Two-Dimensional Visibility Charts for Continuous Curves

Gershon Elber,   Robert Sayegh,   Gill Barequet
Computer Science Department
Technion
Haifa 32000, Israel
{gershon, roberts, barquet}@cs.technion.ac.il

Ralph R. Martin
School of Computer Science
Cardiff University
Cardiff, CF24 3AA, UK
ralph@cs.cf.ac.uk

Abstract

This paper considers computation of visibility for two-dimensional shapes whose boundaries are C^1 continuous curves. We assume we are given a one-parameter family of candidate viewpoints, which may be interior or exterior to the object, and at finite or infinite locations. We consider how to compute whether the whole boundary of the shape is visible from some finite set of viewpoints taken from this family, and if so, how to compute a minimal set of such viewpoints. The viewpoint families we can handle include (i) the set of viewing directions from infinity, (ii) viewpoints on a circle located outside the object (for inspection from a turntable), and (iii) viewpoints located on the walls of the shape itself.

We compute a structure called a visibility chart, *which simultaneously encodes the visible part of the shape's boundary from* every *view in the family. Using such a visibility chart, finding a minimal set of viewpoints reduces to the set-covering problem over the reals. Practical algorithms are obtained by a discrete sampling of the visibility chart. For exterior visibility problems, a reasonable approach is to compute an almost-optimal solution (in terms of number of viewpoints), which can be done in almost-linear time. For interior visibility problems, or when a more correct solution is required, we solve the general set-covering problem, guaranteeing an optimal solution but taking exponential time. In either case, conservative solutions are obtained, ensuring that no part of the curve remains invisible from some viewpoint. Examples are given to illustrate our algorithm.*

1. Introduction

This paper considers external and internal visibility for 2D shapes whose boundaries are continuous curves, represented, for example, as NURBS. Given a one-parameter family of candidate viewpoints, which may be interior or exterior to the object, and at finite or infinite locations, we show how to determine whether the whole boundary of the shape is visible from any *finite* set of viewpoints from this family, and if so, how to compute a *minimal set* of such viewpoints. This general formulation encompasses various important families of viewpoints, such as (i) the set of viewing directions from infinity, (ii) viewpoints on a circle located outside the object (e.g. for inspecting an object placed on a turntable), and (iii) viewpoints placed on the walls of the shape itself.

Visibility calculations of this type have various applications in such areas as mold design, inspection planning, and security. To make an n-piece mold for manufacturing, a shape requires a minimal set of n external viewing directions to be determined (if such exists) so that the exterior boundary of the shape is completely visible from these directions [10]. In vision-based inspection planning, the goal is to inspect the exterior surface of an object, efficiently [25, 26]. Typically, each viewing direction has some set-up cost (in terms of time, etc.) associated with it. If the inspection plan is to be used many times on a production line, it is important to find a minimal set of viewing directions. An idealized security application requires a set of cameras to be placed around a building so that the whole of the interior or exterior may be guarded—many problems of this type, so-called *art-gallery problems*, have been devised and solved by computational geometers for objects with polygonal boundaries [23].

A somewhat related (and well-studied) problem in computational geometry is the computation of the so-called *visibility complex* of a collection of objects [24]. This problem deals with the creation of a data structure that encodes all visibility relations between objects in the collection, and can answer visibility queries efficiently. The objects are usually polygons or have low-complexity descriptions, and the respective structure is maintained in a space dual to that of the objects. In the current work we deal with smooth curves in the plane, and use a completely different approach.

The rest of the paper is organized as follows. Section 2 describes related work in various fields. Section 3 explains

our approach to computing a set of visibility directions (or viewpoints) and gives appropriate algorithms. Various examples illustrating the method are given in Section 4. Finally, conclusions and suggestions for future work are given in Section 5.

2. Background

We first summarize existing related research and discuss how the ideas in this paper are applicable to various problems. Section 2.1 presents their relevance to *mold design*. Visibility analysis in the context of *inspection* is considered in Section 2.2, and in the context of *security*, in Section 2.3. Visibility determination is also a fundamental question in *computer graphics*, which we briefly discuss in Section 2.4.

Various other papers have treated visibility computations in general, rather than targeted a specific application; we do likewise. Thorough treatment of methods for computing *Gauss maps* can be found in [12, 27], for example.

2.1. Mold Design

Design-for-manufacturing requires CAD systems to be able to verify that the designed models can be manufactured, and to generate manufacturing plans. Two-piece molds are used in manufacturing processes such as injection molding and die casting [3, 16]. Various papers have considered *two-piece mold separability* [1, 7, 17]: can a two-piece mold be found with a corresponding opposing pair of separation directions such that the mold may be cleanly removed from the object without interference? Typically, these papers use heuristics to select some candidate directions which are then verified for separability. The resulting algorithms are not complete, in the sense that they cannot guarantee that a valid two-piece mold will be found in every case in which one exists. Although Ahn et al. [1] gave a complete algorithm, the method they implement *in practice* again falls back on heuristically verifying chosen directions, because of the high complexity of the complete algorithm.

A previous paper by Elber [15] considered the problem of determining the existence of a valid two-piece mold for a designed solid model whose boundary is represented as a NURBS surface with C^3 continuity, and finding such a mold if it exists. While that work considered 3D objects, and we consider only 2D cases here, that work is more restricted in other ways: it only considered the question of whether a *pair* of viewing directions is sufficient to see the entire boundary of the object. Here, we consider cases involving more than two viewing directions.

2.2. Inspection

Many types of manufactured goods require inspection, either of a sample, or of every part manufactured. Clearly, the inspection plan should be derived with the aid of the CAD model [21] and visibility analysis [22]. When parts made in large numbers are subject to a high sampling rate or even 100% inspection, the efficiency of the inspection process becomes significant. Much previous work has considered viewpoint planning [13, 25, 30]: the problem of choosing a set of suitable viewpoints.

In [6], the Gaussian sphere is used to determine the set of directions one needs to cover an entire model for inspection and machining applications. Other work that attempts to reason about visibility using surface normal maps over the Gaussian sphere was discussed in [12, 27], for instance. However, simply working in the normal space over the Gaussian sphere fails to take the general hidden surface problem into account—it ignores the fact that certain parts of the object may occlude others.

Certain formulations of the three-dimensional view planning problem are NP-complete [29], if the sensor is allowed to be placed at arbitrary positions and orientations, and we allow for characteristics and setups of real sensing devices. Also significant is that the *quality* of acquired sensor data may be an important consideration to trade off against the *number* of viewpoints. As a result, most work in this area has not tried to find an optimal inspection plan, but merely a good one. As is also the case for mold separation, most papers are based on a generate-and-test approach, rather than using optimization to determine viewpoints; an exception is the work by Tarabanis [28].

Most inspection research discretizes either the object description (if the object is not initially assumed to be polyhedral), e.g., [4, 28], or the viewpoint space, e.g., [29], or both. Consequently, some of these approaches, such as *aspect graphs* [4], are almost never used for objects with smooth curved boundaries due to their inherent complexity. Also, for simplicity, the viewpoints are often assumed to lie on a sphere of some fixed radius, e.g., [29]. (In this paper, we also discretize a continuous search space).

Much work in this field targets the case where the object being inspected is *not* available as a CAD model, or at least, the object under inspection differs significantly from the model. In such cases, the problem devolves to the one of finding the "next-best-view," given what has already been determined [8].

In summary, the recent survey by Scott et al. [26] noted that view planning is still an open problem despite two decades of research. The novelty of our work lies in handling objects described by continuous NURBS boundaries, albeit in 2D, and in a systematic search for the solution, rather than using a hypothesize-and-test approach.
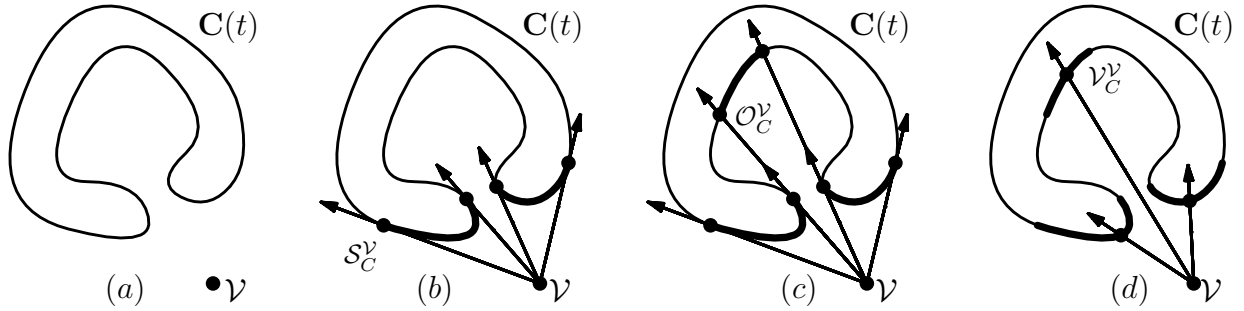
Figure 1. Given a curve $C(t)$ and viewpoint $\mathcal{V}$ **(a)**, the *silhouette* points, $\mathcal{S}_C^\mathcal{V}$, from $\mathcal{V}$ are computed in **(b)**, the *prime occluders*, $\mathcal{O}_C^\mathcal{V}$, are detected in **(c)**, and the *visible regions* (thick lines) of C from $\mathcal{V}$, $V_C^\mathcal{V}$, are computed in **(d)** via ray casting. From these we get the visibility map of $C(t)$ from $\mathcal{V}$. For clarity, only the visible silhouette points are presented.

2.3. Security

The class of *art gallery* problems is much-studied in the computational geometry literature [23]. The basic idea is that an unusually-shaped room has many paintings on the walls, and a minimal number of guards must be positioned inside the room so as to be able to see all the paintings. This type of problem is usually posed in theoretical terms, where the room is a polygon—art gallery theorems rarely consider more complex geometry.

The main difference between such problems, and molding and inspection problems, is that *interior* visibility is usually considered, rather than *external* visibility. Nevertheless, an exterior version of the art gallery problem exists, and has been dubbed the *fortress problem* [23].

The method we give in this paper is equally valid for selection of viewpoints lying inside or outside a 2D shape, being based on the definition of candidate views as belonging to a one-parameter family of viewing points or directions.

2.4. Computer Graphics

Finally, we point out that the problem of determining what is visible from a single viewpoint, given a three-dimensional model, is the well known *hidden surface* problem in computer graphics. Many algorithms exist both for polyhedral objects [2] and continuous surfaces [11]. We simply note here that our problem is clearly related to hidden surface determination, but the latter is also clearly just a first step in the current process. Thus, in this work, we build our visibility chart, as will be demonstrated, using hidden surface removal tools.

3. Selecting Viewpoints

We start by stating the problem to be solved, and then explain our algorithm for solving it.

Let $\mathcal{V}(\cdot)$ be a one-parameter family of viewpoints, $\mathcal{V}:$ $\boldsymbol{R} \rightarrow \boldsymbol{R}^2$. Members of $\mathcal{V}(\cdot)$ can be at finite positions or at infinity; in the latter case they may also be referred to as viewing directions. Informally, the problem is, given a tangent-continuous curve $\mathbf{C}$, to find a minimal set of discrete viewpoints lying on $\mathcal{V}(\cdot)$ such that every point on $\mathbf{C}$ can be seen from at least one of these views.

Viewpoints and viewing directions are handled slightly differently in the following.

3.1. Single Viewpoints

We start by considering a single viewpoint. Given a *single* viewpoint (or viewing direction), we wish to compute the *visibility map* of the curve $\mathbf{C}$ from the viewpoint $\mathcal{V}$, i.e., to identify the part of the curve that is visible from that viewpoint. We do so using a similar approach to that taken in the quantitative invisibility hidden-line removal scheme for polygonal geometry [2] and free-form surfaces [11] in $\boldsymbol{R}^3$. The geometry is first split into contiguous regions such that each region is either entirely visible or invisible; each region's visibility is then determined by shooting a ray from the viewpoint (or in the viewing direction) towards the region in question.

We use silhouette points in 2D as our starting point to solve this visibility problem.

Definition 1 *Given a viewpoint $\mathcal{V}$ in the plane and a C^1-continuous planar curve* $\mathbf{C}$ *(see Figure 1(a)):*

1. *The* silhouette points $\mathcal{S}_C^\mathcal{V}$ *comprise the set of points on* $\mathbf{C}$ *whose curve normals are perpendicular to the line between such a point and $\mathcal{V}$ (see Figure 1(b)).*

2. *The* prime occluders $\mathcal{O}_C^\mathcal{V}$ *comprise the set of points on* $\mathbf{C}$ *computed by shooting a ray from $\mathcal{V}$ through each silhouette point, and finding the* first *point the ray strikes anywhere else on* $\mathbf{C}$*, if any (see Figure 1(c)).*
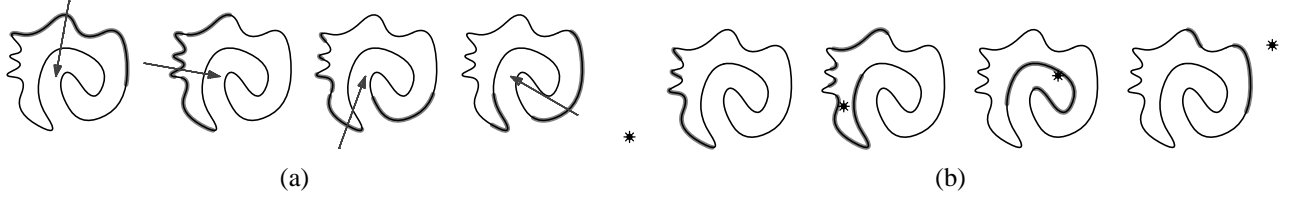
Figure 2. Visible sections (thick gray line) of a curve (thin black line) computed for various viewpoints at infinity (a) and finite viewpoints (marked with asterisks) (b).

3. **C** *is split at* $\mathcal{S}_C^{\mathcal{V}}$ *and* $\mathcal{O}_C^{\mathcal{V}}$ *into sections. The* visible sections *of* **C**, $\mathcal{V}_C^{\mathcal{V}}$, *comprise the parts of* **C** *that are visible when viewed from* $\mathcal{V}$, *and that are not occluded by other parts of* **C** *(See Figure 1(d)).*

Examples are shown in Figure 2(a) for viewpoints at infinity and in Figure 2(b) for finite viewpoints.

Given a viewing direction $\mathcal{V}$ from infinity, the silhouette points, $\mathcal{S}_C^{\mathcal{V}}$, may be readily computed by finding all the roots of

$$\mathbf{C}'(t) \times \mathcal{V} = 0, \qquad (1)$$

where $\mathbf{C}(t)$ is a parametric representation of **C**.

Let $t_s^i \in \mathcal{S}_C^{\mathcal{V}}$ be a silhouette point from $\mathcal{V}$. The prime occluders due to $\mathbf{C}(t_s^i)$ (again with a viewing direction from infinity) can be determined by finding the roots of

$$(\mathbf{C}(t_s^i) - \mathbf{C}(r)) \times \mathcal{V} = 0, \qquad (2)$$

for all silhouette points $t_s^i \in \mathcal{S}_C^{\mathcal{V}}$. A point $\mathbf{C}(r)$ that satisfies Eqn. (2) lies on the ray in the viewing direction through the silhouette point $\mathbf{C}(t_s^i)$. We ignore the trivial solution $t_s^i = r$, and sort such points along the ray to find the $\mathcal{O}_C^{\mathcal{V}}$ prime occluder locations where the curve disappears behind the silhouette points $\mathbf{C}(t_s^i)$.

Splitting the curve **C** at all silhouette locations and all prime occluder locations produces curve segments that are either completely visible or completely invisible. Rays are shot from $\mathcal{V}$ toward the middle of each such curve segment to decide its visibility. A segment is visible if and only if the first point the ray hits lies within the the curve segment.

Minor modifications are needed to the above computations in the case of a *finite* viewing point at $\mathcal{V} = \mathbf{p}$. Silhouette points are found by solving

$$\mathbf{C}'(t) \times (\mathbf{C}(t) - \mathcal{V}) = 0, \qquad (3)$$

and the prime occluders due to the silhouette point $\mathbf{C}(t_s^i)$ are found from the roots of

$$(\mathbf{C}(t_s^i) - \mathbf{C}(r)) \times (\mathbf{C}(t_s^i) - \mathcal{V}) = 0. \qquad (4)$$

Solutions to Eqns. (1)–(4), as well as other constraints presented in this work, are sought using a multivariate solver presented in [14].

3.2. A Family of Viewpoints

So far, given a *single* viewpoint, we have seen how to compute the visible part of the curve from that viewpoint, using a classical hidden-line algorithm. We now generalize this idea. We combine the visibility maps for $\mathbf{C}(t)$ for all possible viewpoints from a one-parameter family parameterized by θ, giving a *single structure* we call a *visibility chart*. It is a bivariate function of θ and t.

Let $\mathcal{V}(\theta)$ denote all possible viewpoints in the family. For example, given viewing points located on a circle, $\theta \in [0, 2\pi]$, parameterizes the unit circle S^1. In the case of viewpoints lying on the circle at infinity, we may rewrite Eqns. (1) and (2) in the form

$$\mathbf{C}'(t) \times \mathcal{V}(\theta) = 0, \qquad (5)$$
$$(\mathbf{C}(t) - \mathbf{C}(r)) \times \mathcal{V}(\theta) = 0. \qquad (6)$$

More generally, for *any* one-parameter family of finite viewpoints $\mathcal{V}(\theta)$, we may replace Eqns. (3) and (4) by

$$\mathbf{C}'(t) \times (\mathbf{C}(t) - \mathcal{V}(\theta)) = 0, \qquad (7)$$
$$(\mathbf{C}(t) - \mathbf{C}(r)) \times (\mathbf{C}(t) - \mathcal{V}(\theta)) = 0. \qquad (8)$$

These pairs of equations, either Eqns. (5) and (6), or Eqns. (7) and (8), in three variables, prescribe the visible portion of **C** for each θ. Hence, one can plot the visible regions of **C** as ranges of t as a function of θ—see Figure 3, for example. The left-hand side of each plot in Figure 3 shows a planar curve to be analyzed. The visibility chart is shown on the right, using a family of viewing directions from infinity. One viewing direction on each curve and each chart is also highlighted.

As the viewing direction changes, so does the visible portion of the curve. The visibility chart encodes the parts of the curve that are visible from each viewing direction.

Such visibility charts have several properties that are crucial to our discussion:

- For a C^1 continuous closed curve $\mathbf{C}(t)$, the visibility chart is periodic along its vertical axis t.

- For a periodic family of viewpoints parameterized as $\mathcal{V}(\theta)$, the visibility chart is periodic along its horizontal axis θ.
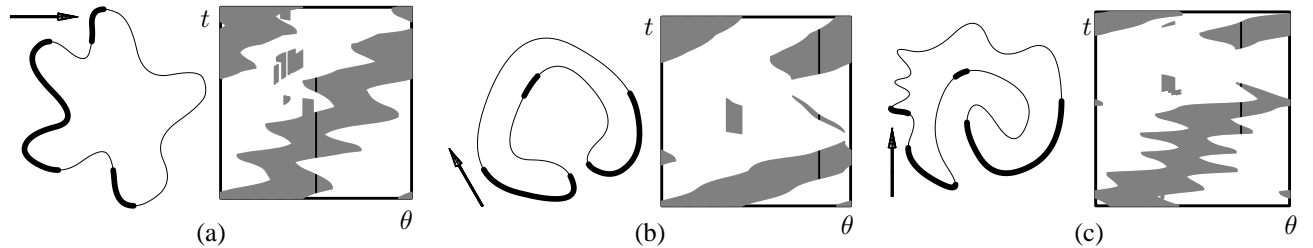
Figure 3. Visibility charts of free-form planar curves. The charts show the visible parameter ranges t of the curve in grey as a function of θ, which parameterizes the family of viewpoints (viewing directions from infinity in this figure). One specific viewing direction for each curve, from the left in (a) and from below in (b) and (c), is highlighted in black in these plots.

- A curve $\mathbf{C}(t)$ is completely visible externally if and only if there exists a connected path within the visible portion of the visibility chart from the bottom of the chart to the top. Stated differently, if there exists any t such that $\mathbf{C}(t)$ is not visible from *any* θ, the visibility chart is disconnected at that t level.

The fundamental question we will answer in the next section is how to find a minimal set of views that covers the entire domain of the curve. We show that, with the aid of visibility charts, this question can be reduced to the set-covering [9] problem over the reals.

3.3. The Minimal Set of Viewing Directions

We now wish to compute a *minimal set* of viewing directions, using the visibility chart.

Given a visibility chart, any vertical line for a particular θ spans some interval or intervals of t. We must thus choose a set of values of θ such that the union of these intervals in t covers the whole t domain; we wish to find a minimal set containing the fewest number of discrete θ values.

The discrete set-covering problem is NP-complete [9, p. 974] in the general case. A greedy algorithm, which at each step chooses a set that maximizes the number of newly-covered elements, provides a solution whose approximation factor (in terms of the number of covering sets) is proportional to the logarithm of the largest set [18, 20].

The situation is more favorable when the domain to be covered is an interval $\mathcal{I}$ on the real line, and the family of subsets of the domain consists of single real intervals too. It is well known that a greedy algorithm efficiently provides the optimal solution, working as follows. First, sort all intervals according to their right endpoints. Among all intervals that contain the left endpoint of $\mathcal{I}$ pick the interval $\mathcal{I}_1$ whose right endpoint is furthest to the right. Then, among all intervals that contain the right endpoint of $\mathcal{I}_1$ pick the interval $\mathcal{I}_2$ whose right endpoint is furthest to the right. At any stage, if the right endpoint of the last interval is not contained in any other interval, the algorithm halts and re-

turns "failure." Otherwise, the procedure terminates when the right endpoint of $\mathcal{I}$ is covered, returning the set of chosen intervals. The simplicity of the problem is not altered when the domain is cyclic, as in our case.

This algorithm requires $O(n \log n)$ time for n intervals. Clearly, the presorting takes $O(n \log n)$ time, and then there are $O(n)$ steps, each of which takes $O(\log n)$ time, using standard data structures for searching and updating the queue. In a sense, this running time is optimal. If we refine the problem so that a minimum number of intervals is required to cover the interval $\mathcal{I}$ or a maximum portion of it (in cases where the entire $\mathcal{I}$ cannot be covered), an easy reduction from the *sorting* problem shows a matching lower bound on the running time of the algorithm.

Figure 4 provides an example. In this case, three viewing directions are sufficient, as shown, both on the curve itself (Figure 4 left) and on the visibility chart (Figure 4 right). Note that due to the periodicity of the shape in t, two of the selected intervals (the dotted-dashed and the dashed) span the top ($\equiv$ bottom) border of $t = 1$ ($\equiv t = 0$). Only one interval in each view is used. The other, unused, intervals are marked by solid black. Horizontal arrows in the visibility chart shown in Figure 4 depict the interval $\mathcal{I}_{i+1}$ that contains the rightmost ($\equiv$ topmost in the visibility chart of Figure 4) endpoint of $\mathcal{I}_i$, so that the interval of $\mathcal{I}_{i+1}$ is farthest to the right ($\equiv$ top).

For many exterior visibility problems, assuming each subset has one interval and then solving the problem in almost linear time as above may be a reasonable trade-off between computation and quality of answer, if we are not too concerned about the possibility of finding a slightly suboptimal solution in terms of number of viewpoints. However, if an inspection chart were to be used many times, we might be prepared to expend more computation to correctly select a truly optimal set of viewpoints. This requires us to take into account that the part of the curve visible from a given viewpoint can be composed of multiple intervals; the resulting NP-complete general set-covering problem [19] takes exponential time to solve. (This may still be feasible if the number of viewpoints in the minimum set is fairly small: e.g., an
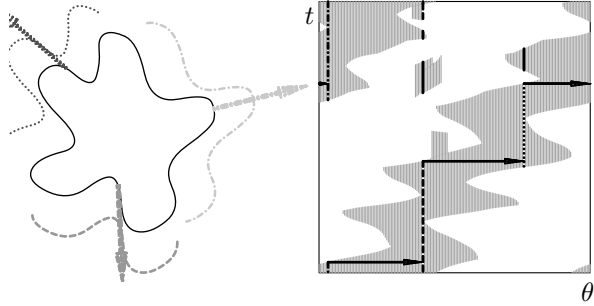
Figure 4. Searching the visibility chart using single intervals to find the minimal set of views covering a curve. Three views suffice.

exhaustive search for a 5-viewpoint solution over a few hundred views can take a few minutes on a modern PC.).

We are, therefore, offering two approaches to solve the visibility covering problem. If only a single interval in the domain of the curve is visible from some view location and/or direction, or if we are willing to consider only one such interval, even if more can exist, a simple, almost linear, solution has been shown to be tractable. We now consider the possibilities in the general, multi-interval case, from each view.

It is common to represent the set-covering problem using a graph in which each subset is a vertex and two vertices are connected by an edge if the intersection of the two respective subsets is not empty. There are several known results for instances of the covering problem based on properties of the respective graphs. For example, upper bounds on the size of the minimum cover are known for trees and triangle-free graphs [19], and for r-partite graphs [5]. Our graphs unfortunately do not possess any such special properties. Thus, to solve our covering problem we use a practical approach: we discretize the search space and perform an iterative search, where the iteration is on the cardinality of the covering set.

Firstly, however, we note that it is simple to test whether *any* finite set of directions is sufficient for viewing the complete 2D object, or whether the object has some parts that are not visible from any view (for example, given a set of viewpoints at infinity, it is clearly impossible to see all of the curve in Figure 2 due to its highly convoluted nature). It is important to carry out such a test to avoid exhaustively searching all of the exponentially-many subsets of viewpoints, as we seek solutions which progressively use more and more viewpoints. We perform this test as follows.

If some point $\mathbf{C}(t)$ on the object is not visible from *any* view, then a horizontal line in the visibility chart corresponding to the parameter value t does not intersect any gray region in the chart. We can quickly test if such horizontal lines exist. We simply compute the maximum and min-

imum values of t corresponding to each separate gray region in the visibility chart, giving an interval of t covered by that region. We compute the union of these intervals for all gray regions. If the union does not cover the entire t domain of the curve, then some points on the object are not visible from any direction. In such a case we report "failure" and terminate the algorithm. Examples are shown in Figures 3(b) and 3(c). While much of these two curves can be seen from suitable viewing directions from infinity, there are clear gaps corresponding to parts of the interior of the concavity in each case which are not visible from any outside view. Such parts correspond to two small gaps in Figure 3(b) and to one large gap in Figure 3(c).
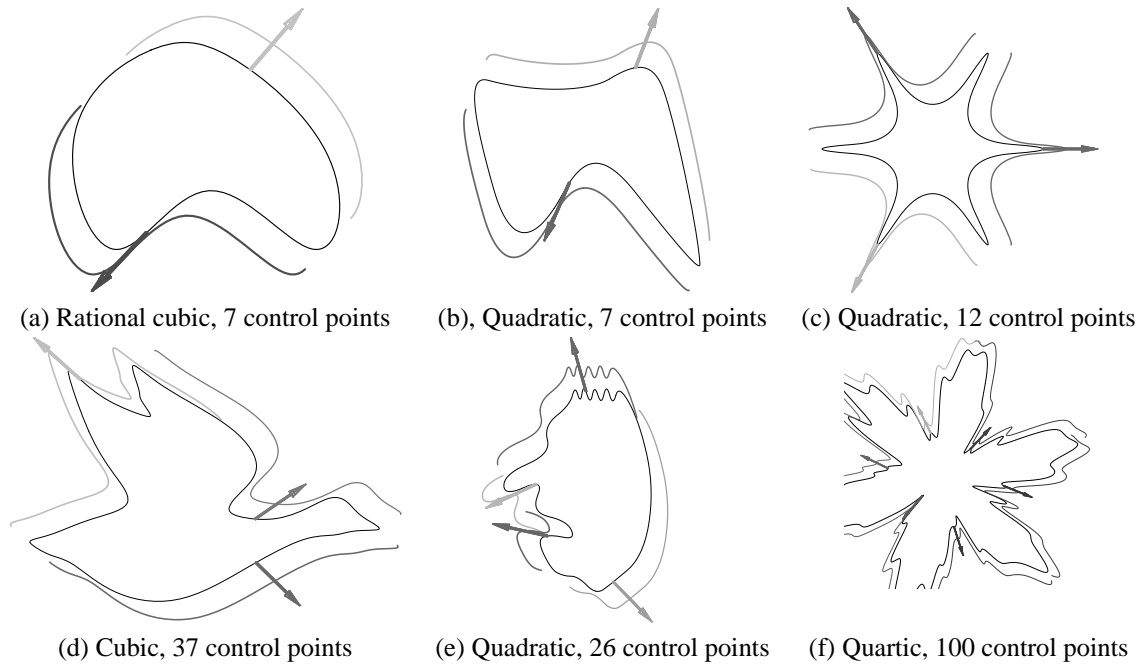
We now return to the case where all parts of the object *can* be seen from at least one view. We first discretize the visibility chart in θ at some desired resolution. Clearly, the higher the resolution, the more time the algorithm will take, while the lower the resolution, the higher the chance that we will find a suboptimal solution, and, for example, report that four viewing directions are needed when three will do.

The primitive object in our covering problem is one or more t-intervals of $\mathbf{C}$ (a so-called *slice*), as seen from one discrete viewing direction θ. We now perform a two-level iterative search. At the outer level of the search we enumerate the cardinality of the set of view directions, starting from $i = 2$ and going up. At the inner level of the ith outer level of the search we iterate over all possible sets of i distinct viewing directions. For each such set S we compute the union of the respective slices. The process stops when it finds a set whose union covers the entire parametric domain of the curve.

Directly implementing this in a straightforward way would be very time-consuming, especially if a fine angular resolution were used. We, therefore, use two further ideas which typically speed this process up:

1. Before the search commences, we compare each slice to the slices on either side of it. If the intervals of the curve in a slice are completely contained in the intervals of the curve in either neighbor, we can discard this slice: all of the curve that can be seen from this viewpoint, and more, can be seen from the neighboring viewpoint. This pruning process is repeated until no such subsumed slices are present. This helps to reduce the size of the search space.

2. Generally, viewpoints that see more of the curve are more useful in constructing solutions than those viewpoints that see a smaller part of the curve. Thus, we always give higher priority (that is, consider first in the search) viewing directions whose slices cover a larger amount of parameter t.

The latter heuristic is essentially a greedy approach, merely controling the *order* in which the algorithm consid-

(a) Rational cubic, 7 control points   (b), Quadratic, 7 control points   (c) Quadratic, 12 control points

(d) Cubic, 37 control points   (e) Quadratic, 26 control points   (f) Quartic, 100 control points

Figure 5. Several examples of planar curves and their decompositions into subcurves that can be seen using a minimal set of viewing directions.

ers the slices, in an attempt to find a solution more quickly. The theoretical complexity of the search is unchanged: it still has to consider an exponential number of cases in the worst case. In any case, we still find a solution if one definitely exists, as eventually all possibilities will be considered.

In order to be able to guarantee that we find a correct solution, given that we have discretized the visibility chart, we must be careful in our handling of the intervals. When two viewing directions are sufficient, these two views must be from antipodal directions, complementing each other. When three or more views are necessary, in all likelihood, the views will share overlapping regions of the curve. Hence, in practice, we expand the domain of each interval by a small portion when we conduct the single-interval set-covering search. If only two views are found necessary, we make sure these views are made precisely antipodal and then verify the solution with the original intervals. For three or more views, we verify the detected solution using the original intervals, expecting the overlaps between adjacent views to compensate for the added portion. Small perturbations of the viewpoints could also be applied, in either case.
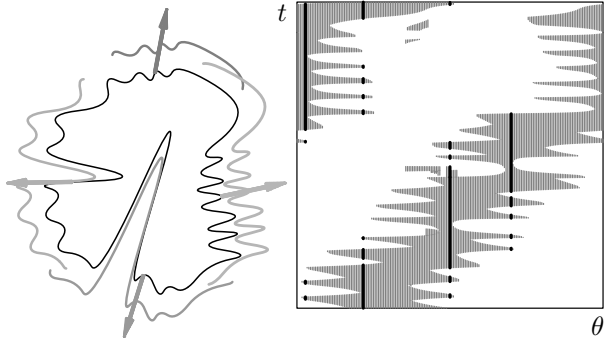
4. Examples

We now provide some further examples. Figure 5 shows several C^1-continuous planar curves of varying complex-

ity, decomposed according to the appropriate minimal sets of external viewing directions. These solve the so-called fortress problem where we have viewpoints located on the circle at infinity.

In Figures 5(a) and 5(b), two simple cases are presented, where two (antipodal) viewing directions are sufficient to cover the entire curve. Figures 5(c) and 5(d) present two cases where three viewing directions are necessary and sufficient, whereas in Figure 5(e) four views are required. While the curve in Figure 5(e) requires four views, these views are not monotone with respect to each other around the shape. This nonmonotonicity has no effect if the application of the algorithm is inspection. Nonetheless, it would make mold-assembly design for this object more complex. Figure 5(f) is a highly complex quartic curve with 100 control points that requires six viewing directions to cover the shape. All these viewing directions were computed using the visibility charts and the single-interval set-covering solution in slightly worse than linear complexity in the number of discrete viewing directions sampled.

The examples in Figure 5 were computed on a modern PC using between 72 and 720 viewing direction samples, with 72 for Figures 5(a) and 5(b), and 720 for Figure 5(f). The entire computation took from a fraction of a second in simple cases such as Figures 5(a) and 5(b) to a little over a minute for the most complex case of Figure 5(f).

Figure 6 presents the visibility chart of another shape. Four views are necessary to cover the entire curve in this

Figure 6. Covering set example with the visibility chart. Four views are necessary and are shown on both the curve regions (left) and the chart (right).

case. The four views are marked on both the curve itself and on the visibility chart. The union of the four intervals covers the entire domain of the curve. Note that again we only consider the central, largest interval in each view, in our set-covering search for efficiency.

As already mentioned, we are able to place the viewpoints along any univariate path. Specifically, we may place them along the interior walls of the curve, building a visibility chart which allows us to solve the (interior) art gallery problem in the continuous domain. Figure 7 shows two such examples where the viewpoints are placed on the curve itself, using $\mathcal{V}(\theta) = \mathcal{V}(t)$, and we restrict viewing to only consider directions pointing into the curve's interior. Note that these visibility charts are symmetric along the diagonal as they portray which curve parameters are visible from which other curve parameters.

An interesting observation can be made by considering Figure 7(a). The viewpoint marked is at a concave curve location and from that point nothing can be seen in the local neighborhood. That is, if $\mathbf{C}(t_0)$ is a point in a concave region, there exists a small $\delta > 0$ neighborhood of $\mathbf{C}(t_0)$ such that the region $[t_0 - \delta, t_0 + \delta] \setminus \{t_0\}$ is invisible to a guard at $\mathbf{C}(t_0)$! In contrast (see Figure 7(b)), if the view location is placed at a convex region $\mathbf{C}(t_1)$, there is a small $\varepsilon > 0$ neighborhood such that the region $[t_1 - \varepsilon, t_1 + \varepsilon]$ is visible.

Unfortunately, the single-region set-covering solution cannot be used to resolve the art gallery problem. The general set-covering solution must be employed, with an exponential time complexity in the number of guards allowed. Figure 8 presents several results of our automatic solution to the art gallery problem for smooth and continuous closed shapes. The original curve is drawn in thick gray whereas the visible region for each guard is differently offset and differently dotted/dashed. The solution for this general set-covering problem is exponential in the number of views ex-

amined. Here, one hundred different views were used to find the (two or three) proper guard positions, in less than a second, on a modern PC, in Figure 8(a) to 8(c). Figures 8(d) and 8(e) are more involved and again using one hundred different views, the positions of the five necessary guards were found in less than a minute.

5. Conclusions and Future Work

We have presented a two-dimensional structure called a visibility chart, and have shown how it can be used in algorithms for choosing a minimal, or nearly minimal, set of viewing directions for inspection of an object whose boundary is a tangent-continuous curve. The algorithms use a set-covering search over a discrete set of intervals to compute the desired viewing directions. We have also demonstrated practical examples showing the solutions we found.

Our tests have shown that, for the fortress problem, a single-interval set-covering solution in slightly worse than linear time can often produce good results quickly. However, the single-interval approach is only an approximation and one could envisage cases where allowing multiple intervals from each view could reduce the number of views necessary. In other cases, such as the art gallery problem, a general NP-complete set-covering algorithm must be used to find the optimal solution; this might also be appropriate where an inspection plan is to be used many times, and the time savings during use may warrant an expensive search for a minimal solution.

Note that for the purpose of mold design, the part of the curve seen from each viewing direction *must* be connected, and hence the single-interval approach is the proper one to use, whereas for inspection, multiple intervals from a single viewpoint are admissible. See, for example, the highlighted view in Figure 3(a) with three disjoint visible regions.

When determining the visibility of each curve region, during the construction of the visibility chart, we cast rays toward each segment (see Figure 1(d)). In [11], it was suggested that since the adjacent region possesses different quantitative invisibility, once can employ adjacency coherence and reduce the number of rays we are required to cast. If a region is visible, its two neighbors must be invisible (but not necessarily the reverse). A visibility propagation tool, similar to that of [11] could be utilised here as well.

The algorithms presented sample the visible region at a discrete set of locations along a univariate function as $\mathcal{V}(\theta)$ or as $\mathcal{V}(t)$ along the curve itself. Needless to say, this one-parameter family of viewpoints could follow any univariate path in the plane. A nontrivial extension would also support visibility charts of viewpoints that are defined using a two-parameter family of views. Let (θ, r) span the plane using polar coordinates. The computed visibility chart will be a volume which prescribes the visibility at curve parameter t as a function of θ and r. Then, we could also consider, for
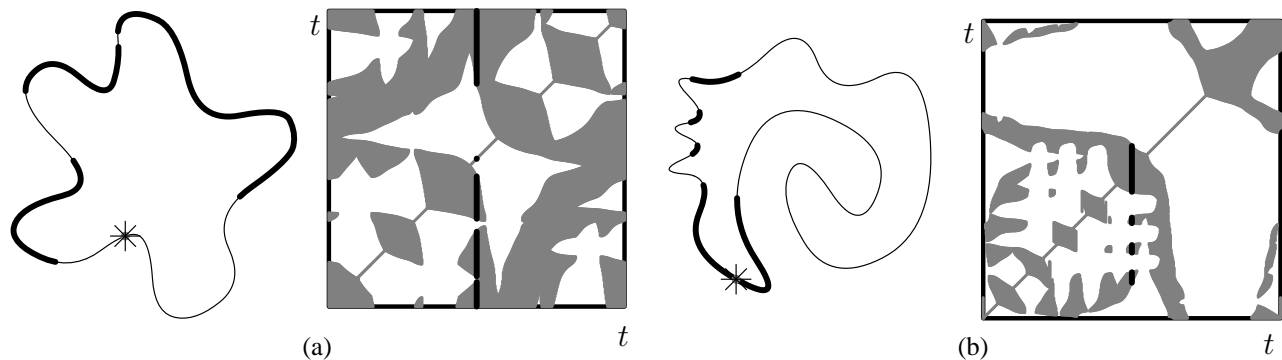
Figure 7. The art gallery problem. Visibility charts for viewpoints along the walls of the shape itself are presented. One viewpoint, is highlighted with the visible region on the curve and in the chart presented in black. Note the view is limited to inspection from inside the shape.

example, guards in the interior of the art gallery and not just along its walls. Similarly, the extension of the presented solution to surfaces in R^3 is highly desirable but also highly complex.

Some manual control over the precise split regions might aid the end-user in his or her design and allow other constraints to be imposed. In particular, for any case involving more than two views, the regions visible from adjacent viewpoints are likely to overlap. The precise location of the split points along the overlapping region could be decided by the end-user, taking into account other constraints, such as distance from the guard, best relief angles in the mold design, etc. Alternatively, the end-user could be allowed to make minor adjustments to the viewpoints used from the viewpoint family, again as long as the shifted region for that viewpoint continues to overlap those of its neighbors and no topological changes occur.

Clearly, a desirable extension to the method presented is to cover the case in which the curve is not tangent continuous, but may have corners. It should be a relatively straightforward addition to include corners as silhouette points for appropriate *ranges* of viewpoints (or directions). These C^1 discontinuities should be analyzed to identify the span of directions from which these singularities are also silhouette points, based on the incoming and outgoing tangents, at each discontinuity.

In addition, certain problems require 3D visibility computations rather than 2D ones, and we intend to examine the extension from 2D to 3D of the ideas presented here.

6. Acknowledgements

References

[1] H.-K. AHN, M. DE BERG, P. BOSE, S. CHENG, D. HALPERIN, J. MATOUŠEK, AND O. CHEONG, Separating an object from its cast, *Computer-Aided Design*, 34 (2002), 547–559.

[2] A. APPEL, The notion of quantitative invisibility and the machine rendering of solids, *Proc. ACM national Conference*, Washington, DC, 387–393, 1967.

[3] J. BOWN, *Injection Molding of Plastic Components*, McGraw-Hill, 1979.

[4] K. BOWYER AND C. DYER, Aspect graphs: An introduction and survey of recent results, *Proc. SPIE Conf. on Close Range Photogrammetry Meets Machine Vision*, 1395, 200–208, 1990.

[5] M. CHEN AND G.J. CHANG, Total interval numbers of complete r-partite graphs, *Discrete Applied Mathematics*, 122 (2002), 83–92.

[6] L. CHEN AND T. WOO, Computational geometry on the sphere with application to automated machining, *ASME J. of Mechanical Design*, 114 (1992), 114, 288–295.

[7] L. CHEN, S. CHOU, AND T. WOO, Parting directions for mold and die design, *Computer-Aided Design*, 25 (1993), 762–768.

[8] C. CONNOLLEY, The determination of next best views, *Proc. IEEE Int. Conf. on Robotics and Automation*, 432–435, 1985.

[9] T.H. CORMEN, C.E. LEISERSON, AND R.L. RIVEST, *Introduction to Algorithms*, MIT Press and McGraw-Hill, 1990.

[10] S. DHALIWAL, S.K. GUPTA, J. HUANG, AND M. KUMAR, A feature-based approach to automated design of multi-piece sacrificial molds, *J. of Computing and Information Science in Engineering*, 1 (2001), 225–234.
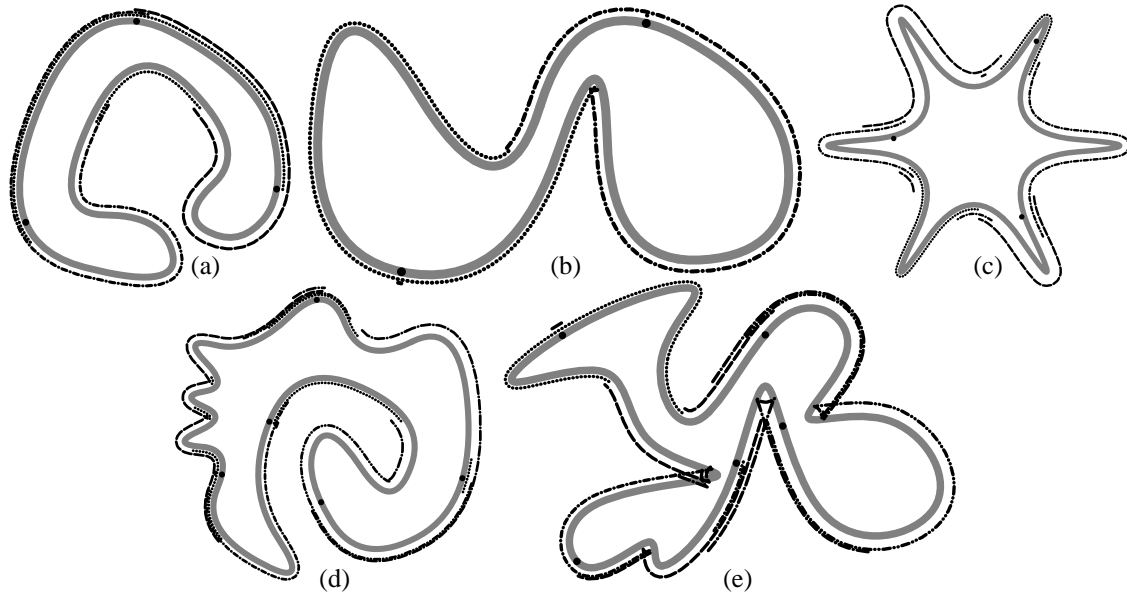
Figure 8. Art gallery solutions, derived with the aid of visibility charts for viewpoints along the walls of each shape. The original shape is drawn in gray while the different visible regions for each guard are differently offset and differently dotted/dashed.

[11] G. ELBER AND E. COHEN, Hidden curve removal for free form surfaces, *Computer Graphics (Proc. SIGGRAPH)*, 24 (1990), 95–104.

[12] G. ELBER AND E. COHEN, Arbitrarily precise computation of Gauss maps and visibility sets for freeform surfaces, *Proc. 3rd ACM/IEEE Symp. on Solid Modeling and Applications*, Salt Lake City, UT, 271–279, May 1995.

[13] G. ELBER AND E. ZUSSMAN, Cone visibility decomposition of freeform surfaces, *Computer-Aided Design*, 30 (1998), 315–320.

[14] G. ELBER AND M.-S. KIM, Geometric constraint solver using multivariate rational spline functions, *Proc. 6th ACM/IEEE Symp. on Solid Modeling and Applications*, Ann Arbor, MI, 1–10, June 2001.

[15] G. ELBER, X. CHEN, AND E. COHEN, Mold accessibility via Gauss map analysis, *Proc. Shape Modeling International*, Genova, Italy, 263–274, June 2004, to appear in *J. of Computing & Information Science in Engineering*.

[16] R. ELLIOTT, *Cast Iron Technology*, Butterworths, London, UK, 1988.

[17] K. HUI AND S. TAN, Mold design with sweep operations— A heuristic search approach, *Computer-Aided Design*, 24 (1992), 81–91.

[18] D.S. JOHNSON, Approximation algorithms for combinatorial problems, *J. of Computer and System Sciences*, 9 (1974), 256–278.

[19] T.M. KRATZKE AND D.B. WEST, The total interval number of a graph, II: Trees and complexity, *SIAM J. on Discrete Mathematics*, 9 (1996), 339–348.

[20] L. LOVÁSZ, On the ratio of optimal integral and fractional covers, *Discrete Mathematics*, 13 (1975), 383–390.

[21] A.D. MARSHALL AND R.R. MARTIN, *Computer Vision, Inspection and Models*, World Scientific, Singapore, 1992.

[22] D.J. MEDEIROS AND S. KWEON, Part orientations for CMM inspection using dimensioned visibility maps, *Computer-Aided Design*, 30 (1998), 741–749.

[23] J. O'ROURKE, *Art Gallery Theorems and Algorithms*, Oxford University Press, Oxford, 1987.

[24] M. POCCHIOLA AND G. VEGTER, The visibility complex. *Int. J. of Computational Geometry and Applications*, 6 (1996), 279–308.

[25] D.R. ROBERTS AND A.D. MARSHALL, Viewpoint selection for complete surface coverage of three dimensional objects, *Proc. British Machine Vision Conference* (P.H. Lewis and M.S. Nixon, eds.), Southampton, England, 2, 740–750, 1998.

[26] W.R. SCOTT, G. ROTH, AND J.-F. RIVEST, View planning for automated three-dimensional object reconstruction and inspection, *ACM Computing Surveys*, 35 (2003), 64–96.

[27] T. SMITH AND R. FAROUKI, Gauss map computation for free-form surfaces, *Computer-Aided Geometric Design*, 18 (2001), 831–850.

[28] K. TARABANIS, R. TSAI, AND A. KAUL, Computing occlusion-free viewpoints, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18 (1996), 279–292.

[29] G. TARBOX AND S. GOTTSCHLICH, Planning for complete sensor coverage in inspection, *Computer Vision and Image Understanding*, 61 (1995), 84–111.

[30] T. WOO, Visibility maps and spherical algorithms, *Computer-Aided Design*, 26 (1994), 6–16.