Criticality Considerations in the Design of Geometric Algorithms

Bernard Chazelle
Department of Computer Science
Brown University
Providence, RI 02912

Abstract

We examine a class of geometric problems which all share a feature of considerable relevance: the underlying set is defined implicitly and is at least *quadratically* larger than the input size. Is efficient computation possible without expanding the set explicitly? If not, are non-trivial lower bounds provable? We give partial answers to these questions by proposing various approaches for attacking these problems. In doing so, we improve a number of complexity bounds for problems of range searching, frame-to-frame coherence, segment stabbing, geometric predicate computation, etc.

1. Triangular Search

Let $S = \{p_1, \ldots, p_n\}$ be a set of n points in E^2. The triangular search problem is to preprocess S so that, for any query triangle T, the subset of points of S lying inside T can be computed efficiently. An $(f(n), g(n))$ solution refers to an algorithm for this problem that requires $O(f(n))$ space and $O(g(n) +$ output size) query time. The most efficient solutions to date are an $(n, n^{.695})$ algorithm by Edelsbrunner and Welzl [EW] and an $(n^{2+\epsilon}, \log n \log \frac{1}{\epsilon})$ by Cole and Yap [CY] (see also [EKM,W] for other methods). Our first observation is that quadratic space complexity seems to be the price to pay for fast retrieval. This can be understood by looking at the problem of determining whether a query line L passes through any point of S. Obviously, this is a subproblem of the original problem: simply make the triangle T infinitely long and skinny. Searching is (at least conceptually) facilitated by turning to a dual space, whereby lines becomes points and points become lines. The dual of S is an arrangement of n

lines, so the subproblem is reduced to locating the dual point of L in its $O(n^2)$ possible (topologically distinct) locations. This suggests a straightforward $O(n^2, \log n)$ solution via planar point location, but also points to the difficulty of escaping this costly approach. Cole and Yap's solution to triangular search can be understood in this light: its high space complexity comes from its enforced discrimination among *critical* positions. We propose to show that interestingly the subproblem where T becomes a skinny elongated triangle (going towards a line) is not just *some* hard subproblem: it is the *only* hard subproblem. We will prove that an efficient solution always exists provided that T is not exceedingly lean. Specifically, we require that each angle of T should be bounded below by a constant α.

Theorem 1. There exists an $(n \log n, \log^2 n)$ solution to the bounded-angle triangular search problem.

Proof: We need two preliminary results: to begin, we recall a result shown in [CG]. Given a set of S of n points, there exists an $O(n)$ data structure, denoted $T_L(S)$, s.t. for any query grounded right triangle q, the set $S \bigcap q$ can be computed in time $O(\log n + |S \bigcap q|)$ (a right triangle is grounded if one of its non-hypothenuse edges lies on a fixed line — the result in [CG] actually also applies to trapezoids, but this is not needed here). Consider now the original problem, with the added restriction that T be a right triangle with one of its non-hypothenuse edges parallel to a fixed line L. Wlog assume that no two points in S form a segment parallel to L. Let $\{p_1, \ldots, p_n\}$ be the points of S sorted according to the order of their projections on a normal to L. Let $L_0, L_1, \ldots, L_n$ be a set of lines parallel to L with p_i being the only point to lie between L_{i-1} and L_i. Let $\mathcal{T}$ be the complete binary tree with its left-to-right leaf sequence in one-to-one correspondence with $\{p_1, \ldots, p_n\}$. Note that each node v of $\mathcal{T}$ spans a subset $S(v)$ of S between some $L'(v) = L_i$ and $L''(v) = L_j$. For each node $v \in \mathcal{T}$, compute the two structures $T_{L'(v)}(S(v))$ and $T_{L''(v)}(S(v))$. This provides us with an $(n \log n, \log^2 n)$ solution for the restricted problem via a trivial canonical decomposition of the query triangle T into $O(\log n)$ (overlapping) triangles.

Next we turn to the general problem. Assume that each angle of T exceeds α. Let d_i be the unit vector with $\angle(\vec{x}, \vec{d_i}) = i\alpha$, defined for each i; $0 \leq i \leq \lceil \frac{2\pi}{\alpha} \rceil - 1$. We set up a data structure of the previous type for each direction in $\Delta = \{d_0, \ldots, d_{\lceil \frac{2\pi}{\alpha} \rceil - 1}\}$. The theorem will trivially follow once we show how to partition T into a constant number (≤ 12) of right triangles with one (non-hypothenuse) side parallel to some $d \in \Delta$. Let ABC be an arbitrary triangle, each of whose angles exceeds α. We can always assume wlog that each angle is at most $\pi/2$. If this is not the case, split T in two by drawing the angle bisector from the unique obtuse-angled vertex. With these conditions satisfied, there must exist a segment AA' with direction D_a and $A' \in BC$ (Fig.1). Let I_B (resp. I_C) be the intersection of AA' with the normal to AA' passing through B (resp. C). Since $\angle(AB, AC) \leq \frac{\pi}{2}$, T lies entirely on one side of the normal to AA' passing through A, therefore we have $I_B \in AI_C$ or $I_C \in AI_B$. Wlog assume the former case. Let B' be the intersection of AC with the line passing through BI_B. Similarly, there exists a segment CC' with direction D_c and $C' \in BB'$. Since AI_BB' forms a right triangle, $\angle(B'B, B'C)$ is obtuse, therefore the normal to CC' passing through B' intersects CC' at some point, denoted $J_{B'}$. Let D be the intersection of BC with the line passing through $B'J_{B'}$. Since $CJ_{B'}D$ forms a right triangle, $\angle(DB', DB)$ is obtuse, so the normal to BB' passing through D intersects BB' at some point $D' \in BB'$. This completes the partitioning of ABC into six right triangles $\{AI_BB, AI_BB', BD'D, B'D'D, DJ_{B'}C, B'J_{B'}C\}$, each with one (non-hypothenuse) side parallel to a direction in Δ. ∎

This example illustrates the *buffering* effect of "filtering search" [C1], whereby reporting many points seems relatively as easy as reporting only one. The notion of efficiency for retrieval problems is traditionally captured by complexities of the form $(nPOLYLOG(n), POLYLOG(n))$. Whether triangular search is in this class, i.e. whether the quadratic complexity of the underlying criticality set can be overcome, remains open. What we have shown is that we can restrict future investigation to the case where T is arbitrarily close in shape to a segment.

2. Frame-to-Frame Coherence

Hidden-line elimination problems are many, varied, and often quite difficult. One reason for the latter is that once again the input is generally not amenable to searching, so one has to expand the input into a larger set where critical objects appear explicitly. The problem of maintaining frame-to-frame coherence in *Flatland* (i.e. in E^2) has been studied extensively by Edelsbrunner et al [EOW]. The difficulty of this restricted case of hidden-line elimination is well encapsulated in the following set of problems. Let $S = \{s_1, \ldots, s_n\}$ be a set of n segments in the plane and let C be a circle enclosing S. Let $\text{int}(A)$ denote the interior of figure A. We assume that $[\forall i, j; 1 \leq i \leq j \leq n \mid \text{int}(s_i) \bigcap \text{int}(s_j) = \emptyset]$. This will allow S, for example, to be a set of simple polygons. Let (Ox, Oy) be an orthogonal system of coordinates with O at the center of C. For each θ $(0 \leq \theta < 2\pi)$, let $v(\theta)$ be the point of C s.t. $\angle(Ox, Ov(\theta)) = \theta$ and let $T(\theta)$ be the tangent to C at $v(\theta)$. Define $V(\theta)$ to be the parallel view of S from $T(\theta)$, i.e. the function $p \in T(\theta) \mapsto f(p) \in \{0, 1, \ldots, n\}$, where $s_{f(p)}$ contributes the nearest intersection to p between S and the normal to $T(\theta)$ at p (Fig.2). For consistency, $f(p) = 0$ if the intersection is empty. Also, let L be an oriented line and $H(L)$ be the index of the first segment of S to be intersected by L; once again, $H(L) = 0$ if no intersection.

$P1$. For any θ; $0 \leq \theta < 2\pi$, compute $V(\theta)$.

$P2$. For any θ; $0 \leq \theta < 2\pi$, and any segment $q \subseteq T(\theta)$, compute the restriction of $V(\theta)$ to the domain q.

$P3$. For any oriented line L, compute $H(L)$.

$P4$. Is it possible to see through S? More formally put, is there at least one value of θ s.t. 0 appears at least three times in the sequence $V(\theta)$?

$P5$. Is there a line L that has exactly k segments of S on one side and $n-k$ on the other side (s.t. $S \bigcap L = \emptyset$).

Theorems 2,3 improve the best previous results by factors of n space (1,2), $\log n$ space (3), and $\log n$ time (4,5).

Theorem 2. Problems 1,2,3 can be solved in $(n^2, \log n)$.

Proof: Let $\Omega = C \bigcap T(\theta) = v(\theta)$. We consider the local reference system $(\Omega\theta, \Omega z)$ where (θ, z) are the coordinates of the point p of $T(\theta)$ at a distance z from Ω. We resolve the ambiguity of this definition by stipulating that z is positive iff O, Ω, p form a left turn. The functions $V(\theta)$ are easily seen to subdivide the θz plane into $O(n^2)$ regions delimited by curves of the form $z = r\sin(\alpha - \theta)$ — Fig.3 depicts the subdivision corresponding to Fig.2. Each region is assigned an index i indicating that any ray of light emanating from a point in this region towards $\text{int}(C)$ in a direction normal to $T(\theta)$ will first hit s_i. To construct the subdivision, denoted $\mathcal{T}$, can be done using a standard sweep-line technique. One will first sort the slopes of all inter-endpoint segments and compute $V(0)$. The precomputed slopes are taken as event-points of a sweep-line process updating $V(\theta)$ as θ grows from 0 to 2π — details omitted. The computation is easily done in $O(n^2 \log n)$ time and $O(n^2)$ space. The subdivision $\mathcal{T}$ is represented by a DCEL [MP] or quad-edge structure [GS], with each edge (vertex) pointing to the one (two) segment(s) of S from which they originate. The interesting observation is that since each edge is monotone along the θ-axis, 1) $\mathcal{T}$ can be preprocessed for efficient point location as described in [EGS], 2) one can build up a *hive-graph* on the set of edges [C1]. Recall that the latter construction will require here $O(n^2 \log n)$ time, and will allow us to compute all the intersections between $\mathcal{T}$ and a query segment parallel to the z-axis in $(n^2, \log n)$. With this preprocessing, solving Problems 1–3 is easy: 1,2) – compute $\{\Theta - \theta = 0\} \bigcap \mathcal{T}$ (1) or $q \bigcap \mathcal{T}$ (2), using the hive-graph; 3) – compute $T(\theta)$, the first normal to L tangent to C traversed by L. Next locate $(\theta, z) = L \bigcap T(\theta)$ in $\mathcal{T}$ and report the corresponding index. ∎

Observation 1. Let $I = \{I_1, \ldots, I_n\}$ be a list of n closed intervals on a circle C, with all the endpoints stored in sorted order in a circular list. Let J be the set of intervals formed by $C - \left(\bigcup_{1 \le i \le n} I_i\right)$. If J is not empty, let $J_1, \ldots, J_p$ be its elements in sorted order around C, and let $K = C - \left(\bigcup_{1 \le i \le p} J_i\right)$. K consists of p intervals $K_1, \ldots, K_p$, with the convention that K_i immediately follows J_i in clockwise order. Note that $\bigcup_{1 \le i \le p}(J_i \cup K_i) = C$. Let k_i denote the number of intervals of I that make up K_i. In $O(n)$ time it is possible to determine whether J is empty, and if not, compute the sorted list $\{(J_1, k_1), \ldots, (J_p, k_p)\}$.

Proof: Choose a point O on C and, in $O(n)$ time, use this point to split in two every interval of I overlapping O. This allows us to order each endpoint v according to the length of the clockwise arc Ov. In $O(n)$ time, we express the augmented set of intervals as a list of the form $\{[a_1, b_1], \ldots, [a_m, b_m]\}$ $(n \le m \le 2n)$ with $b_1 \le \ldots \le b_m$. We compute J by considering each interval $[a_i, b_i]$ in turn. During the computation, the list of intervals currently in J is stored in a stack. For each $i = 1, \ldots, m$, process $[a_i, b_i]$ by popping off the stack all the intervals covered by $[a_i, b_i]$; update top of stack and iterate. To compute $\{k_1, \ldots, k_p\}$ is trivially done in $O(n)$ time by simulating a merge between the endpoints of J and those of I (taken here before splitting).

∎

Theorem 3. Problems 4,5 can be solved in $O(n^2)$ time.

Proof: Let $R(v, \theta)$ be the ray emanating from v with slope θ; $(0 \le \theta < 2\pi)$. For any endpoint v in S, define $R(v) = \{\theta \in [0, 2\pi) \mid \forall s \in S, s \bigcap R(v, \theta) = \emptyset\}$. $R(v)$ consists of a (possibly empty) set of open intervals. This set corresponds to all the maximal wedges centered at v from which v can see to infinity. Suppose that $R(v)$ is not empty; then let $R(v) = \{J_1, \ldots, J_p\}$ in angular order around v, and let k_i be the number of segments of S in the wedge between J_i and J_{i+1} (mod p). Let $L(v)$ be the list of endpoints in S (except v) sorted angularly around v. If $L(v)$ is given for each endpoint v in S, Observation 1 can be trivially adapted to compute the list $\{(J_1, k_1), \ldots, (J_p, k_p)\}$ in $O(n)$ time. By *sliding* the interval $[\theta, \theta + 2\pi]$ through $R(v)$ from $\theta = 0$ to $\theta = \pi$, it is then easy in $O(n)$ time to compute 1) each line L passing through v not strictly intersecting any segment in S; 2) how many segments lie on each side of L. This immediately leads to $O(n^2)$ solutions to Problems 4,5, provided that the set of $L(v)$'s can be computed in the same amount of time. To see this, transform the $2n$ endpoints in S into dual lines. A point $p : (a, b)$ is mapped to the line $D_p : Y = aX + b$, and similarly any line $L : Y = kX + d$ is mapped to the point $D_L : (-k, d)$. Observe that two points are mapped to two parallel lines if and only if these points have the same x-coordinates. We can assume that no endpoints in S share the same x-coordinates, which may possibly entail rotating the axes by a small angle, at a cost of $O(n \log n)$ time. The dual set of the $2n$ endpoints forms a line arrangement, whose underlying planar graph can be computed in $O(n^2)$ time [CGL,EOS]. The key observation now is that for each endpoint v, the line D_v intersects exactly $2n - 1$ lines, which can be retrieved in their intersecting order w.r.t. D_v in $O(n)$ time. But the x-coordinate of $D_v \bigcap D_w$ is precisely $-$slope(vw), therefore $L(v)$ can be computed in $O(n)$ time, after $O(n^2)$ preprocessing. This completes the proof. ∎

3. Computing Algebraic Predicates

Given n lines and n points in the plane, does any line pass through any of the points (problem posed by Hopcroft)? Given n points in E^3, are any 5 co-spherical or do any 17 have their interdistances adding up to 1? More generally, given n geometric objects in arbitrary dimensions, do any k interact in a pre-specified manner? This general class of questions can be easily formalized. Let $S = \{V_1, \ldots, V_k\}$ be a collection of k sets of the form $V_i = \{v_1^{(i)}, \ldots, v_{p_i}^{(i)}\}$. Each $v_j^{(i)}$ is a vector $\in \Re^{c_i}$ of the form $v_j^{(i)} = (x_{j,1}^{(i)} \ldots, x_{j,c_i}^{(i)})$. S is the *input* and the quantity $n = \sum_{1 \le i \le k} p_i c_i$ is the *input size.* Let $Q_d(y_{1,1}, \ldots, y_{1,c_1}, y_{2,1}, \ldots, y_{2,c_2}, \ldots, y_{k,1}, \ldots, y_{k,c_k})$ be a rational $\lambda-$variate polynomial of degree d; $\lambda = \sum_{1 \le i \le k} c_i$. If $w_i = (y_{i,1}, \ldots, y_{i,c_i})$, we use the abbreviation $Q_d(w_1, \ldots, w_k)$. In the following, the quantities $d, k, c_1, \ldots, c_k$ are considered to be constants.

Problem P: Does there exist a k-tuple $(j_1, \ldots, j_k) \in \Pi_{1 \le i \le k}\{1, \ldots, p_i\}$ s.t. $Q_d(v_{j_1}^{(1)}, \ldots, v_{j_k}^{(k)}) = 0$?

To begin with, using elementary elimination theory, one can easily rephrase the problems previously mentioned as special cases of Problem P. There is a trivial $O(n^k)$ solution involving the testing of all possible k-tuples. The following result asserts that it is always possible to do (a little) better.

Theorem 4. Let $c = \min(c_1, \ldots, c_k)$. Problem P can be solved in $O(n^{k-\frac{1}{2^{c+7}}})$ time.

Proof: We generalize the batching strategy introduced by Yao in his work on higher dimensional MST [Y]. Wlog assume that $c = c_k$. Consider the m c_k-variate polynomials of the form $Q_d(v_{j_1}^{(1)}, \ldots, v_{j_{k-1}}^{(k-1)}, z_1, \ldots, z_{c_k})$, where $z_1, \ldots, z_{c_k}$ are the variables and $(j_1, \ldots, j_{k-1}) \in \Pi_{1 \le i \le k-1}\{1, \ldots, p_i\}$; note that $m \le n^{k-1}$. Divide up this set of polynomials into subsets of size α, denoted $W_1, \ldots, W_t$ $(t < 1 + \frac{1}{\alpha}\Pi_{1 \le i \le k-1} p_i)$. In [C2] a method is described for preprocessing a set of polynomials so that determining whether a query vector is a zero of one of them can be done in logarithmic time. More precisely, let $F = \{P_1, \ldots, P_\alpha\}$ be a family of α fixed-degree r-variate polynomials with rational coefficients and let $S = \{x \in E^r \mid \Pi_{1 \le i \le \alpha} P_i(x) \ne 0\}$. In $O(\alpha^{2^{r+6}})$ time and space, it is possible to compute a set of algebraic points, one in each connected region of S, as well as set up a data structure for computing the predicate $[\exists i\ (1 \le i \le \alpha) \mid P_i(q) = 0]$, for any $q \in E^r$. In $O(2^r \log \alpha)$ time, the algorithm will return an index i such that $P_i(q) = 0$ if such an index is to be found, otherwise it will return the algebraic point associated with the unique region of S that contains q. Applying this method to each set $W_1, \ldots, W_t$ requires $O(t\alpha^{2^{c+6}}) = O(n^{k-1}\alpha^{2^{c+6}-1})$ preprocessing time. On the other hand, this will allow us to test the p_k vectors of V_k in $O(p_k t \log n) = O(\frac{n^k \log n}{\alpha})$ overall query time. Setting $\alpha = (n \log n)^{1/2^{c+6}}$ leads to a time complexity of $O(n^{k-\frac{1}{2^{c+6}}}(\log n)^{1-\frac{1}{2^{c+6}}})$, which for the sake of simplicity we conservatively estimate as $O(n^{k-\frac{1}{2^{c+7}}})$ time. ∎

4. Conclusions

This work attempts to place the notion of *criticality* in implicit set representations at a more central location. By doing so we are able to improve on a number of previous complexity results. For the most part, our improvements are algorithmic in nature; we feel that more geometric insights will be needed to decide whether the criticality of point sets and their duals, line arrangements, constitutes an unsurmountable obstacle to computational efficiency. For example, is quadratic space required to answer the point-on-query-line question in polylogarithmic time?

REFERENCES

[C1] Chazelle, B. *Filtering search: A new approach to query-answering*, Proc. 24th Ann. Symp. Found. Comp. Sci. (1983), pp. 122–132.

[C2] Chazelle, B. *Fast searching in a real algebraic manifold with applications to geometric complexity*, Brown Tech. Rep. CS–84–13, June 1984.

[CG] Chazelle, B., Guibas, L.J. *Fractional cascading: A data structuring technique with geometric applications*, to appear in ICALP'85.

[CGL] Chazelle, B., Guibas, L.J., Lee, D.T. *The power of duality*, Proc. 24th Ann. Symp. Found. Comp. Sci. (1983), pp. 217–225.

[CY] Cole, R., Yap, C.K. *Geometric retrieval problems*, Proc. 24th Ann. Symp. Found. Comp. Sci. (1983), pp. 112–121.

[EGS] Edelsbrunner, H., Guibas, L., Stolfi, J. *Optimal point location in a monotone subdivision*, to appear in SIAM J. Comp.

[EKM] Edelsbrunner, H., Kirkpatrick, D.G., Maurer, H.A. *Polygonal intersection search*, Inform. Process. Lett. 14 (1982), pp. 74–79.

[EOS] Edelsbrunner, H., O'Rourke, J., Seidel, R. *Constructing arrangements of lines and hyperplanes with applications*, Proc. 24th Ann. Symp. Found. Comp. Sci. (1983), pp. 83–91.

[EOW] Edelsbrunner, H., Overmars, M.H., Wood, D. *Graphics in Flatland: a case study*, Adv. in Comput. Res., Vol. 1, ed. F.P. Preparata, JAI Press, (1983), pp. 35–59.

[EW] Edelsbrunner, H., Welzl, E. *Halfplanar range search in linear space and $O(n^{0.695})$ query time*, Rep. F111, Inst. Inform. Proc., Tech. Univ. Graz, Austria, 1983.

[GS] Guibas, L.J., Stolfi, J. *Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams*, Proc. 15th Ann. SIGACT Symp. (1983), pp. 221–234.

[MP] Muller, D.E., Preparata, F.P. *Finding the intersection of two convex polyhedra*, Theoret. Comput. Sci., 7 (1978), pp. 217–236.

[W] Willard, D.E. *Polygon retrieval*, SIAM J. Comp., 11 (1982), pp. 149–165.

[Y] Yao, A.C. *On constructing minimum spanning tree in k-dimensional space and related problems*, SIAM J. Comput. 11 (4), 1982, pp. 721–736.

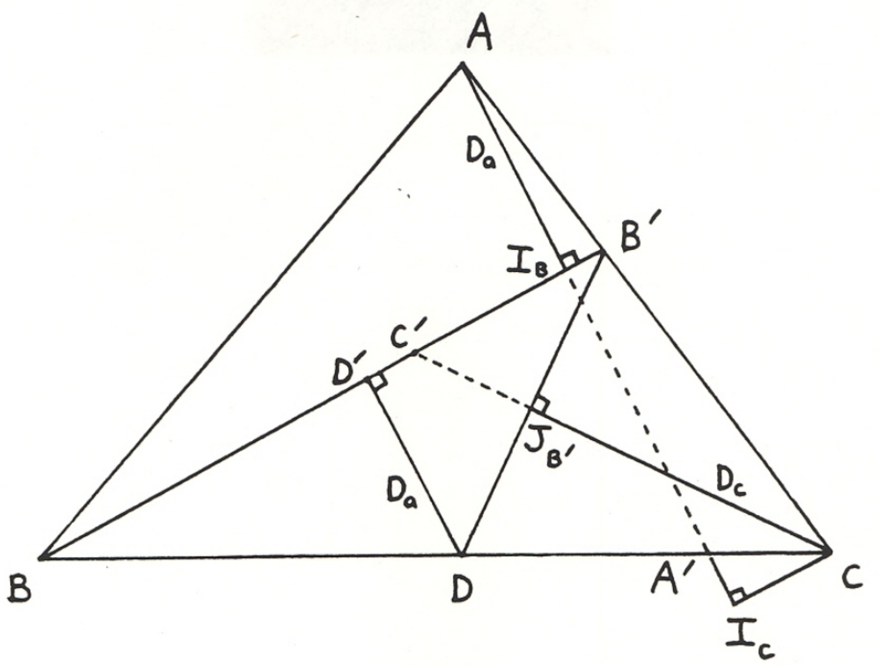

Figure 1

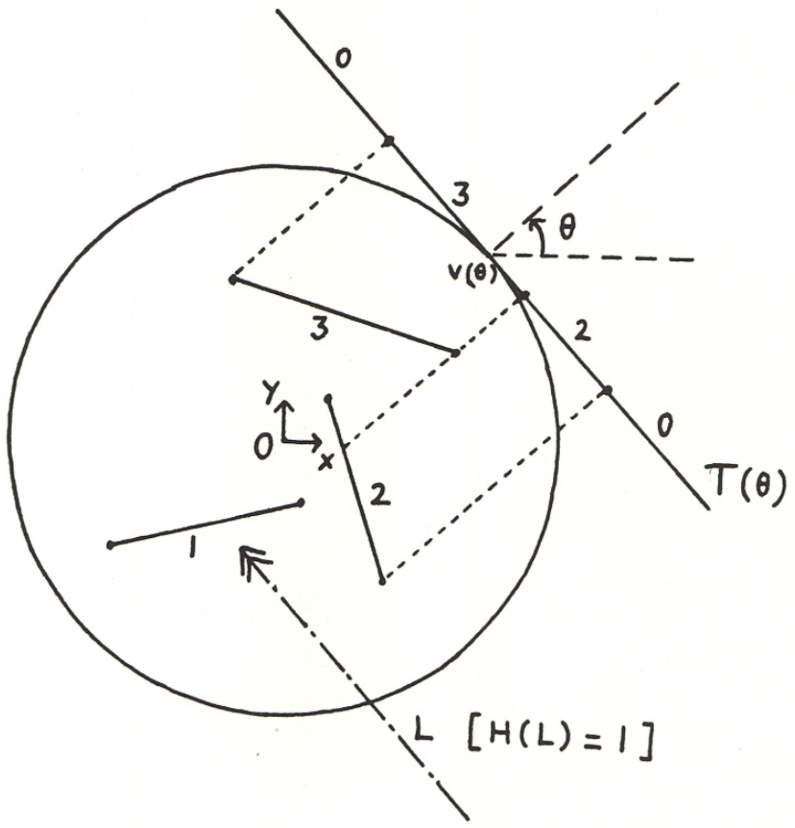

Figure 2

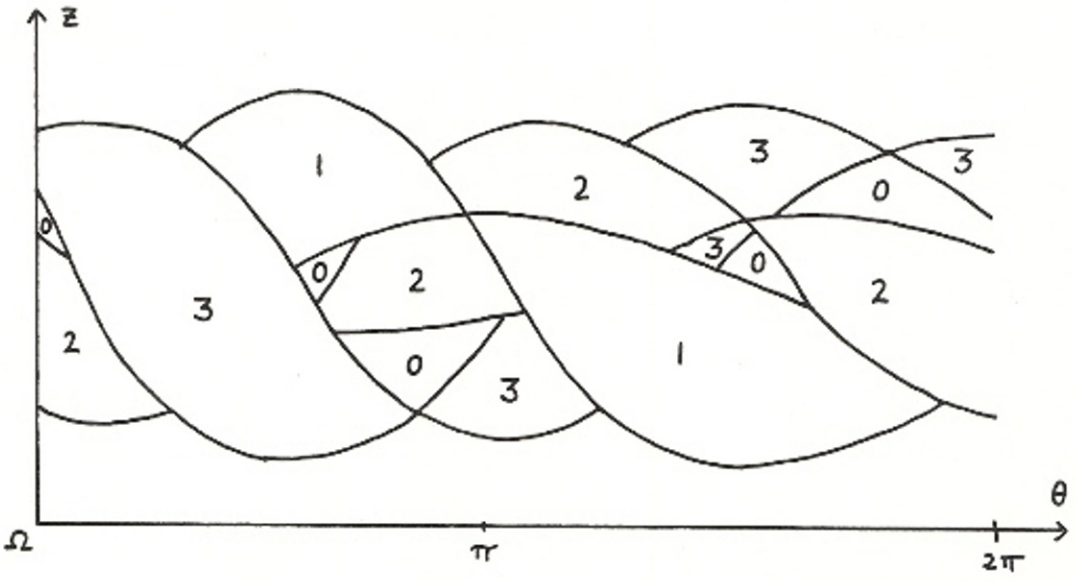

Figure 3