overline{C}$, there exist path $PATH$ from p to q in $\overline{C}$. Since there exist only finite number of R_i's, there exist a point $w \in PATH$ such that $w \notin F_{i_j} \forall i = 1,\ldots,r, \forall j = 1,\ldots,k$. But then there does not exist any feasible cell containing w which implies w is not feasible. But $w \in \overline{C}$ and we reach a contradiction. ∎

Given this, the idea of motion planning is as follows. For any given starting point $s \in \tilde{\mathcal{F}}$, we map s to a point on the roadmap. Call this point s_r. Similarly, the final point $f \in \tilde{\mathcal{F}}$ is mapped to f_r. The mapping is well defined, and we solve an LP for each mapping. The complexity of the mapping is $O(n + Ql)$. There exists path from s to f iff there exists path form s_r to f_r on the roadmap. We search on the roadmap for a path from s_r to f_r. The complexity of search is $O(e)$ where e is the number of edges in the roadmap.

We maintain the following data structure. We form an abstract graph with polygons as vertices, two polygons P_1 and P_2 are connected if they share an edge or a vertex in the skeleton. Contained polygons also form nodes in the graph but they are connected to their adjacent polygons as well as to those polygons to which they are connected by artificial edges. When s and f are specified, they are mapped to the skeleton and a search process is initiated on the abstract graph, and path is found on the graph. The polygons involved in the abstract graph are then traced to find the actual path on the skeleton.

References

[1] C. Ó. Dúnlaing, M. Sharir and C.K. Yap, *" Generalized Voronoi Diagrams for a Ladder : II. Efficient Construction of the Diagram"*, Algorithmica, Vol 2, pp 27–59, 1987.

[2] R.A. Brooks, *" Solving the Find-Path Problem by Good Representation of Free Space"*, IEEE Trans on Syst, Man and Cyb., 13(3), pp 190–197, 1983.

[3] J.Canny, The Complexity of Robot Motion Planning, MIT Press, cambridge, MA, 1988.

[4] D. Leven and M. Sharir, *" Planning a Purely Translational Motion for a Convex Object in Two-Dimensional Space Using Generalized Voronoi Diagrams"*, Disc. and Comput. Geo., Springer-Verlag, Vol 2, pp 9–31, 1987.

[5] J.Canny and B.Donald, *" Simplified Voronoi Diagrams"*, Disc. and Comput. Geo., Springer-Verlag, Vol 3, pp 219–236, 1988.

[6] S. Stifter, *" An Axiomatic Approach to Voronoi Diagrams in 3D"*, J. of Computer and Syst. Sc.,Vol 43 No 2, pp 361–379, 1991.

[7] A. Sanderson, RPI, Troy, New York, *Personal Communication.*

[8] V. Srinivasan, IBM T.J. Watson Research Center, New York, *Personal Communication.*

[9] S.S. Keerthi, N.K. Sancheti and A. Dattasharma, *" Transversality Theorem : A Useful Tool for Establishing Genericity"*, Proc. IEEE Conf. on Decision and Control, 1992.

[10] J. Canny, *" A Voronoi Method for the Piano Mover's Problem"*, Proc. IEEE Int. Conf. Robotics and Auto., St. Louis, Mo, 1985.

[11] R.T. Rockafeller, Convex Analysis, Princeton University Press, Princeton, New Jersey, 1972.

[12] A. Dattasharma and S.S. Keerthi, *" Rectilinear Voronoi like Roadmap for Translational Motion of a Convex Polyhedron Amidst Convex Polyhedra in a 3D Euclidean World"*, Technical Report #02-93, Indian Institute of Science, June 1993.

[13] M. de Berg, J. Matousek and O. Schwarzkopf, *"Piecewise Linear Paths Among Convex Obstacles"*, Proc. 25th ACM STOC, pp 505–514, 1993.

[14] N. Meggido, *"Linear Programming in Linear Time When the Dimension is Fixed"*, J. ACM., Vol 31, pp 114–127, 1984.

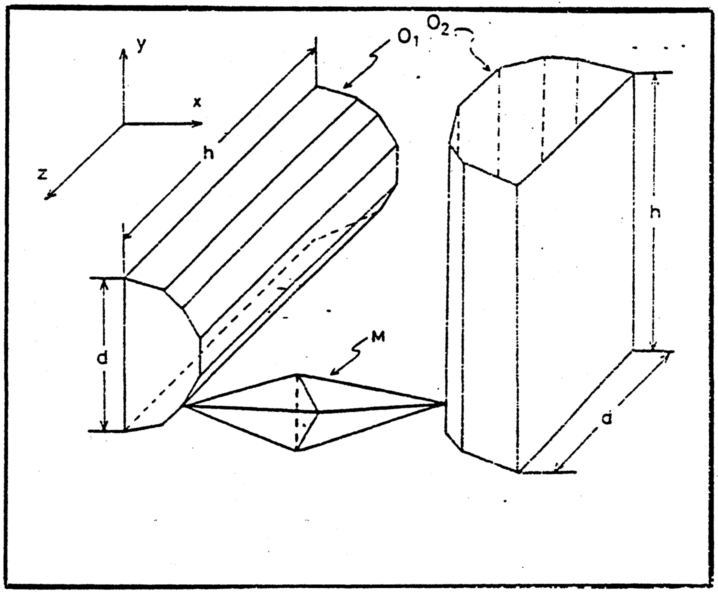

Fig 4.1

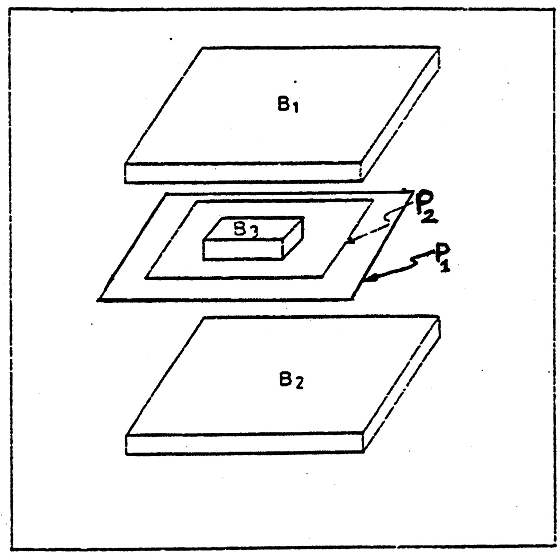

Fig 5.1