On the visibility locations for continuous curves

Abstract

The problem of determining visibility locations (VLs) on/inside a domain bounded by a planar C^1-continuous curve (without vertices), such that entire domain is covered, is discussed in this paper. The curved boundary has been used without being approximated into lines or polygons. Initially, a few observations regarding the VLs for a curved boundary have been made. It is proposed that the set of VLs required to cover the domain be placed in a manner that the VLs and the lines connecting them form a spanning tree. Along with other observations, an algorithm has been provided which gives a near optimal number of VLs. The obtained number of VLs is then compared with a visibility disjoint set, called as witness points, to obtain a measure of the 'nearness' of the number of VLs to the optimum. The experiments on different curved shapes illustrate that the algorithm captures the optimal solution for many shapes and near-optimal for most others.

Keywords: Curved boundary, Continuous curves, Visibility Locations, Guard placements, Sensor location, Splinegon, Covering problem, Camera placement

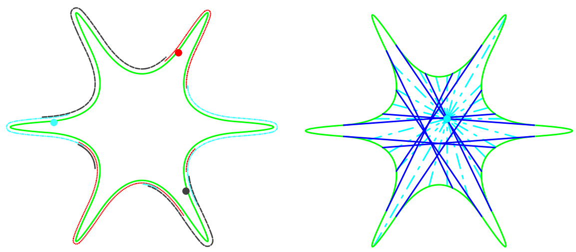
1. Introduction

The problem of identifying regions of visibility within a domain (or from outside of it) has been useful in many applications such as mold design for manufacturing, inspection of models, shortest path identification, placing guards to cover an art gallery, sensor location, robot motion planning etc. In the case of mold design, the problem is posed as 'whether the model is a two-piece, given a set of viewing directions' [1]. Alternatively, given a model, the problem is to identify optimal parting directions that reduce the number of mold pieces [2, 3].

Viewpoint selection that covers the entire object has been used in inspection. Clearly, creating an optimal set of viewpoints (or visibility locations (VLs)) will then reduce the overall cost of inspection (see [4] for a detailed survey on this topic). In the case of shortest path identification [5], the visibility graph has been a very popular construction, which can be computed using tangents [6].

Sensor location also depends on the visibility of a feature, apart from several other factors [7]. Other applications including security, computer graphics (hidden surface removal) etc. also come under the realm of visibility region identification. For further details on the applications, see [8].

The problem of visibility locations has usually considered domain bound by a polygon (a closed shape with well-defined vertices and edges as straight lines which do not intersect except at their vertices), at times with holes, typically solved by computational geometers and termed as art-gallery problem (see [9]) and occasionally polyhedra [10, 11]. However, the problem of VLs rarely considers complex objects such as curved ones. Recently, this problem for curved polygons has been addressed in [12, 13] by replacing straight edges with curves which are either piecewise convex or piecewise concave, but not a mixture of both. In the current available literature for curved polygons, vertices are well defined.



(a) Three boundary guards (VLs). (b) One interior guard (VL).

Figure 1: Boundary guards [8] vs. guard at the interior for a star-shaped domain (guards shown as dots).

Determining optimal visibility locations for a single closed continuous curved boundary (i.e without explicit notion of vertices), particularly when the locations can be interior to the domain has not been addressed, to the best of our knowledge. A conservative estimate on the number of VLs when the locations can be on the walls of the curved boundary has been provided in [8] using a visibility chart. The algorithm in [8] requires a set of candidates as input, from which either a set of VLs may be obtained or the algorithm results in failure. To aid practical solutions, they also discretize the visibility chart. Other works which discretize for practical solutions include generalised discrete framework for visibility problems in [14] and geometric multi-covering [15].

Problem Statement, approach and the obtained results

More often than not, in practice, VLs (hereafter, termed as *guards*, for ease and clarity of explanation) are required to be placed not just on the boundary but also interior to it. For example, for a star-shaped curved polygon, as opposed to three guards on the boundary (Figure 1(a)), a single guard interior to the domain can cover it (as in Figure 1(b)). In practice, it would

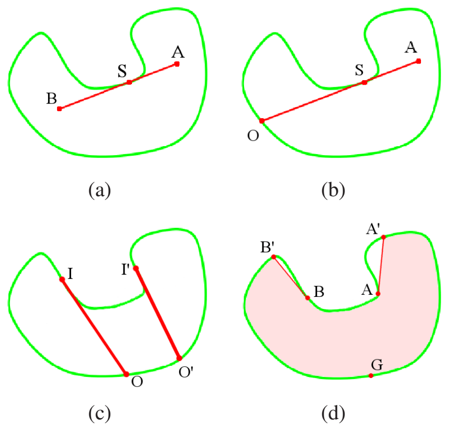


Figure 2: (a) An internal tangent AB, and (b) its silhouette and occlusion points S and O respectively. (c) Inflection points I and I', and their IPTs IO and $I'O'$. (d) The visibility region $\mathcal{V}(G)$ of a guard G shown in red color.

be useful if the guards are allowed to be placed interior/on the domain (and not just on its boundary). Hence, given a planar domain bounded by a smooth (i.e. without C^1 discontinuities), parametric, non-convex closed simply-connected curve $C(t)$, this paper aims to find the near-optimal number of guards that cover the entire domain. To the best of the knowledge of the authors, this is perhaps the first work aimed in this direction. Vertices are not explicitly defined in the curved boundary considered in this paper, and hence it is different from the curved domains considered in [12, 13]. Also, no discretization approach has been employed like in [8], though we use a rule-based approach like the one used in [16]. However, the rules have been arrived upon based on our observations. The approach presented is heuristic-based and greedy in nature. It adds one guard at a time. Moreover, we employ a first order approach to solve this problem, typically employed in hidden Markov models. It can also be noted that there can be many measures to come up with a 'good' guard from a candidate set (as can be seen in [16]) and our approach is based on internal tangents as it is related to the visibility in the case of a curved boundary.

We also have proposed an algorithm to compute 'witness points', a technique introduced in [16], based on which the optimality of the solution obtained has been conjectured to be no worse than twice the actual optimal number of guards. In practice, based on the experiments conducted, the proposed approach returns the optimal solution for many of the tested curves.

2. Preliminaries

Let the boundary of a domain $\mathcal{D}$ be bounded by a parametric closed curve $C(t)$ without C^1 discontinuities. Let the exterior of the curve be denoted by $\mathcal{D}^c$.

2.1. Definitions

Definition 1. *A point on a curve at which the curvature changes sign is called an* inflection point.

Definition 2. *A point on a curve is concave if its center of curvature and outward normal at that point are in the same direction, otherwise the point is termed convex (S in Figure 2(a) is concave, whereas O in Figure 2(b) is convex).*

Definition 3. *An internal tangent (denoted as IntT) is a line segment completely lying to the interior/on the curve (no point of the line segment lies exterior to the curve) which is a tangent to at least one point on the curve (e.g., the line segment AB in Figure 2(a) or AO in Figure 2(b)).*

Definition 4. *The point at which an internal tangent touches a curve tangentially is called its silhouette point (henceforth denoted by S) [8]. S in Figure 2(a) is the silhouette point of the IntT AB.*

Definition 5. *If an internal tangent has another point lying on the curve (apart from its silhouette point), then such a point is called an occlusion point (O in Figure 2(b)) [8], and is henceforth denoted by O.*

Definition 6. *An internal tangent starting at an inflection point is called inflection point tangent (IPT). Its starting point coincides with its silhouette point (Figure 2(c)).*

Definition 7. *A point $P \in \mathcal{D}$ is considered visible to another point $Q \in \mathcal{D}$, if for all points $x \in \overline{PQ}$, $x \cap \mathcal{D}^c = \phi$, i.e. no point on the line segment PQ lies completely exterior to the boundary of $\mathcal{D}$. Grazing contact is allowed i.e. the line segment can touch the boundary (typically tangentially).*

For example, in the Figure 2(b), O is considered visible to A even though the line segment OA has a grazing contact at S.

Definition 8. *Let $\mathcal{V}(G) = \{x \mid x \in \mathcal{D}$ and x is visible to $G\}$ be the set of points forming the visibility region of the point G (the red area in Figure 2(d) indicates the visibility region of G). A set W, consisting of points on or inside $C(t)$ are termed as witness points [16] if visibility regions in the set are pairwise disjoint, i.e., $\forall_{q,r \in W} \mathcal{V}(q) \cap \mathcal{V}(r) = \phi$.*

2.2. Observations on the visibility of a guard

The motivation for our observations comes from the fact that, unlike a polygonal boundary, a C^1 continuous curved boundary does not have explicitly defined vertices. A guard is assumed to be represented as a point which can see in every direction (i.e. has a 360° range of visibility). A set of guards is said to cover the domain if every point in the domain is visible to some guard [9]. Also, a guard cannot see through the curved boundary (i.e. the boundary is assumed to be opaque), and can either lie on or interior to it. An *IntT* is assumed to have at most one silhouette point. The following propositions/rules are crucial to develop an algorithm to determine the guards in a domain bound by a curved boundary. In the subsequent sections, when we draw an internal tangent from a point, we exclude the ones that are coincident with IPTs.

Proposition 1. *Let $p \in \mathcal{D} \backslash C(t)$ be a point interior to the domain. $\mathcal{V}(p) \neq \mathcal{D}$ if and only if $\exists$ an internal tangent which can be drawn from p.*

Proof. Refer [17] (Section 2.1) □

Our algorithm only uses the forward direction of this double implication, which allows us to draw internal tangents from points which cannot guard the entire domain.

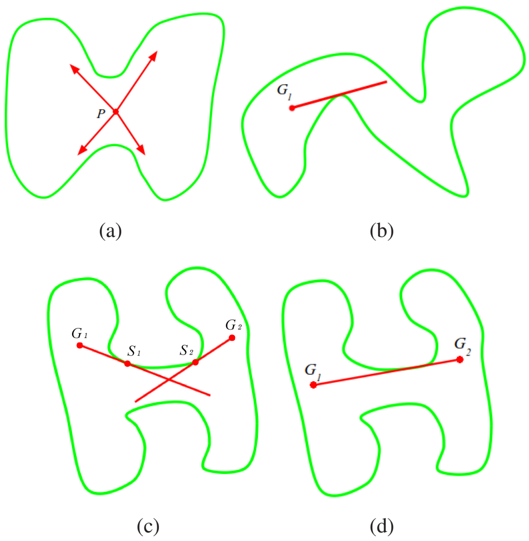


Figure 3: (a) A guard located at P has no internal tangents and covers the entire domain. (b) Even though only one IntT can be drawn from G_1, one or more guards are still required to cover the remaining region. (c) Unguarded scenario for a domain requiring two guards (d) Full coverage obtained by moving them so that one guard lies on the IntT of the other.

For example, in the Figure 2(b), a guard placed at A will not see the curve portion counterclockwise from S to O due to the presence of an internal tangent. However, in the Figure 3(a), a guard located at point P can see the entire region as no internal tangent can be drawn from the point to the curve. For an intricate region within a curve (Figure 3(b)), only one internal tangent can be drawn from the guard G_1. Nevertheless, G_1 is still required to cover this region.

Further it can be intuitively observed that, for a curved domain requiring only two guards to cover it, the guards can be placed in such a way that one guard can see the other and the line segment connecting them is tangential to the curve. For example, let us instead place two guards which do not see each other (Figure 3(c)), i.e. the line joining the guards intersects the boundary of the curve. By Proposition 1, there exists at least one internal tangent which can be drawn from each of them. Let the corresponding silhouette points be S_1 and S_2. The region on $C(t)$ lying between S_1 and S_2, if seen clockwise, is left unguarded. Now if these guards are moved so that their internal tangents become collinear (i.e. by placing one guard on the internal tangent of the other like in Figure 3(d)), S_1 and S_2 will coincide. As a consequence, there is no unguarded region between S_1 and S_2 like in the previous case. Based on this intuition, we have the following rule:

Rule 1. *Let $G_1 \in \mathcal{D}$ be a guard which does not see the entire domain. Another guard can be obtained by drawing an internal tangent from G_1 and choosing a point on the internal tangent beyond its silhouette point.*

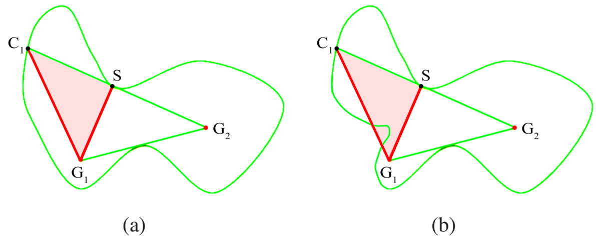


Figure 4: A figure showing two example curved domains. Guards G_1 and G_2 have been placed, and a new candidate guard C_1 lying on the IntT drawn from G_2 has been obtained. (a) C_1 and S form an empty triangle with G_1 (b) $\Delta C_1 S G_1$ intersects the curved boundary.

Rule 1 implies that G_2 in Figure 3(d), drawn as internal tangent from G_1, is able to see the silhouette point of G_1 and some region beyond it. Rule 1 can be applied repeatedly to find new guards to cover the domain. Section 4.1 later describes how to place new guards beyond the corresponding silhouette points. The algorithm in this paper involves the identification of one guard as per this rule in each iteration, and hence this guard is termed as the *subsequent* guard.

While Rule 1 enables us to find subsequent guards, this rule alone is not sufficient to ensure termination of the algorithm in all cases. This is because, in curves which require more than one guard, every interior point has an internal tangent. Hence, regardless of where we place our first guard, we can keep finding new internal tangents (and thus new guards on them), leading to a doubling back into already guarded regions. Hence, we add a further rule which enables us to cut down on possibly redundant candidates for the subsequent guard. This is based on the intuition that a subsequent guard placed on an internal tangent of an existing guard G may not be required when the region beyond the intT's silhouette point is already visible to another existing guard. For example, in Figure 4(a), let G_1 and G_2 be the current set of guards. Let G_2C_1 be an internal tangent drawn from G_2 and C_1 be the subsequent guard obtained by Rule 1. Since $\overline{SC_1}$ is visible to an existing guard G_1, the new candidate C_1 is not required in this case. So we propose the rule:

Rule 2. *Empty triangle property: An internal tangent drawn from an existing guard is considered for determining a subsequent guard only if the triangles formed by its silhouette point S and occlusion point O with each of the remaining guards are not completely interior to the domain, i.e. given a set of existing guards $\mathcal{G}$ and an intT drawn from G_1, no subsequent guards will be placed on intT if $\exists G_i \in \mathcal{G} \backslash G_1$ such that $\Delta(SOG_i) \subset \mathcal{D}$.*

Such triangles which lie completely inside $\mathcal{D}$ are referred to as *empty triangles*. For example, the internal tangent G_2C_1 forms an empty triangle $\Delta(SC_1G_1)$ with G_1 in Fig. 4(a) and hence no guard be placed on G_2C_1. It can be seen that G_1 and G_2 are sufficient to cover the domain. On the other hand, in Fig. 4(b), where $\Delta(SC_1G_1)$ is not an empty triange, a guard at C_1 is needed.

Definition 9. *An internal tangent from a guard is called a valid internal tangent, and needs to be considered for determining*

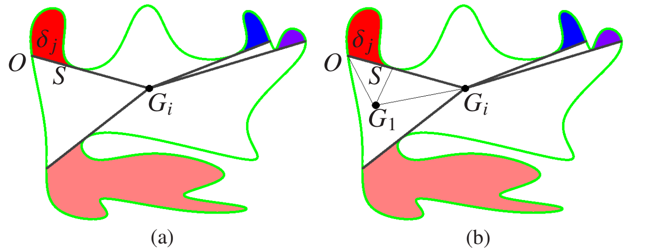


Figure 5: Proof of Proposition 2 - repeated application of Rules 1 and 2 can cover the entire domain.

subsequent guards, only if it satisfies the empty triangle property (Rule 2).

Proposition 2. *Given a domain $\mathcal{D}$ and an initial set of guards $\mathcal{G}$, repeated application of Rule 1 and Rule 2 will eventually give a new set of guards $\mathcal{G}'$ which can cover the entire domain.*

Proof. Assume, on the contrary, that we have a set of guards $\mathcal{G}'$ which do not fully guard $\mathcal{D}$ and that no subsequent guards can be added by Rule 1 because of Rule 2 (i.e. none of the existing guards have valid internal tangents). Let $\{\delta_1, \delta_2, \ldots\}$ be the set of disjoint components in $\mathcal{D}$ that are left unguarded by $\mathcal{G}'$ (i.e. the union of shaded regions in Figure 5(a)). The guarded portion of $\mathcal{D}$ adjacent to any δ_j must be guarded by some guard $G_i \in \mathcal{G}'$ such that a part of the internal tangent T drawn from G_i to $C(t)$ forms the boundary of δ_j (the segment SO in Figure 5(a)). Since δ_j is unguarded, T must have been rejected as a valid internal tangent due to Rule 2, and hence it forms an empty triangle with some guard. Let this guard be G_1, i.e. we have $S, O \in \mathcal{V}(G_1)$. Because the domain is simply connected, the segment $\overline{SO} \in \mathcal{V}(G_1)$. Since G_1 can see any point on $\overline{SO}$, it can also see some point P lying on SO which is also a part of the boundary of δ_j. So $\exists P$ not lying on $C(t)$ such that $\overline{G_1 P}$ can be extended to cross over into δ_j. This means that a point interior to δ_j is visible to the guard G_1, which contradicts the assumption that δ_j is, in fact, unguarded. □

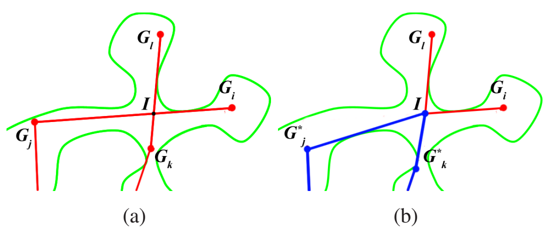


Figure 6: Corollary 1: (a) A configuration of guards such that internal tangents intersect at I, (b) I is added to the guard set, and G_j and G_k are recomputed via internal tangents from I to give a new set of guards joined by non-intersecting internal tangents.

Corollary 1. *There exists a set of guards covering the entire domain such that every guard is connected to at least one other guard by a valid internal tangent, and the internal tangents are non-intersecting except at their end points.*

Proof. Let us start with a set of guards which are connected by internal tangents such that some tangent $\overline{G_iG_j}$ intersects another tangent, say $\overline{G_kG_l}$ at the point I. Since we do not want intersections, we create a new guard at I. However, doing this means that I is not connected via an internal tangent to one of the guards from G_i and G_j, and one from G_k and G_l (segments IG_j and IG_k in Figure 6(a) are no longer internal tangents). More precisely, if the silhouette point of $\overline{G_iG_j}$ lies on the segment G_iI, G_iI remains an internal tangent and IG_j is no longer one, and vice versa.

This is a problem because we require the line segments joining all our guards to be internal tangents. To remedy this, we can delete such guards whose segments with I no longer form internal tangents, and their subgraphs, from the guard set. This will leave some part of the domain uncovered. Now, by Proposition 2, we have valid internal tangents which we can construct from I to place new guards (G_j^* and G_k^* in Figure 6(b)) until the domain is covered again. The empty triangle property is not broken at any point doing this construction and now have a set of guards covering the entire domain, joined by non-intersecting internal tangents. □

Proposition 3. *Let $\mathfrak{G}$ denote a planar graph with its vertices as guards and the edges as the internal tangents between the guards as obtained by Rule 1 and Rule 2. There exists at least one set of guards forming the vertex set of $\mathfrak{G}$ that can cover the entire domain such that $\mathfrak{G}$ forms a spanning tree [18].*

Proof. Once an initial guard is located (see Sections 4.1 and 4.2), by Rule 1, we can draw internal tangents from the existing guard, where a subsequent guard is placed on one of the internal tangents. This process can be repeated till the entire domain is covered (when there are some portions left uncovered, then there exists a valid IntT (Proposition 1 and Proposition 2) using which subsequent guards can be identified). The graph $\mathfrak{G}$ will not have a cycle because the only way of getting a cycle is by placing a subsequent guard at the same location as an existing guard, and that would contradict Rule 2 because both the silhouette and occlusion point of the new guard will be visible to the existing guard. Further, every edge in $\mathfrak{G}$ is disjoint except at its endpoints by Corollary 1. Moreover, the graph is connected as every guard has an edge to at least one another guard via an internal tangent. Hence the Proposition. □

Proposition 4. *Let $\mathcal{G}$ be any set of guards that can cover the entire domain $\mathcal{D}$. Let W be a set of visibility-disjoint witness points (refer Definition 8) in $\mathcal{D}$. For any $\mathcal{G}$ and W, we will always have $|W| \leq |\mathcal{G}|$.*

Proof. We will prove this by contradiction. Assume that $\exists W \in \mathcal{D}$ such that $|W| > |\mathcal{G}|$. Since $\mathcal{G}$ can see the entire domain $\mathcal{D}$, we have $\forall W_i \in W, \exists G_j \in \mathcal{G}$ such that $W_i \in \mathcal{V}(G_j)$. But since $|W| > |\mathcal{G}|$, there exists at least one guard which can see two or more witness points because of the pigeonhole principle, i.e. $\exists G_k$ such that $\mathcal{V}(G_k)$ contains both W_m and W_n for some m and n. However, this also means that $G_k \in \mathcal{V}(W_m)$ and $G_k \in \mathcal{V}(W_n)$, which means that $\mathcal{V}(W_m) \cap \mathcal{V}(W_n) = G_k \neq \phi$, which means that

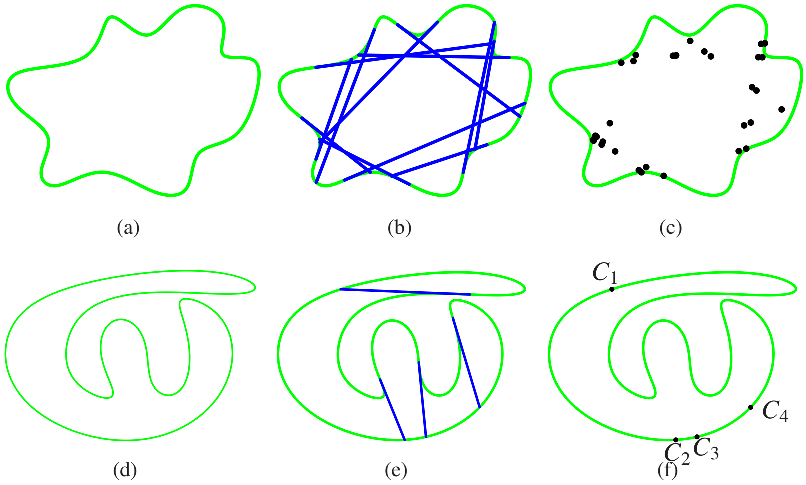


Figure 7: Demonstration of how the initial candidate guards are obtained for two test examples. (a),(d): Two C^1-continuous curves; (b),(e): IPTs drawn from each of them; (c), (f): The candidate guard set comprising of the points of intersection of IPTs with each other, or the occlusion points in case the IPTs are disjoint.

W_m and W_n are not visibility-disjoint. This is a contradiction on the definition of a witness point. □

Corollary 2. $|W| = |\mathcal{G}|$ *implies* $\mathcal{G}$ *is the minimal set of guards required to cover the domain.*

As a result of this, we can say that W is a lower bound on the number of guards $\mathcal{G}$ and can give a measure of approximately how close the value of $|\mathcal{G}|$ is to the optimum.

3. Overview of the algorithm

The algorithm starts by finding a set of candidate guards derived from IPTs and their intersections (Section 4.1.1). Out of the candidate guards, one guard is chosen in each iteration based on maximum visibility (identified using the number of internal tangents from the guard) and the first order approach followed in hidden Markov models [19] (Section 4.2). Identification of internal tangents, a local-based approach has been combined with a 'look ahead' approach (local approaches have been shown to be working well in practise, see [16]). Though the 'look ahead' can be adopted at any level, in this work, we have employed to one level, termed as 'first order'. A graph structure $\mathfrak{G}$ is initiated with the starting guard as a vertex and no edges at this juncture.

In each iteration of the algorithm, the guard among all the existing guards which has the minimum number of valid internal tangents (Definition 9) is identified (this is dealt in Section 4.3) and will henceforth be referred to as *anchor guard* for that iteration. The candidate guards for the next iteration are only chosen from points lying on the internal tangents of the anchor guard (see Section 4.1.2 for details). The subsequent guard is chosen from these candidates by using maximum visibility and the first order approach along with the empty triangle property (Section 4.2). The vertices in $\mathfrak{G}$ are updated with the identified guard and the edges are updated with the corresponding internal tangent joining the identified guard and the anchor guard. The above procedure is repeated until no valid internal tangent can be found from any of the guards in the graph. The algorithm then terminates.

4. Algorithm details

The algorithm consists of the following steps:

- Finding a candidate set of guards.
- Finding a subsequent guard from the set of candidate guards.
- Picking the anchor guard from the guards in the graph.
- Termination of the algorithm.

4.1. Candidate guards

4.1.1. Candidate set of guards at the start

As there are no vertices in a given closed curve (Figure 7(a)), the inflection points present in the curve are used as reference. IPTs are drawn from all the inflection points till they intersect the curve at their occlusion points (Figure 7(b)). Points of intersection of the IPTs, when they intersect, and the occlusion points when they do not, are used as candidates for the first guard (Figure 7(c)). In another test example (Figure 7(d)), as all IPTs (Figure 7(e)) only intersect the curved boundary (the IPTs do not intersect among themselves), the corresponding occlusion points are chosen as candidate guards (Figure 7(f)).

4.1.2. Candidate guards after identifying at least one guard

An anchor guard is chosen from the existing guards (as per Section 4.3) and only the valid internal tangents drawn from the anchor guard are used for finding candidates for the subsequent guard. Let G be the anchor guard and GO be an internal tangent drawn from G whose silhouette point is S. Clearly, G does not cover the entire domain because it has a valid internal tangent GO. As per Rule 1, another guard should lie somewhere on the line segment SO. Hence, we consider the occlusion point O

and the points of intersection of the IPTs with the line segment SO as candidates for choosing the next guard.

For example, in the Figure 8(a), the candidates for the tangent GO_1 are C_1, C_2 and O_1. In a similar manner, further candidate guards such as C_3, C_4 and O_2 can be obtained from the other valid internal tangent GO_2, as they lie beyond its silhouette points (S_1 and S_2 in the Figure 8(a)) from G (do note that P_1, P_2, P_3 and P_4 have been excluded as they lie before the silhouette points). Figure 8(b) shows the set of candidate guards.

Though Proposition 1 only talks about points lying strictly interior to the domain, it holds good even when the candidate guard is a point on the boundary $C(t)$ if the point is convex, whereas it need not be true if the point is concave (Definition 2). For example, assume that we have a domain as shown in Figure 8(c) and the guard G_1 has been placed. This guard has only one valid internal tangent, G_1o, and since no IPT intersects with it, the only candidate guard is the occlusion point o. Placing the subsequent guard at o does not cover $\mathcal{D}$ fully, and since one cannot draw any further internal tangent from o, the algorithm will terminate even though G_1 and o do not cover the entire domain. This happens because o is concave. In practise, when a candidate guard is a concave point and no valid internal tangent can be drawn from that guard, a small perturbation towards the interior of the domain is made (such as o perturbed to G_2 in Figure 8(c)) and this is added as the subsequent guard so that the algorithm does not halt prematurely.

Though there is no hard and fast rule to select a set of candidate guards, the procedure described here is similar to the one used in [16] from the algorithms A_2 and A_{11} that eventually were proven to be a good candidate set. Description for determining candidate guards is shown in Algorithm 1.

Algorithm 1 $\{CG\} = CandGuards(G)$

Input: An anchor guard G (NULL at the start).
Output: A set of candidate guards $\{CG\}$.

- Let P be the set of IPTs.
- **if** $G \neq$ NULL **then**
  - Find all valid internal tangents from G. Let the silhouette points be $\{S\}$ and corresponding occlusion points $\{O\}$.
  - Candidate Guards $\{CG\} = \{O\} \cup \{P_i \cap S_jO_j\}$, $\forall P_i \in P$, $\forall j \neq i$
- **else**
  - Candidate Guards $\{CG\} = \{O\} \cup \{I | I = P_i \cap P_j \forall \{P_i, P_j\} \in P, j \neq i\}$
- **end if**
- **return** $\{CG\}$.

4.2. *Selecting from the candidate guards - First order approach*

Once the candidates have been obtained, one needs to choose a guard among these and update the graph $\mathfrak{G}$ accordingly. For this, the visibility of each candidate is determined by looking at internal tangents from it, as the presence of further valid internal tangents implies it cannot see part of the domain (Proposition 1). It may be noted that the candidate guards themselves are arrived at by drawing internal tangents from an anchor guard.

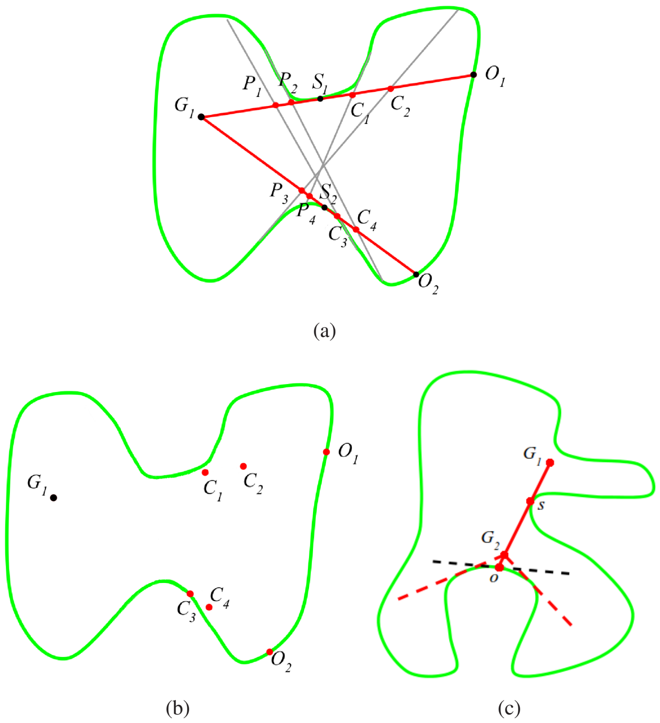


Figure 8: (a) Candidate set of guards for the given curve. The IPTs of the curve are shown in grey and the internal tangents from the existing guard G are in red, (b) The candidate guards C_1, C_2, C_3, C_4, O_1 and O_2 obtained as per the approach in 4.1.2. (c) Guard at concave segments.

Algorithm 2 $G = SelectTheGuard(\{CG\})$

Input: A set of candidate guards $\{CG\}$ obtained from the anchor guard G_a (=NULL at the start).
Output: The chosen guard G^*.

- Let $\{CG\} = \{C_1, C_2, \ldots\}$
- **for** Each guard C_i **do**
  - Draw the set of internal tangents I from C_i.
  - **if** $G_a \neq NULL$ **then**
    - **for** Each I_j in I **do**
      - Let S_j be the silhouette point and O_j be the occlusion point.
      - Let $\mathcal{G}$ be the vertex set of $\mathfrak{G}$.
      - **if** $\exists g \in \mathcal{G}$ such that $\Delta(S_jO_jg) \cap C(t) = \phi$ **then**
        - $I = I \backslash I_j$.
      - **end if**
    - **end for**
  - **end if**
  - $G_1{}^* = \{C_i | C_i$ has minimum $|I|\}$.
  - Let G^* be a random guard from the set $G_1{}^*$.
- **end for**
- $\mathfrak{G} = \mathfrak{G} \cup (G^*, I_i)$, where I_i is the internal tangent between G^* and G_a (I_i is NULL for the starting guard).
- **return** G^*.

Looking at the internal tangents from the candidate guards to choose the subsequent guard is akin to the first order approach typically followed in hidden Markov models [19]. Broadly, an nth order approach implies that the state at a predecessor level is

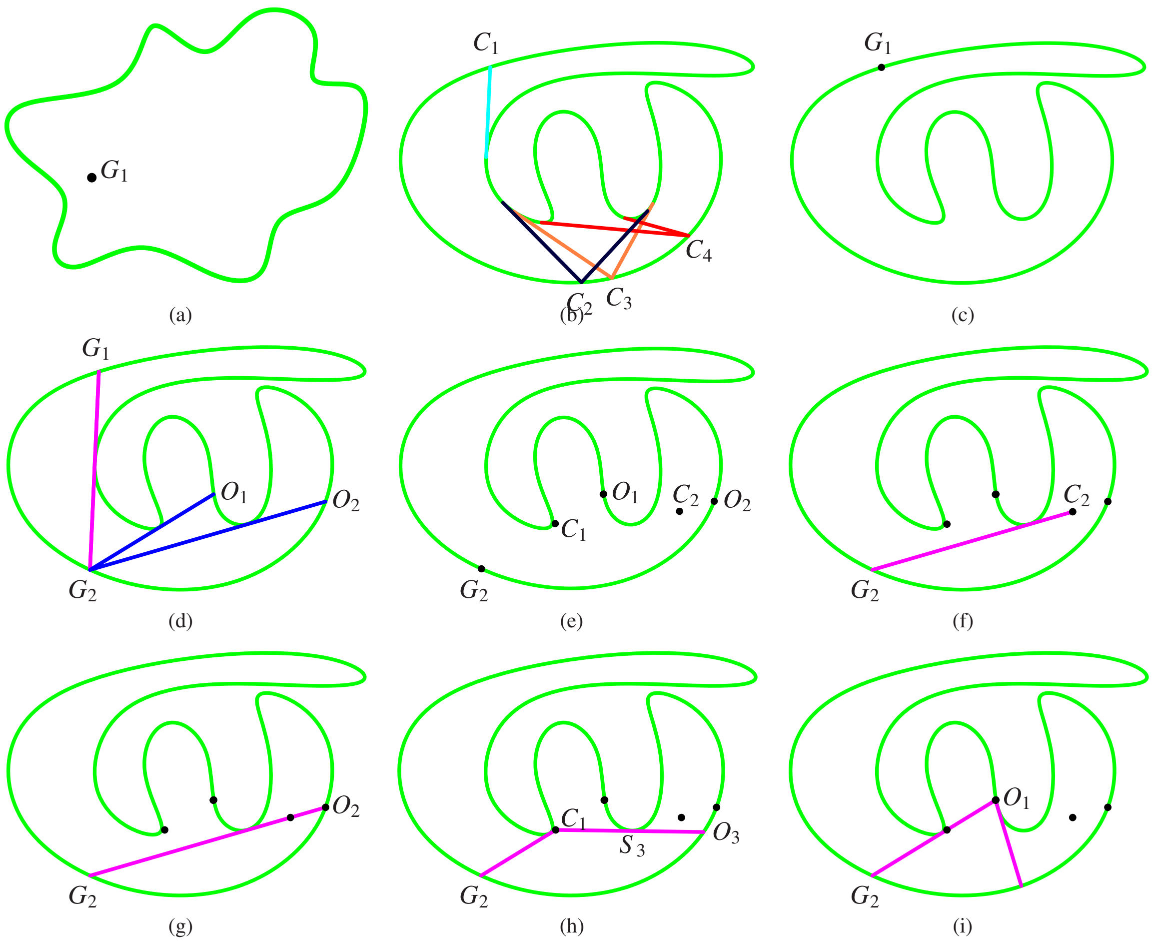


Figure 9: Illustration of choosing a guard from a set of candidate guards. (a) From the candidates in figure 7(c), G_1 is chosen as a subsequent guard since it has no internal tangents (and hence it also covers the entire domain). (b)-(c) Internal tangents are drawn from the candidate points obtained in Figure 7(f) and G_1, having only one valid internal tangent, is chosen. (d)-(e) An intermediate step in the algorithm where two guards G_1 and G_2 have been obtained and new candidates C_1, C_2, O_1, and O_2 are obtained taking G_2 as the anchor guard. Figures (f)-(i) show the internal tangents drawn from each of these candidate guards. Since all of them form empty triangles with G_2, they have no internal tangents and any can be chosen as the subsequent guard. C_2 is chosen in this case.

arrived at after a prediction made at n successive levels. If only one successive level is employed, then it is termed as first-order approach.

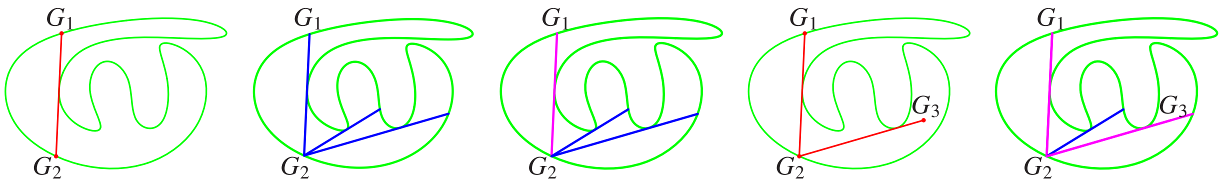
To select a guard from the candidate set, the number of valid internal tangents from each candidate guard is counted. Recall that a valid internal tangent should not form an 'empty triangle' with any of the existing guards. Note that while choosing starting guard from the candidates obtained in Section 4.1.1, the empty triangle check is redundant because no guards exist at the start. So the guard with the minimum number of internal tangents is chosen as the starting guard. For example, Figure 9(b) shows the candidate guards at the start of the algorithm and their internal tangents. All candidates but one (C_1) have two internal tangents and hence C_1 is used as the starting guard (G_1 in Figures 9(c), 9(a) for the respective test examples in Figures 7(d), 7(a)) and is added to the currently empty graph $\mathfrak{G}$.

However, when $\mathfrak{G}$ is non-empty, one needs to check for the presence of empty triangles between an internal tangent and each of the existing guards. The number of valid internal tangents (Definition 9) is counted for each candidate guard, and the one with the minimum number is picked as the subsequent guard. The graph $\mathfrak{G}$ is then updated with this guard as a vertex and the internal tangent between the guard and its corresponding anchor guard as an edge.

For example, given the anchor guard G_2, the candidate guards are shown in Figure 9(e). Figures 9(f)-9(i) each shows internal tangents drawn from one of the four candidates. All the internal tangents form empty triangles with G_2, and hence each candidate guard has the same number of valid internal tangents, implying any of them can be chosen as the subsequent guard. For example, if C_2 is selected as the guard, then C_2 is added to the vertex set of $\mathfrak{G}$ and the internal tangent G_2C_2 is added to the edges. Algorithm 2 encodes the procedure for selecting a guard.

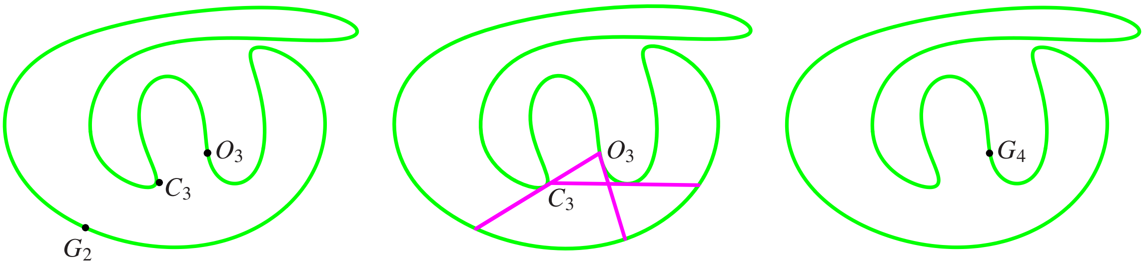
4.3. *Picking an anchor guard*

Next, an anchor guard needs to be picked among the guards currently in $\mathfrak{G}$. This is the guard from which internal tangents will be drawn to find candidates for the next iteration. A simple procedure is used to do this. All the valid internal tangents



(a) (b) (c) (d) (e)

Figure 10: (a)-(c) show how an anchor guard is picked from a guard set of G_1 and G_2. (a) Existing guards, (b) Internal tangents drawn from each of them (blue), (c) The tangent G_1G_2 is not considered valid as it overlaps with an existing edge in $\mathfrak{G}$. Hence, G_1 has zero intTs while G_2 has two. G_2 is picked as the anchor guard. (d)-(e) illustrate the picking of an anchor guard after G_3 is added to the guard set. (d) Guards G_1,G_2 and G_3 and the current graph $\mathfrak{G}$ (e) The internal tangents drawn from each of the guards (the ones which are not valid are shown in pink). G_2 is picked as an anchor guard because it has least non-zero valid intTs.

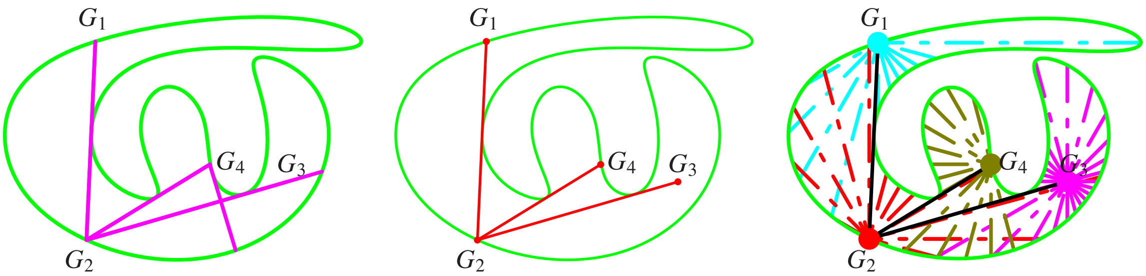


(a) Candidate guards from G_2 (after the set of guards obtained till Figure 10(e)).

(b) Internal tangents from the candidate guards.

(c) G_4 is added to the graph.

Figure 11: Finding the candidate guards from G_2 and choosing the subsequent one.



(a) From all the guards in the graph, no guard is having a valid internal tangent.

(b) Final graph (spanning tree) having four guards covering the entire domain.

(c) Guards with covering regions

Figure 12: Termination of the algorithm.

from each of the guards in $\mathfrak{G}$ are found (excluding the tangents forming edges in $\mathfrak{G}$). The guard with the minimum number of non-zero valid internal tangents is then picked as an anchor guard.

Figure 10(a)-10(c) illustrate the flow for the graph consisting of guards G_1 and G_2, where G_2 gets picked as an anchor guard for finding a subsequent guard. Figure 10(d)-10(e) show another instance of the graph from which an anchor guard has to be picked for further iteration. The internal tangents which are not valid, either because of the empty triangle property or because they overlap with existing edges in the graph, are indicated in pink. Algorithm 3 indicates the steps in picking an anchor guard from the current graph.

Algorithm 3 $SG = PickAnchorGuard(\mathcal{G})$

Input: Current vertex set (guards) $\mathcal{G}$ of $\mathfrak{G}$.
Output: NULL or an anchor guard G_a from $\mathcal{G}$.

```
Let I_i be the set of valid internal tangents for each G_i ∈ G.
if ∀i, |I_i| == 0 then
    return NULL
else
    return G_a with minimum non-zero |I_i|.
end if
```

4.4. Termination of the algorithm

The algorithm terminates when no guard with non-zero number of valid internal tangents is available as an anchor guard from the current graph, as it indicates that no internal tangents can be drawn from any of the guards in the graph, i.e. the current set of guards can see the entire domain. The termination is encapsulated in Algorithm 3 itself.

4.5. Illustration of the algorithm

A high level description of the entire algorithm which returns the set of guards covering the domain having a curved boundary is given in Algorithm 4.

Algorithm 4 FindGuards(Curved Domain $\mathcal{D}$)

```
Input: Domain 𝒟 having a continuous curved boundary.
Output: 𝔊 whose vertex set give the guards (VLs) covering 𝒟.
𝔊 = {}, i = 0
Find the set of candidate guards, CG = CandGuards(NULL)
Select the guard from CG, G_1 = SelectTheGuard(CG)
while (G_a=PickAnchorGuard(⋃_{i=1}^{n} G_i)) ≠ NULL do
    Find the set of candidate guards, CG = CandGuards(G_a)
    Select the subsequent guard, G_i = SelectTheGuard(CG)
    i = i + 1
end while
return 𝔊
```

Algorithm 5 FindWitnessPoints(Curved Domain $\mathcal{D}$)

```
Input: Domain 𝒟 having a continuous curved boundary.
Output: W consisting of witness points inside the domain 𝒟.
W = {}, i = 0
Find the set of inflection points ℐ of C(t). Set the unvisited inflection points I_rem = ℐ.
Find the visibility regions 𝒱(I) for each inflection point by finding its IPTs and IntTs.
while I_rem ≠ NULL do
    Find the inflection point I* which is visible to least number of other inflection points.
    Select this as a witness point, W = W ∪ I*.
    Update I_rem = {I | I ∈ ℐ and I ∩ 𝒱(I*) = φ}
end while
for all w ∈ W do
    Find the occlusion points of IPTs and IntTs from w, and draw IntTs from each of them.
    Look for silhouette points s such that 𝒱(s) ∩ 𝒱(W) = φ and add s to W.
end for
return W
```

The illustration of the algorithm (Algorithm 4) uses the test example 2 (Figure 7(d)). The candidate guards are identified using the intersection points of the IPTs and the occlusion point of an IPT, if it does not intersect any (Figures 7(e) and 7(f)). Using the first-order approach, the starting guard (G_1) is then identified (Figure 9(c)). Graph $\mathfrak{G}$ is initiated with the guard G_1 with no edges.

Since there is only one guard, this guard gets picked as the anchor guard for the next iteration. In the next iteration, candidate guards are identified by internal tangents drawn from G_1 and using the first order approach, G_2 is identified as the subsequent guard (Figure 10(a)). $\mathfrak{G}$ is then updated with the vertex G_2 and the edge G_1G_2. Now, using $PickAnchorGuard(G_1 \cup G_2)$, among G_1 and G_2, G_2 is picked as the next anchor guard (Figure 10(c)). From G_2, the set of candidate guards are identified (Figure 9(e)) and following the first order approach and performing checks for valid internal tangents (Figures 9(f)-9(i)), guard G_3 is added to $\mathfrak{G}$ along with the internal tangent as the edge (Figure 10(d)). Then, among G_1, G_2 and G_3, G_2 is picked as the next anchor guard (Figure 10(e)). Candidate guards from G_2 and their further processing using first order approach is illustrated in Figure 11. Both C_3 and O_3 have no valid internal tangents, so O_3 is added as G_4 to $\mathfrak{G}$ with the internal tangent between G_4 and its anchor guard G_2 as the edge. After this, $PickAnchorGuard(\bigcup_{i=1}^{4} G_i)$ returns NULL, as no guard has a valid internal tangent (Figure 12(a)). Hence the algorithm terminates and returns the graph $\mathfrak{G}$ (which essentially is a spanning tree, Figure 12(b)). Figure 12(c) shows the coverage of each guard in dash-dot lines.

4.6. *Finding a set of witness points*

Unlike the guard set, the witness points need to be visibility disjoint. Since we want as many witness points as possible in order to estimate a good approximation ratio, they should be placed at regions which have low visibility. Thus, inflection points, silhouette points of IntTs, and points on concave regions make for good candidates for witness points, as opposed to the occlusion points of IPTs and points lying interior to the domain (which are good candidates for finding the guards).

Algorithm 5 describes the steps required for finding witness points. A candidate-based first order approach is used wherein the inflection point whose visibility regions intersect with the least number of other inflection points is chosen as the starting witness point. A set of 'unvisited inflection points', I_{rem}, containing the inflection points which are visibility disjoint with every point in the witness set is maintained. This set serves as the candidate set for finding new witness points. In each step, the point which can see least number of points in I_{rem} is chosen as a witness point and the inflection points visible to it are removed from I_{rem}. The process iterates until all inflection points are visited, i.e. until I_{rem} becomes a null set.

It can be noted that the witness points in the above procedure have been obtained solely from inflection points. Hence, this will not give a good estimate in the case of curves having long spiral regions without inflection points. In order to account for these regions, internal tangents are drawn from the occlusion points of the IPTs and IntTs of each witness point. If any silhouette points of these IPTs and IntTs is also visibility disjoint from the existing witness points, it is added to the set of witness points.

5. Results

The developed algorithm has been implemented using the IRIT geometric kernel [20], which contains function for the computation of internal tangents, and inflection, silhouette and occlusion points (for further details on such computations, please refer [8]). The curves used are represented using non uniform rational B-spline (NURBS) for the purpose of implementation, though the algorithm itself has no such restriction.

Figure 13: Results: Guards (shown in red dots), spanning tree (red dots with red lines), witness points (dots in dark blue).

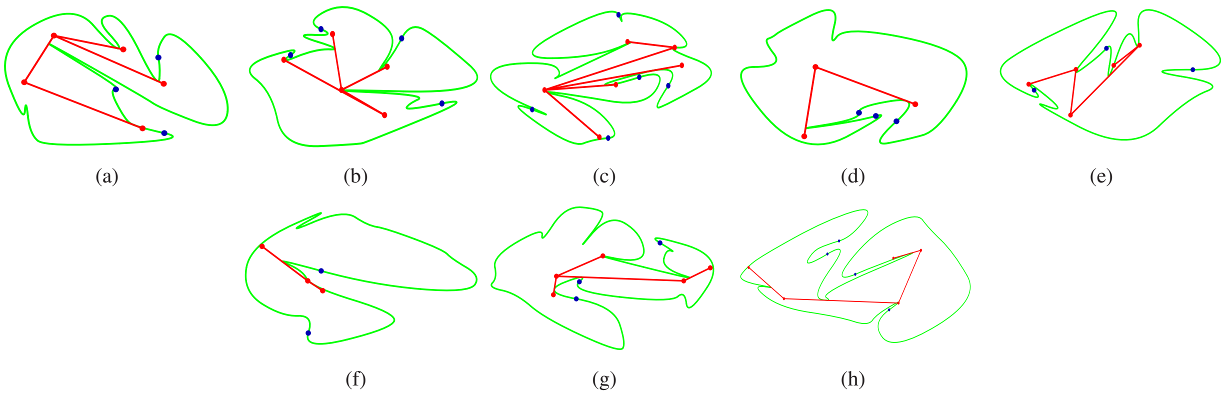


Figure 14: Results: Guards (shown in red dots), spanning tree (red dots with red lines), witness points (dots in dark blue) for random shapes.

Fig. no.	13(a)	13(b)	13(c)	13(d)	13(e)	13(f)	13(g)	13(h)	13(i)	13(j)	13(k)	13(l)	13(m)	13(n)	13(o)	13(p)	13(q)	13(r)
NOG	1	2	3	5	2	2	3	2	2	4	3	4	4	5	6	6	4	6
WP	1	2	3	4	2	2	3	2	2	3	2	3	4	5	4	5	2	4
AR	1	1	1	1.25	1	1	1	1	1	1.33	1.5	1.33	1	1	1.5	1.2	2	1.5
Time(s)	4.11	4.47	3.49	5.5	3.9	7.98	5.01	8.88	4.7	4.02	7.53	8.4	5	8.54	8.88	8.85	7.54	7.5

Table 1: The number of guards (NOG), number of witness points (WP) for various curved shapes, AR (= NOG/WP) - Approximation ratio and Running Times (in seconds) for curves in Figure 13.

Fig. no.	14(a)	14(b)	14(c)	14(d)	14(e)	14(f)	14(g)	14(h)
NOG	5	5	6	3	5	3	5	5
WP	3	4	5	3	3	2	3	4
AR	1.67	1.25	1.2	1	1.67	1.5	1.67	1.25
Time(s)	7.98	11.17	18.95	5.57	10.18	5.73	11.59	14.8

Table 2: The number of guards (NOG), number of witness points (WP) for randomly generated curved shapes, AR (= NOG/WP) - Approximation ratio and Running Times (in seconds) for curves in Figure 14.

Implementation results in Figure 13 indicate that the algorithm can handle a wide variety of curves, right from curves having a large number of inflection points to those having very few. The tested shapes include typical used ones in visibility location problems such as star-shapes, comb-like objects, spirals etc. In general, depending on the characteristics of the shape, we can either have a larger number of guards than the number of inflection points, or vice versa. For example, locally spiral-shaped objects (Figure 13(d)) have fewer inflection points but require more number of guards. This scenario has been captured by the algorithm (also see Figure 13(m) - 13(p)). On the other hand, star-shaped objects that have larger number of inflection points might require only one guard (Figure 13(a)). This has also been captured. Results for comb-like shapes are shown in Figures 13(b) and 13(c). Results for a combination of high curvature thinner and thicker regions are shown in Figures 13(e) - 13(g). The algorithm has also produced guards for objects that have low curvature regions in conjunction with higher ones in Figures 13(i) and 13(l). Guards for an object with a constricted passage is shown in Figure 13(k). The algorithm can also handle high curvature regions having different widths (Figure 13(c)). The algorithm also captures scenarios where all the guards may lie interior to the curved boundary (Figure 13(i), 13(q) and 13(r)). A few more results for randomly shaped curves having very sharp turns is shown in Figure 14. These results show that the algorithm can generate guards for differently configured curves. In both Figures 13 and 14, the outputs demonstrate that the guards form a spanning tree.

5.1. Discussion

5.1.1. On the optimal number of guards

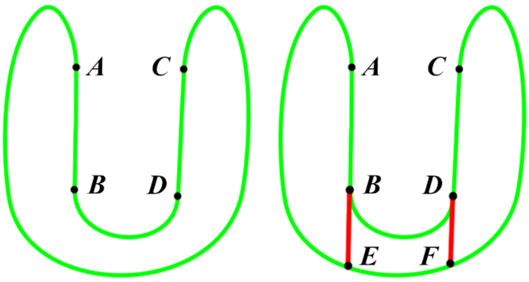
Let N_g be the number of guards returned by our algorithm. Let $|W_p|$ be the maximum cardinality visibility independent set (i.e., maximum number of witness points that one can determine for a given curve). Then the ratio $N_g/|W_p|$ can be said to estimate how close our algorithm's output is to the likely optimum [16]. Let this ratio be termed as approximation ratio (*AR*). Tables 1 and 2 show the number of guards, witness points and *AR* for the set of input curves in Figures 13 and 14 respectively. In many of the inputs, the obtained *AR* was 1, indicating the corresponding N_g is optimal (Corollary 2). In few of the cases, the *AR* was 1.5 and in others, between 1 and 1.67. In the worst case, the AR obtained was 2 (Figure 13(q)). Another point worth noting is that the maximum number of witness points need not always be equal to the minimum number of guards (e.g. in Figure 13(q), there is no way of placing three or more witness points whose visibility regions do not intersect), i.e. W_p is not necessarily a 'tight' lower bound in all cases and at times, it may be less than the optimal number of guards. This might hence lead to a higher apparent approximation ratio than is actually the case. The algorithm has been tested on many cases such as star-shaped, coil-like, comb-like etc. typically used as worst-case scenarios in the visibility location problems. Based on the testing for a number of curves (other than the results shown in Figures 13 and 14) and to be on the safer side, we can clearly say that the N_g is not more than twice the optimal number and hence the following conjecture:

Conjecture 1. *For a given curved boundary, $N_g \leq 2N_{op}$, where N_{op} is the optimal number of guards.*

5.1.2. Comparison

To the best of our knowledge, no algorithm seems to exist for the visibility location problem for a domain having a curved boundary with no explicit vertices. Placing the guard on the curved boundary has been addressed in [8] using a discretized approach with an exhaustive search and gives a conservative estimate (not optimal). Perhaps, we use only a subset of computations as that of [8]. We believe that our approach of using internal tangents characterizes the local shape and geometry of the domain well. Framework for visibility problems in [14] also use a discretization-based approach whereas the approach provided in [15], apart from discretization, also uses one hundred guards as the initial set. Running time of the results on an intel core i5, 2.80GHz with a 4GB RAM are of the order of a few seconds, as indicated in Tables 1 and 2 (the running times of other works cannot be compared directly as the configurations are different from those in this paper). As our running times are not too slow, it is a reasonable trade off with discretization, which introduces gaps in visibility maps and hence not guaranteeing 100% coverage [14] as opposed to 100% coverage as can be seen from the results in Figures 13 and 14. It can be noted that, in [8], the running times are of the order of a few seconds to over a minute on a modern PC. As our starting candidate guards include intersection of IPTs, our algorithm has the ability to capture domains that are guarded by a single guard, which is not possible to achieve if the guards are only on the boundary (such as [8]), see Figure 1.

Other literatures related to curved boundary such as [12, 13] use the explicit notion of vertices. It is not possible to compare with their results even after artificially adding vertices at inflection points and splitting the curve boundary into concave and convex segments. This is because [12] only considers polygons with edges which are either *all* piecewise convex or *all* piecewise concave, and [13] is only restricted to curves where the



(a) (b)

Figure 15: (a) An example where straight line segments may lead to absence of inflection points; (b) A possible fix by adding the tangents at the endpoints of the straight line sections, if interior to $C(t)$, to the list of IPTs (BE and DF).

edges between any two vertices are convex.

5.1.3. Limitations

One limitation of this algorithm is that the number of candidate guards could be very large at the start, even though the curve might finally require only a single guard. For example, Figure 7(c) has a few candidate guards at the start, where as only one guard is required to cover the entire domain (Figure 13(a)). Also, ties obtained when multiple candidates have the same number of valid internal tangents are currently broken randomly. A better method of breaking ties remains to be found. The input curves are assumed to be C^1-continuous and hence algorithm does not handle curves with discontinuities. If there is a discontinuity such as a cusp, then there is no unique well-defined tangent at such a point, and hence the algorithm has to be suitably modified to handle such cases. One possible direction to explore would be adding the segments of the tangents at each cusp that lie interior to the curve to the set of existing IPTs at the start of the algorithm. Moreover, the algorithm cannot handle curves which contain straight line portions if the straight line portions begin or end at what otherwise would have been inflection points (such as segments AB and CD in Figure 15(a)). It might be possible to address this by detecting straight line segments and adding the tangents drawn at their endpoints, if interior to the curve, to the set of IPTs (see Fig 15(b)). Also, an internal tangent with more than one tangent point will introduce more than one silhouette point and we have not handled this case.

6. Conclusions and future work

In this paper, an algorithm for visibility locations (guards) that can be interior/on the curved boundary has been developed and implemented. It has been proposed that the guards have to form a spanning tree that provides a near-optimal number of visibility locations. Using the witness points, it has been shown that, in many of the tested cases, the algorithm results in optimal number of VLs. The results also enabled us to conjecture that the algorithm does not result in more than twice number of optimal VLs, as the approximation ratio was no more than 2. Results indicate that the algorithm is very amenable for implementation.

Future work would involve identifying ways for reducing the number of starting candidate guards. Another possible work is to employ the algorithm for a curved domain having holes (or obstacles) as well as to curved surfaces. Applications are also being looked at.

References

[1] H.-K. Ahn, M. de Berg, P. Bose, S.-W. Cheng, D. Halperin, J. MatouÅek, O. Schwarzkopf, Separating an object from its cast, Computer-Aided Design 34 (8) (2002) 547 – 559. doi:http://dx.doi.org/10.1016/S0010-4485(01)00119-1.

[2] K. Hui, S. Tan, Mould design with sweep operations a heuristic search approach, Computer-Aided Design 24 (2) (1992) 81 – 91. doi:10.1016/0010-4485(92)90002-R.

[3] G. Elber, X. Chen, E. Cohen, Mold accessibility via gauss map analysis, in: Proceedings Shape Modeling Applications, 2004., 2004, pp. 263–272. doi:10.1109/SMI.2004.1314513.

[4] W. R. Scott, G. Roth, J.-F. Rivest, View planning for automated three-dimensional object reconstruction and inspection, ACM Comput. Surv. 35 (1) (2003) 64–96. doi:10.1145/641865.641868.

[5] B. R. Sundar, R. Muthuganapathy, Shortest path in a multiply-connected domain having curved boundaries, Computer-Aided Design 45 (3) (2013) 723 – 732. doi:http://dx.doi.org/10.1016/j.cad.2012.12.003.

[6] J. K. Johnstone, A parametric solution to common tangents, in: Proceedings International Conference on Shape Modeling and Applications, 2001, pp. 240–249. doi:10.1109/SMA.2001.923395.

[7] A. Bottino, A. Laurentini, L. Rosano, A new lower bound for evaluating the performances of sensor location algorithms, Pattern Recognition Lettersdoi:10.1016/j.patrec.2009.05.020.

[8] G. Elber, R. Sayegh, G. Barequet, R. R. Martin, Two-dimensional visibility charts for continuous curves, Shape Modeling and Applications, 2005 International Conference (2005) 206–215.

[9] J. O'Rourke, Art Gallery Theorems and Algorithms, The International Series of Monographs on Computer Science, Oxford University Press, New York, NY, 1987.

[10] M. Liu, Y. shen Liu, K. Ramani, Computing global visibility maps for regions on the boundaries of polyhedra using minkowski sums, Computer-Aided Design 41 (9) (2009) 668 – 680. doi:http://dx.doi.org/10.1016/j.cad.2009.03.010.

[11] M. Liu, K. Ramani, On minimal orthographic view covers for polyhedra, in: 2009 IEEE International Conference on Shape Modeling and Applications, 2009, pp. 96–102. doi:10.1109/SMI.2009.5170169.

[12] M. I. Karavelas, C. D. Tóth, E. P. Tsigaridas, Guarding curvilinear art galleries with vertex or point guards, Computational Geometry 42 (6-7) (2009) 522–535. doi:10.1016/j.comgeo.2008.11.002.

[13] M. I. Karavelas, Guarding curvilinear art galleries with edge or mobile guards via 2-dominance of triangulation graphs, Comput. Geom. Theory Appl. 44 (1) (2011) 20–51. doi:10.1016/j.comgeo.2010.07.002.

[14] N. Shragai, G. Elber, Geometric covering, Computer-Aided Design 45 (2) (2013) 243 – 251, Solid and Physical Modeling 2012. doi:http://dx.doi.org/10.1016/j.cad.2012.10.007.

[15] R. Strauss, F. Isvoranu, G. Elber, Geometric multi-covering, Computers & Graphics 38 (2) (2014) 222 – 229. doi:http://dx.doi.org/10.1016/j.cag.2013.10.018.

[16] Y. Amit, J. S. B. Mitchell, E. Packer, Locating guards for visibility coverage of polygons, Int. J. Comput. Geometry Appl. 20 (5) (2010) 601–630.

[17] G. Elber, J. K. Johnstone, M.-S. Kim, J.-K. Seong, The kernel of a freeform surface and its duality with the convex hull of its tangential surface, International Journal of Shape Modeling 12 (2006) 129–142.

[18] T. H. Cormen, C. Stein, R. L. Rivest, C. E. Leiserson, Introduction to Algorithms, 2nd Edition, McGraw-Hill Higher Education, 2001.

[19] L. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (2) (1989) 257–286. doi:10.1109/5.18626.

[20] G. Elber, IRIT 10.0 User's Manual, The Technion—IIT, Haifa, Israel (2009).