# SHORTEST PATHS BETWEEN SOURCE AND GOAL POINTS LOCATED ON/AROUND A CONVEX POLYHEDRON

W. Randolph Franklin and Varol Akman
Electrical, Computer, and Systems Engineering Dept.
Rensselaer Polytechnic Institute
Troy, New York 12181

## ABSTRACT

Determining the minimal free paths between two given points inside a 3-D workspace cluttered with rigid polyhedra is an outstanding open problem in Robotics. In this paper, we solve three simple instances of this problem which is commonly known as "Findpath." Our instances always treat the case of a single convex polyhedron with two points located on/around it. The solution of the third instance consists of an extension of Voronoi diagrams to the surface of a polyhedron. Thus, given several fixed points on the polyhedron, we wish to preprocess the surface so that we can find both the closest fixed point, and the minimal path, to a new point specified on the surface. This extension arises naturally from a consideration of minimal paths in 3-D where there are many polyhedral obstacles. Such a path will travel alternately through free space, and along a polyhedron's surface.

## 1. PROBLEM STATEMENT

A regular duty of a robot arm inside a 3-D "workspace" W (an imaginary bounding box in which several polyhedral objects reside) is to transfer, if possible, a given polyhedron from a given location to another while evading clashes with the other polyhedra (commonly termed as "obstacles"). This is known in Artificial Intelligence as the "Findpath" problem [Brooks 1983]. In this paper we solve three very specific instances of Findpath which currently stands as a very hard problem to conquer in its full generality [Schwartz and Sharir 1983; Hopcroft et al. 1984].

For the instances we treat here, we will compute the minimal (under the Euclidean metric) "free" (obstacle-avoiding) paths between two points assuming that only one point (normally chosen as the fingertip of the robot arm) is being moved in the presence of only one convex obstacle. It is noted that there exists a well-known method to shrink a moved (strictly speaking, translated) object to a point while simultaneously enlarging all the obstacles in W [Lozano-Perez and Wesley 1979; Lozano-Perez 1983]. Thus a reduction of a "Polyhedron Findpath" problem to a "Point Findpath" problem of the sort presented here is validated for a broad class of practical problems.

Recently, there has been much work on related problems in the computational geometry of motion-planning in addition to those mentioned above. The reader is referred to [Akman 1984] for a relatively rich set of references. Among these, especially [Donald 1983; O'Dunlaing and Yap 1983; Spirakis and Yap 1983; Sharir and Schorr 1984] are relevant to our work. Sharir and Schorr's work is especially interesting in that it mentions many useful results on the nature of minimal paths on a convex polyhedron. The crux of their paper is the following result

which is arrived after employing complex (and thus hard to program) data structures and algorithms:

> "Given a convex polyhedron P with n vertices and a point s on it, P can be preprocessed in $O(n^3 \log n)$ time to produce a data structure (taking $O(n^2)$ space) with the help of which one can find in $O(n)$ time the minimal path along the surface of P from s to any other specified point g."

## 2. INSTANCES

Let P be a convex polyhedron and s and g ("source" and "goal") two points in the 3-D Euclidean space. Assume that $s, g \in Ext(P) \cup On(P)$ throughout this paper. (On(P), Int(P), and Ext(P) denote the surface, the interior points, and the exterior points of P, respectively.) In the sequel, we first present algorithms to compute the minimum-length s-to-g (or vice versa) paths which do not interfere with P (i.e., at most touch P but never intersect it) for the following two instances of Findpath:

> [FP1] $s, g \in On(P)$ and both points are given; and
> [FP2] s and/or $g \in Ext(P)$ and both points are given.

Finally, as a third and more interesting instance, we solve the following "Single Source-Many Goals" version of Findpath. Let s be fixed and g be varied inside W. This corresponds to a manipulator consuming a pile of objects by continuously moving them to different locations inside W. (The symmetric case of a manipulator picking up objects from several piles in order to assemble them in a fixed location is transformable to this instance by simply trading the roles of s and g.) Assume that s and the particular face $F_g$ of P on which g is guaranteed to be located are given. (It is emphasized that the exact location of g on $F_g$ is not known yet.) Clearly, the aim is to do some preprocessing using this limited information (i.e., P, s, and $F_g$) so as to compute the s-to-g minimal paths efficiently for specified "query" points which will generically be denoted by g. In our terminology:

> [FP3] $s, g \in On(P)$; s is given while g is not (but $F_g$, the face of P that g belongs, is known). We are allowed to preprocess to quickly answer later queries that specify many $g \in F_g$ and ask for the s-to-g minimal paths.

## 3. ALGORITHMS

In the following, $\alpha_0$, $\alpha_1$, and $\alpha_2$ denote the number of vertices, edges, and faces of P, respectively. We reserve the terms "edge," "vertex," and "path" only for polygons and polyhedra; for graphs we use the less common terms "link," "node," and "sequence" in order to avoid potential confusions.

### 3.1. Solution of FP1

Throughout this paper, we assume that the line segment [sg] is not the shortest path since there is a very fast algorithm to detect this, that is, to intersect a line segment with a convex polyhedron P [Chazelle

1980]. This algorithm works in time $O(\log^2 \alpha_0)$.

In the following, we summarize a simple but useful fact about the shortest paths on a dihedral angle.

Lemma 1.1 (Dihedral Principle): If s and g are on different faces of a dihedral angle D, then the shortest path between them is [sx] U [xg] where x is a point on the common line cc' of the faces such that $\langle sxc = \langle c'xg$ where "$\langle$" denotes a planar angle.
Proof: Trivial. Algorithmically, x is found as follows. Rotate g about cc' until it touches to the plane of s (but specifically to the half-plane (described by cc') not including s). Thus g is transformed to g'. This is called a "development" of D into the plane. Draw [sg'] and intersect it with cc' to find x. When we fold the faces back to their original position the minimal path is placed into three dimensions on D again. □

To solve FP1 we will develop a partly geometric and partly graph-theoretic machinery. By its definition, a polyhedron is a figure in space formed by a finite number of polygons situated in a certain mutual relationship. This can be represented by a system of polygons ("evolvent" or "planar polygonal schema") in the plane [Frechet and Fan 1967].

Given a polyhedron P, we associate a polygon with each of its faces, this polygon being subject to the single condition of having the same metrical form as the associated face of P. We thus obtain a finite number of polygons in the plane each of which is separated from the others. We will couple the edges of these polygons in pairs in such a way that two "coupled" edges in the plane lead to the same edge of P. Now denote each coupled edges in the plane by the same letter. This does not yet suffice to describe the mutual relationship of the faces of P. To make it precise, each edge in the plane must be oriented by placing an "arrow" on it in an arbitrary sense, except that for two coupled edges the heads of the two arrows on them coincide when they are identified in forming the corresponding edge of P.

The evolvent $\Delta$ of P, obtained as described above, defines the intrinsic geometry of P in the sense that for a given $\Delta$ it is possible to determine the minimal paths between two points on P [Aleksandrov 1958]. In our solution of FP1, $\Delta$ is the main geometric tool. It is noted that $\Delta$ satisfies the following conditions:

> i) the coupling ("gluing" in Aleksandrov's terminology) of polygons may occur either at edges or vertices;
> ii) if two polygons $F_i$ and $F_j$ are glued at vertex v, then they are either glued together along edges containing v, or there exists a sequence of polygons, starting at $F_i$ and ending at $F_j$, which are glued to one another along edges containing v; and
> iii) each edge is glued to exactly one edge.

In addition to $\Delta$, we need another simple tool. Define the "face adjacency" (or shortly, face) graph $\Omega = (V, E)$ as an undirected graph with unit links where $V = \{v \mid F_v$ is a face of P$\}$ and $E = \{(v_1, v_2) \mid F_{v1}$ and $F_{v2}$ are adjacent along an edge of P$\}$. Now let t be a minimal

path on P from s to g. The faces that t touches while going from s to g define the "face visit sequence," $F_s, \ldots, F_g$, where each face in this sequence is adjacent to (shares an edge with) its preceding and following neighbors.

Lemma 1.2: A face visit sequence in Ω associated with a minimal path on P is "simple" (loop-free).
Proof: After recalling the fact that each face of P is also convex, one would further shorten the portion(s) of the path corresponding to the cyclic sequence(s). But then the given path is not minimal. □

Lemma 1.3: There may be O(n!) simple sequences between two specified nodes of an undirected graph with n nodes.
Proof: In the worst-case the graph is complete. (But see the argument below.) □

The number of links in Ω is much less than the number of links in a complete graph with the same number of nodes. This is due to the observation that the surface of a convex polyhedron has the structure of a planar graph; hence it can have only a linearly growing number of links in terms of its nodes. This O(n) factor of reduction in the number of links renders less freedom in moving from one node to another and helps reduce the number of simple sequences considerably.

Before we present our algorithm for FP1 we emphasize that Δ is a geometric data structure. Hence it is possible, for instance, to obtain from Δ such information as "$F_1$ and $F_4$ share edge e," or "$F_6$ and $F_3$ are disjoint." With these examples, a convenient internal representation for storing an evolvent should be self-evident (cf. [Abelson and DiSessa 1982], for example). On the other hand, Ω contains only topological information; that is, it is less informative of shape.

After above preparations we can now present our algorithm for FP1:

> ALGORITHM A1.
> 1. Given s and g, find the faces $F_s$ and $F_g$ holding them. Assume that they are different; otherwise we are finished with path [sg].
> 2. Mark these two faces in Ω and enumerate all simple sequences between these nodes in any order.
> 3. For each sequence (which may be thought as a face visit sequence) compute the associated development of the relevant faces. This is accomplished using the information provided by Δ.
> 4. For each development compute the minimal path between the images of s and g and finally choose the smallest one(s) among them.
> END A1.

In Figures 1, 2, and 3 we demonstrate this algorithm on a cube. Figure 1 depicts the evolvent and the face graph of the given cube. Figure 2 lists first few simple face visit sequences. Finally, Figure 3 shows the corresponding developments for the sequences given in Figure 2.

To the best of our knowledge, counting the number of simple sequences between two nodes of a graph is a difficult problem. This is due to the fact that given graph G = (V, E), |V| = n, |E| = m, length l(e) ε $Z^+$ for

each e ε E, specified nodes s, g ε V, and positive integers B and K, it is stated in [Garey and Johnson 1979] that the problem "Are there K or more distinct simple paths from s to g in G, each having total length B or less?" is NP-hard. Although this problem is not known to be in NP, the corresponding enumeration problem is #P-complete [Valiant 1979]. It is emphasized that #P-completeness still holds even when G is planar.

In [Yen 1971], an $O(Kn^3)$ time algorithm is given to enumerate the sequences in increasing path length. Recently, [Katoh et al. 1982] gave a new improved algorithm which works in time $O(Kc(n, m))$ if the shortest sequences from one node to all the other nodes are obtained in $c(n, m)$ time. Since $c(n, m)$ is at most $\min(O(n^2), O(m \log n))$ this algorithm is more efficient than Yen's.

## 3.2. Solution of FP2

When s and/or g are outside a convex polyhedron we obtain the instance FP2. In this case, it is still possible to apply the method summarized in Section 3.1 but only after a simple transformation of P to another polyhedron P' which now has s and g necessarily on its surface. The transformation step is given as follows:

ALGORITHM A2.
1. Compute the two visible outlines (silhouettes) $Z_s$ and $Z_g$ of P from the viewpoints s and g, respectively. These are generally nonplanar polygons which have as edges some edges of P.
2. Construct a new polyhedron P' which is the combination of three polyhedra, $P' = Q_s \cup P \cup Q_g$. Here $Q_s$ is the pyramid having s as apex and $Z_s$ as its base outline, and $Q_g$ is the pyramid having g as apex and $Z_g$ as its base outline. Note that these pyramids have generally concave bases consisting of the visible faces of P from s and g, respectively.
END A2.

The correctness of Algorithm A2 follows from the following properties:

Lemma 2.1: The respective planar projections of $Z_s$ and $Z_g$ from s and g are convex polygons.
Proof: Due to fact that a convex polyhedron casts a convex shadow when illuminated by a point source. □

Lemma 2.2: P' is also a convex polyhedron.
Proof: Take two points x and y inside P'. If $x \varepsilon \text{Int}(Q_s)$ and $y \varepsilon \text{Int}(P)$ then [xy] is always inside $Q_s \cup P$ for y will always be inside the infinite pyramid originating from s. A similar argument shows that the union of $Q_g$ and P is also convex. The only unproved case is when $x \varepsilon Q_s$ and $y \varepsilon Q_g$. Let us check the dihedral angles of P' where $Q_s$ and P, and $Q_g$ and P are joined. Clearly, those angles are all nonreflex implying that [xy] must be totally contained inside P'. □

The following result shows that after the problem-reduction step (i.e., the computation of P' from P) the complexity of this instance is the same as with FP1.

Lemma 2.3: $\text{Size}(P') = O(\text{Size}(P))$.

Proof: The "size" of a polyhedron can be defined as the number of its vertices, edges, or faces depending on choice. In the case of a convex polyhedron they are all asymptotically equivalent since a convex polyhedron can be embedded in the plane as a graph and since $\alpha_1$ is at most equal to $3\alpha_0 - 6$ in a planar graph. Hence, choose $\alpha_1$ as the size measure. From Lemma 2.2, $Q_s$ and $Q_g$ may contribute $O(\alpha_1)$ new edges in the worst-case. □

How fast can we compute $Z_s$ and $Z_g$? Below we summarize the computation of $Z_s$; $Z_g$ is computed similarly. Find a point $c_i$ inside every face of P. Compute, using Chazelle's method, the intersections of P and $[sc_i]$, $i = 1, \alpha_2$. If the intersection corresponding to a particular $c_i$ is null then $F_i$ is visible from s; else it is obstructed by other faces of P. In this manner all visible faces of P from s are found. This clearly takes $O(\alpha_2 \log^2 \alpha_0)$, or equivalently $O(\alpha_0 \log^2 \alpha_0)$ time. Make a list of the edges of these visible faces to compute $Z_s$. Any edge which appears twice in this list cannot belong to $Z_s$. Thus, the edges of $Z_s$ are determined in $O(\alpha_0 \log \alpha_0)$ time via sorting this list and eliminating both elements of duplicate pairs. To get $Z_s$ with ordered edges from this unordered set, take any point q inside any of the visible faces and sort the vertices of this set of edges anglewise about q. As soon as we find $Z_s$, we know all the faces of $Q_s$. Once we have $Q_s$ and $Q_g$, P' can be easily constructed (Figure 4).

## 3.3. Solution of FP3

To solve FP3, we follow a Voronoi-based strategy [Franklin 1982; Franklin et al. 1983; Guibas and Stolfi 1983]. We want to partition the face $F_g$ into subregions via a planar subdivision made of straight edges such that if a query point g is discovered to be belonging to a particular subregion of this subdivision, it must be possible to list the face visit sequence that should be taken by an s-to-g minimal path. The following algorithm accomplishes this:

ALGORITHM A3.
I. PREPROCESSING:
1. Develop onto the plane of $F_g$ all simple visit sequences in $\Omega$ between nodes $F_s$ (the face of P holding s) and $F_g$ using $\Delta$. At the end of this process, one obtains, using the Dihedral Principle, a set of points in the plane which are the "images" of s under given visit sequences.
2. Store with each image point, the visit sequence to arrive it.
3. Using standard algorithms construct the Voronoi diagram V which has these images as Voronoi centers [Guibas and Stolfi 1983].
4. "Clip" V against $F_g$. This simply means finding the subset V' of V included within the polygon $F_g$.
II. QUERYING:
1. Given a new query point g, first make sure that $g \in F_g$.
2. Using standard point location algorithms locate the subregion R including g in subdivision V' [Lee and Preparata 1977].
3. Using the stored sequence in the Voronoi center corresponding to R develop the faces specified by this sequence and find the minimal path with straightforward application of the Dihedral Principle.
END A3.

Once we preprocess the given scene for a given s and $F_g$, for each goal g we must determine which region of the subdivision V' it belongs to. Assume that V' has y edges. In [Lee and Preparata 1977], an algorithm is given which, after O(y log y) preprocessing time, requires $O(\log^2 y)$ search time. A faster algorithm that is based on triangulating the planar graph has the same preprocessing time but O(log y) search time [Kirkpatrick 1983].

The size y of V' depends on two things: the number of simple visit sequences between $F_s$ and $F_g$ in $\Omega$, and the shape and relative location of $F_g$. If $F_g$ is very big, then it will probably include most of V. If it is almost coincident with the most crowded parts of V then V' will again have many subregions. Similarly, if there are many visit sequences then the Voronoi diagram V will have many edges. In general, the complexity of constructing V on the surface of P for many specified source points will depend on the average number of regions that each face is partitioned into.

In Figure 5, we demonstrate this preprocessing method for a cube. The lists written next to each image of s in this figure specify the order of unfolding applied to arrive that point. Not all images of s are shown. Notice that if g is found to be inside the vertically hatched subregion then the minimal path is via the unfolding of simple sequence (3,2,6). On the other hand, it is via the unfolding of (3,6) when g is inside the horizontally hatched subregion.

A dynamizing technique which would allow good insertion time for an incoming point in the Voronoi diagram proves very useful in this context. Thus, we start with only a few image points (commonly corresponding to the first few short simple sequences in the face graph) and work our way to an "incomplete" Voronoi diagram easily. When we want to see the effect of another image point (not yet tried) we use a dynamic data structure to insert it. In [Gowda et al. 1983], it is proved that this is achievable in O(y) time if the Voronoi diagram has y points prior to insertion.

## ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grants No. ECS-8021504 and ECS-8351942.

## REFERENCES

ABELSON, H., AND DISESSA, A. 1982. Turtle Geometry: The Computer as a Medium for Exploring Mathematics, MIT Press, Cambridge, Mass.

AKMAN, V. 1984. Findminpath algorithms for task-level (model-based) robot programming. Manuscript, Electrical, Computer, and Systems Eng. Dep., Rensselaer Polytechnic Inst., Troy, N.Y. (Mar.)

ALEKSANDROV, A. D. 1958. Konvexe Polyeder (German, translated from Russian), Akademie-Verlag, Berlin.

BROOKS, R. A. 1983. Solving the Findpath problem by good representation of free space. IEEE Trans. on Systems, Man, and Cybernetics 13, 3 (Mar.), 190-197.

CHAZELLE, B. M. 1980. Computational geometry and convexity. Ph.D. dissertation, Computer Science Dep., Yale Univ., New Haven, Conn.

DONALD, B. R. 1983. The Mover's problem in automated structural design. In Proc. of the Harvard Computer Graphics Conf., Cambridge, Mass. (July)
FRANKLIN, W. R. 1982. Partitioning the plane to calculate minimal paths to any goal around obstructions. Tech. Rep., Image Processing Lab, Rensselaer Polytechnic Inst., Troy, N.Y. (Nov.)
FRANKLIN, W. R., AKMAN, V., AND VERRILLI, C. 1983. Voronoi diagrams with barriers and on polyhedra. Tech. Rep., Image Processing Lab, Rensselaer Polytechnic Inst., Troy, N.Y. (Dec.)
FRECHET, M., AND FAN, K. 1967. Initiation to Combinatorial Topology (translated from French), Prindle, Weber, and Schmidt, Inc., Complementary Series in Mathematics, Vol. 7, Boston, Mass.
GAREY, M. R., AND JOHNSON, D. S. 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, San Fransisco, Calif.
GOWDA, I. G., KIRKPATRICK, D. G., LEE, D. T., AND NAAMAD, A. 1983. Dynamic Voronoi diagrams. IEEE Trans. on Information Theory 29, 5 (Sep.), 724-731.
GUIBAS, L., AND STOLFI, J. 1983. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. In Proc. of the 15th ACM Symp. on the Theory of Computing (Apr.), 221-234.
HOPCROFT, J. E., SCHWARTZ, J. T., AND SHARIR M. 1984. On the complexity of motion planning for multiple independent objects: PSPACE hardness of the "Warehouseman's Problem." Tech. Rep., Courant Inst. of Mathematical Sciences, New York Univ., New York (Feb.)
KATOH, N., IBARAKI, T., AND MINE, H. 1982. An efficient algorithm for k shortest simple paths. Networks 12, 411-427.
KIRKPATRICK, D. G. 1983. Optimal search in planar subdivisions. SIAM Journal on Computing 12, 1 (Feb.), 28-35.
LEE, D. T., AND PREPARATA, F. P. 1977. Location of a point in a planar subdivision and its applications. SIAM Journal on Computing 6, 3 (Sep.), 594-606.
LOZANO-PEREZ, T. 1983. Spatial planning, a configuration space approach. IEEE Trans. on Computers 32, 2 (Feb.), 108-120.
LOZANO-PEREZ, T., AND WESLEY, M. A. 1979. An algorithm for planning collision-free paths among polyhedral objects. Comm. of the ACM 22, 10 (Oct.), 560-570.
O'DUNLAING, C., AND YAP, C. K. 1983. The Voronoi diagram method of motion planning I: the case of a disc. Tech. Rep., Courant Inst. of Mathematical Sciences, New York Univ., New York.
SCHWARTZ, J. T., AND SHARIR, M. 1983. On the Piano Mover's problem I: the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. Comm. on Pure and Applied Mathematics 36, 3 (May), 345-398.
SHARIR, M., AND SCHORR, A. 1984. On shortest paths in polyhedral spaces. In Proc. of the 16th ACM Symp. on the Theory of Computing, 144-153.
SPIRAKIS, P., AND YAP, C. K. 1983. On the combinatorial complexity of motion coordination. Tech. Rep., Courant Inst. of Mathematical Sciences, New York Univ., New York (Apr.)
VALIANT, L. G. 1979. The complexity of enumeration and reliability problems. SIAM Journal on Computing 8, 3 (Aug.), 410-421.
YEN, J. Y. 1971. Finding the k shortest loopless paths in a network. Management Science 17, 712-716.

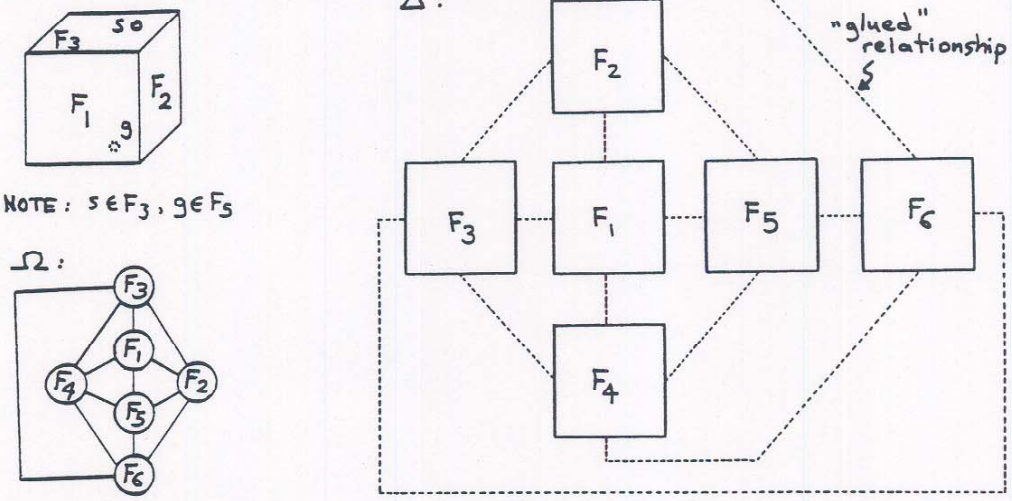



Figure 1

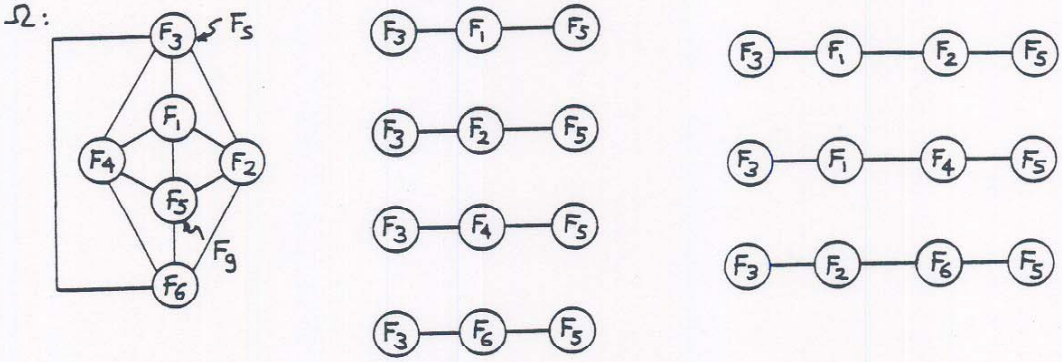



Figure 2

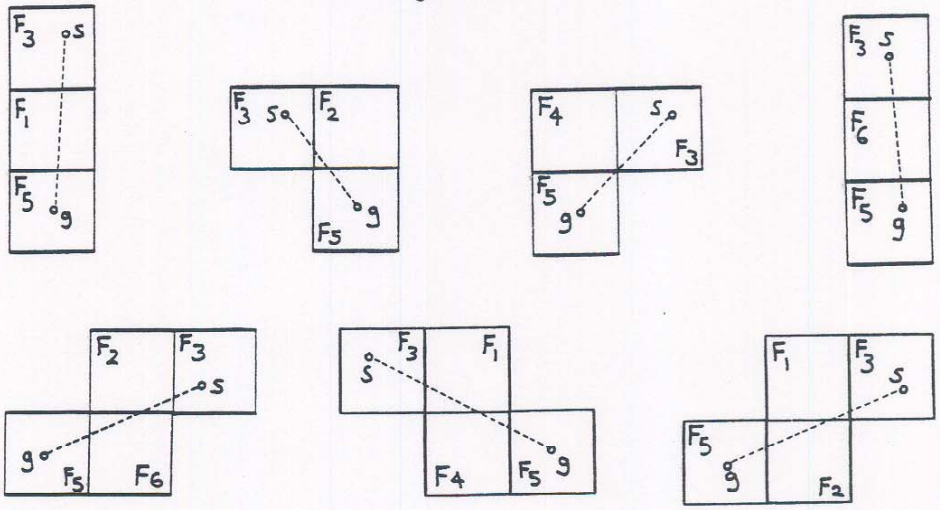



Figure 3

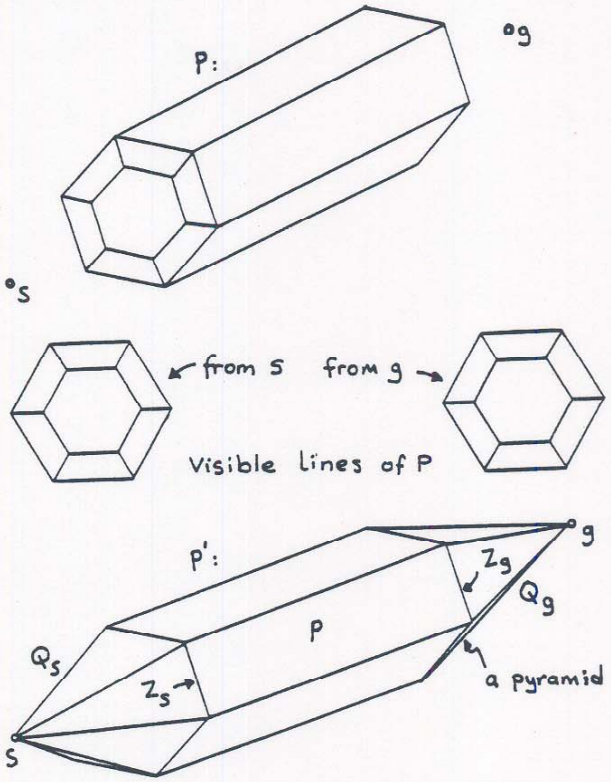
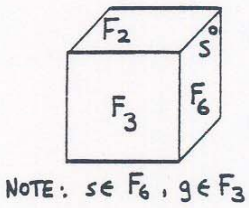



Figure 4



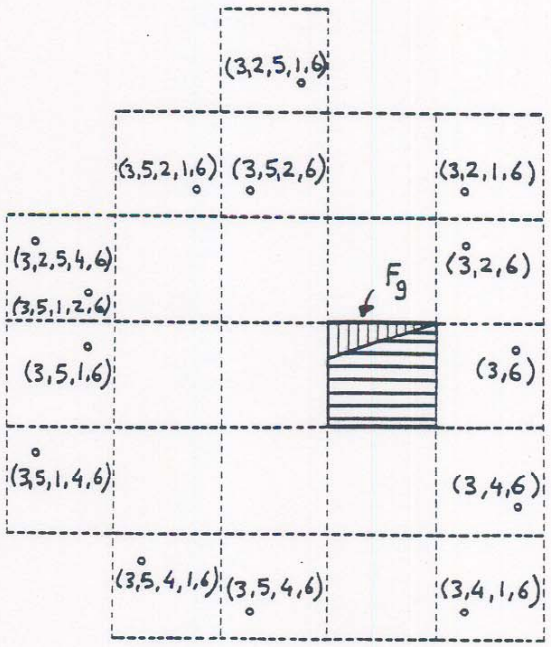




Figure 5