Geometric Covering

Nadav Shragai, Gershon Elber

Computer Science Department
Technion, IIT. 32000 Haifa, Israel

Abstract

In this work, we present a single unified framework that can solve many geometric covering queries such as inspection and mold-design. The suggested framework reduces a geometric covering query to the classic computer science set-covering problem. The solution is of exponential complexity due to the inherent complexity of the classic set-covering problem. However, in practice, we are able to efficiently offer almost optimal solutions for small scale problems of several covering entities. Finally, using the portrayed framework, we demonstrate some results on the mold-design problem in manufacturing.

Keywords: Set-cover, Mold-design, Security, Inspection.

1. Introduction

Covering problems are fundamental questions in numerous disciplines. The need to find a minimal (optimal) set of elements that provide a covering for a given universe emerges in many scientific problems. Interestingly enough, it is closely related to many geometric questions as well. In mold-design, the typical aim is to find the minimal number of mold parts that, when assembled, form a complete mold into which to inject the model. This problem is also related to the 2D *fortress problem* [1] where guards are to be placed around a fortress, making sure no one escapes in or out of the fortress without noticing. If the guards are at infinity (i.e. orthographic projection), the fortress region that each guard inspects is, in essence, defining one (not necessarily connected) mold piece.

The 2D fortress and art-gallery problems have clear immediate needs in security applications once extended to 2.5D terrain. Consider some compound whose fences must all be continuously inspected. The optimal placement of guards in the given compound's terrain is another reflection of geometric covering. As yet another example, consider a shopping center or alternatively an airport field that must be inspected all the time in all public (and possibly restricted) areas by a minimal set of guards[1].

Covering problems in geometry can have many variations. Covering by itself can mean different things in different disciplines. For example, in inspection, all points in the inspected model must be visible to at least one camera. In mold-design, every point of the input model must be included in exactly one mold part. A guard can be stationary or can be moving along a prescribed path, in which case a single *inspection location* is a curve-segment. The guards or the inspection locations can be selected from a finite set of points, or an infinite set. That is, each inspection location can be a finite portion of a curve or even a surface, having infinitely many points.

One may require that any point in the region to be covered, $\mathcal{C}$, be inspected by at least a single guard. Alternatively, one may require some level of redundancy, in which case every point in $\mathcal{C}$ is inspected by more than one guard:

Definition 1. *Consider the region to be covered, $\mathcal{C}$. If every location in $\mathcal{C}$ is to be covered by at least $k > 0$ guards, we denote that problem a k-*covering *problem.*

In this work, we present a simple framework that can address the above problem and more. The framework is indeed of exponential time complexity in the number of guards, following the time complexity of many existing similar (yet application specific) solutions due to the inherent complexity of *set-cover* [2].

Consider a set of elements $\mathcal{C}$ and let $\boldsymbol{\mathcal{I}}$ be a set of subsets of $\mathcal{C}$. Solving the set-cover for $\mathcal{C}$, one seeks the union of minimal cardinality of elements in $\boldsymbol{\mathcal{I}}$ that covers $\mathcal{C}$, if possible. In our context, we can now define geometric covering:

Definition 2. *Let $\mathcal{S}$ be some scene in $\mathbb{R}^{n_1}$, and let $\mathcal{C} \subset \mathcal{S}$ be the region requiring covering. Let $\boldsymbol{\mathcal{I}} \subset \mathbb{R}^{n_2}$ (n_2 not necessarily equal to n_1) be a set of possible inspection locations and let $\mathbf{I} \subset \boldsymbol{\mathcal{I}}$ be a finite subset of all possible inspection locations such that the union of elements in*

[1] Hence after and unless otherwise stated, we will use the term *guards* to denote human guards, cameras or similar sensing device, and also an orthographic view for mold-design.

$\mathbf{I}$ *k-covers $\mathcal{C}$. The solution of the* Geometric Covering Problem *(GCP) is defined as* $\mathbf{I}$ *of minimal size.*

It should be noted that in this work the union is not a simple Boolean set operation but rather a function that can adhere to the k-covering requirement. In other words, the union function is no longer Boolean and it must count the number of guards covering each point of $\mathcal{C}$. A careful examination of all the examples we portrayed above shows that Definition 2 subsumes them all. Further, Definition 2 leaves room for the geometric covering variations already discussed. For example, $\boldsymbol{\mathcal{I}}$ can be finite or can be an infinite set of potential inspection locations. Moreover, $\boldsymbol{\mathcal{I}}$ prescribes inspection locations and not necessarily points of inspection as, for example, each inspection location can define a moving camera along some predefined parameters. Finally, note that the dimensions of $\mathcal{S}$ and $\boldsymbol{\mathcal{I}}$ can be arbitrary.

So far, we imposed no restrictions on the type of the input geometry which can, in fact, be spline surfaces, polygonal meshes, or other geometric representations. However, given a GCP, as a prerequisite to apply the framework proposed in this work, we do require a parameterization for the region $\mathcal{C}$ requiring covering. If $\mathcal{C}$ is a curve it must be parameterized as $\mathcal{C}(t)$. If $\mathcal{C}$ is a (trimmed) surface (or a set of surfaces) or even a trivariate, it better be parameterized as $\mathcal{C}(u, v)$ or $\mathcal{C}(u, v, w)$, etc. While clearly a GCP can have a continuous domain $\mathcal{C}$ to be covered and a continuous inspecting domain, $\boldsymbol{\mathcal{I}}$, in this work, these domains are discretized and the framework we propose solves a discrete approximation to the problem.

During the early days of computer graphics, quite a few algorithms were developed to produce scenes with hidden lines/surfaces removed. These algorithms worked at the geometric primitives' level, like polygons or surfaces, and suffered from high complexity, instability, and improper handling of special cases. The Z-buffer, when introduced, offered a discrete approximation that is very robust, simple, and general. The framework we present here follows a similar trend and reduces the geometric covering analysis to the pixel level, yielding a simple and general framework that we similarly believe can be made robust.

The rest of this work is organized as follows. In Section 2, we briefly discuss the state-of-the-art in mold-design. The proposed framework is presented in Section 3, and examples for mold-design analysis are presented in Section 4. Further possible extensions are discussed in Section 5, and finally we conclude in Section 6.

2. Previous Work

A large body of work can be found on specific solutions of GCP for mold-design and inspection. We now briefly discuss the state-of-the-art in this field.

The problem of mold-design in 2D was addressed by many. 2D queries for partitioning planar curves into 2D mold parts and/or regions covered by guards are, for example, discussed in [3, 4, 5]. [3, 4] find all the directions in which a 2D polygon/free form can be separated into two opposite directions in a linear time. [5] finds a minimal number of directions from which a C^1 continuous curve can be completely inspected/molded.

Some publications discussed the simpler question of parting a 3D model into two opposite directions such as [10], which gave necessary and sufficient conditions for how to find a feasible separation direction and whether a separation exists when a separation direction is already given or such as [9] that worked with NURBS surfaces by classifying the topology of their silhouettes into equivalent classes. While in this work we handle the general separation problem of arbitrary number of mold parts, some interesting attempts were made concerning this simpler two opposite directions question. [6, 7, 13] all handled the separation of a 3D model into the two main parts, while also considering some small side parts. Yet the two main mold parts still required opposite direction separation. Some attempts were also aimed at handling only certain classes of objects such as 3D surfaces of extrusion [11] or solids of revolution [12] by exploiting their special properties. Some publications has argued that the constraint of parting direction to allow smooth extraction of the mold isn't enough. Other constraints should be considered as well such as providing room for an extraction angle [14], small drawing depth (the distance to extract the part before it is cleared) [15, 16], flatness of parting surface [15, 16], small surface area of undercuts [15] etc. Some of those aspects are also considered in Section 5, in this work's context. Finally, [8] presented an accelerated algorithm using GPU, thus opening a way for accelerating other algorithms such as the one discussed in this article and again, we also address GPU exploitation in Section 5.

Several results such as [17, 18, 19] also addressed the more complex problem of partitioning a 3D model into a minimal set of mold parts and hence are closer to this work. [17] first calculates all the directions from which each face of a polyhedron is fully visible, and then suggests possible view directions to cover the polyhedron. A set-cover over those views is solved in an exponential time. A main drawback that is revealed in [17] is the requirement to cover entire faces by single views. The work presented herein, is based on neither a-priori partitioning of the input model nor a-priori constraints such as visibility of whole faces by single views. [18] considers concave regions in polyhedron and finds, for each region, all the directions that inspect the entire region. These regions may be partitioned into several faces if no view direction exists that fully covers the region. A set of possible view directions is suggested, which together with the set of concave regions and faces are used as input to a set-cover. As in [17], the weakness of [18] is its

requirement to cover entire faces by single views. Furthermore, in some cases, [18] requires the covering of an entire concave region by a single view direction, as can be seen in Figure 1. [19] handles the problem of separating a polyhedron into a minimal set of mold parts as an optimization problem. [19] reformulates the visibility problem and the minimum number of parting directions as a variation of linear programming. The polyhedron is embedded in the constraints as a list of faces while the requirement to cover entire faces by single views, persists.

[20, 21, 22, 23] do not solve the problem of covering a 3D model, yet their examination of visibility is relevant. [20] shows that the view directions of a surface S, that are occluded by another surface O can be calculated as the Minkowski sum of O and $-S$. Furthermore, [20] proves that only a subset of the Minkowski sum is required. The view directions of S that are occluded by O are an essential part of the visibility calculation. Therefore, the use of Minkowski sum significantly reduces the calculation time. [23] solves the visibility problem simultaneously for all view directions, using three equations with six variables. One of the constraints is an inequality, therefore the zeroes of the formulas are four-manifolds in a six-dimensional space. The result is a function of four parameters that prescribes the visibility of every point (u, v) on the surface, from a hemisphere of possible directions above the tangent plane, parameterized using spherical coordinates (θ, ϕ).

Some works such as [24] addressed the use of *visibility graph* as a way to solve certain guarding problems. An edge exists in the visibility graph between two nodes when the nodes can see each other. In this article, however, we require the inspection of complete (3D) surfaces, not necessarily with single views. Such a requirement is hard to accomplish using visibility graphs.

This work does not require the visible portion from any view to be connected, as might be required in some cases when considering a 3D mold model. One should note that mold parts that are disconnected on the surface of the model can still be connected in three-space and hence, we enforce no such constraints. In this work, we introduce an approach different than known previous work in geometric covering. Rather than conducting the geometric analysis (and the set-cover computation) in the Euclidean space, we always reduce the GCP into handling canonical one-dimensional vectors, in pixels level analysis, using the parametrization of $\mathcal{C}$, and perform the set-cover analysis over these linear vectors, yielding a general framework for solving GCP's. This introduced framework is now described.

3. Algorithmic Framework

A discrete framework that can handle any GCP where the region to be covered is parameterized, following Definition 2, is now presented. Let $\mathcal{I}_i \in \boldsymbol{\mathcal{I}}$, $i = 1, ..., n$ be n potential inspection locations, and for now assume these are inspection points. Further and for the sake of our example, let the region to be covered be a single parametric surface $\mathcal{C}(u, v)$. For each inspection point $\mathcal{I}_i$, create a 2D binary visibility map, $\mathcal{V}_i(u, v)$ as follows: $\mathcal{V}_i(u_0, v_0)$ is one (visible) if and only if $\mathcal{C}(u_0, v_0)$ is visible to $\mathcal{I}_i$ and zero (invisible) otherwise. Having $\mathcal{V}_i(u, v)$, $i = 1, ..., n$ as 2D binary maps, surface $\mathcal{C}(u, v)$ will be properly k-covered if the union of the subset selected out of these n visibility maps k-covers (the entire parametric domain of) $\mathcal{C}(u, v)$.

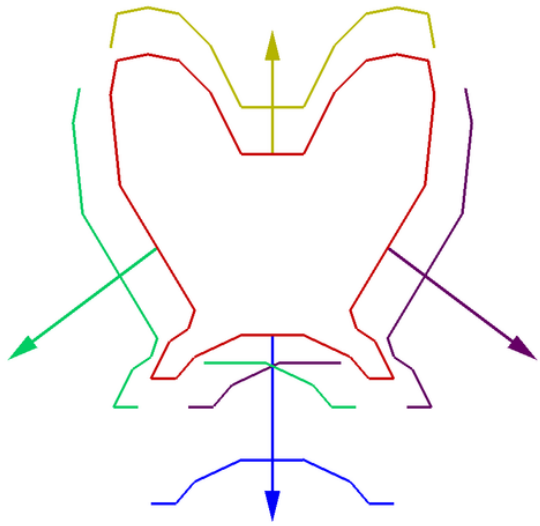

Figure 1: A shape and the parts seen from some directions. From the purple and green directions note the visible portion is disconnected. Yet, the shape can be seen entirely by the green, yellow and purple observation directions. However, [18] will require the blue observation direction as well since the bottom cavity would not be partitioned.

While the parametric domain of $\mathcal{C}(u, v)$ is continuous, we approximate $\mathcal{V}_i(u, v)$ discretely as a 2D binary image, using a two-step Z-buffer rendering algorithm from view point $\mathcal{I}_i$, as in Algorithm 1.

Line 1.2 in Algorithm 1 performs the first regular Z-buffer pass Scan Conversion [25] in Euclidean space whereas the rest of the algorithm performs the second UV parametric space pass. The u_j, v_j values as well as the x_j, y_j, z_j values of pixel p_j are derived in Lines 1.6 and 1.7 of Algorithm 1 using the parametric space (UV) as well as the Euclidean (XYZ) coordinates of the three vertices of T_l, as are computed by the scan conversion process in the (u, v) plane. In Line 1.8, the approximation symbol is used because the Z-buffer algorithm is inherently only a discrete approximation algorithm.

Now, given n such visibility maps, $\mathcal{V}_i(u, v)$, $i = 1, ..., n$ as 2D binary images, we serialize each image into a vector of bits. Let the visibility map images be of size $(q \times q)$. Then, the end result are n 1D visibility map vectors, each of which of size q^2. A classic set-cover algorithm is then applied to these n vectors to find a minimal covering set of all the q^2 pixels. That is, a minimal size subset

Algorithm 1: BUILDING $\mathcal{V}_i(u,v)$ FOR $\mathcal{C}(u,v)$

input : A Surface $\mathcal{C}(u,v)$;
A view point $\mathcal{I}_i$;
output: A 2D visibility map $\mathcal{V}_i(u,v)$ as a binary discrete approximation image;

1.1 Tessellate $\mathcal{C}(u,v)$ into triangles $\mathcal{T} = \{T_l\}_{l=0}^{N}$;
1.2 $\mathcal{Z}_{buf} \leftarrow$ Z-buffer Scan Conversion of $\mathcal{C}(u,v)$, using $\mathcal{T}$, from $\mathcal{I}_i$, keeping only depth information;
1.3 **for** *each triangle* T_l *in* $\mathcal{T}$ **do**
1.4 Scan convert T_l in (u,v) space using the UV parameterization's coordinates:
1.5 **for** *each pixel* p_j *in* T_l, *in the* (u,v) *plane* **do**
1.6 $u_j,\ v_j \leftarrow UV$ coordinates of p_j;
1.7 $x_j,\ y_j,\ z_j \leftarrow XYZ$ coordinates of p_j;
1.8 **if** $z_j \approx \mathcal{Z}_{buf}(x_j, y_j)$ **then**
1.9 $\mathcal{V}(u_j, v_j) \leftarrow$ visible;
1.10 **else**
1.11 $\mathcal{V}(u_j, v_j) \leftarrow$ invisible;

of these n vectors, such that the union of those vectors renders every pixel visible from at least k $\mathcal{I}_i$. The union function in this work treats each binary vector as an integer vector of 0's and 1's and sums together elements of the vectors in the candidate covering set . If each element in the sum is at least k then the selected inspection locations are indeed k-covering the region.

The search of the minimal covering set is an NP-Hard problem [2] and therefore of exponential time complexity in the number of guards. Hence, in practice, one can only hope for optimal solutions for cases of a few guards/mold pieces. While research on approximated solutions to the set-cover problem is beyond this work, there exist heuristic algorithms that, under certain conditions, can be used to approximate the solution more efficiently.

One observation that should be made here is that many entries in this configuration are going to be identical. Having q^2 pixels in a visibility map, q^2 is going to be in the millions and typically $n << q^2$. Consider one pixel in all n visibility maps, $\mathcal{V}_i(u_0, v_0)$, $i = 1, ..., n$. The n visibility maps create a vector of n-bits for pixel (u_0, v_0). This vector prescribes the visibility of pixel (u_0, v_0) from all n view points. Now consider a similar n-bit vector for another pixel (u_1, v_1). These two n-bit vectors might be identical. While, in theory, n bits can synthesize 2^n different binary vectors, neighboring points in $\mathcal{C}$ are likely to have similar visibility behavior and hence similar n-bit visibility vectors.

As a direct consequence of this observation, n-bit vectors that are identical can be collapsed, resulting in a smaller number of n-bit vectors, as a pre-processing step before the set-cover is computed. Section 4 provides some reduction statistics on actual examples.

The algorithmic approach portrayed so far can be also adjusted to allow variations:

1. The visibility map of a trimmed surface may contain locations which are trimmed away. The visibility map of a polygonal input may contain locations that are not mapped to any polygon at all. In both cases those locations are marked as "don't-care". These "don't care" locations are completely removed during the pre-processing serializing step.
2. Given an input mold in the form of several, possibly trimmed, surfaces, visibility maps can be computed for the individual surfaces and a 1D visibility vector can be formed by serializing the domains of the individual surfaces one after the other. Then, similar vectors can be collapsed and the set-cover will be applied to these global visibility maps, as before.
3. The GCP can be handled in any dimension. Clearly $\mathcal{C}$ can be a 1D curve in the plane, but also a 3D volume to be covered in $\mathbb{R}^4$. This, as long as the visibility problem can be resolved for the given inspection locations and the parametric domain can be serialized into a 1D vector. A region $\mathcal{C}$ that is a $p-manifold$ has a parameterization in p variables and is expected to have a domain that is a (subset of, or trimmed) p dimensional box. Having a discrete representation as a p dimensional image, each pixel in the image is tagged as 'visible' or 'invisible'. One can clearly serialize that p-dimensional (sub) box into a 1D vector and hence the set-cover solver can be equally applied herein as well.
4. The inspecting domain $\boldsymbol{\mathcal{I}}$ can also be of arbitrary dimension. For the mold-design problem in $\mathbb{R}^3$, $\boldsymbol{\mathcal{I}}$ can consist of all vectors of the unit sphere, considering all possible viewing directions. However, for terrain inspection, $\boldsymbol{\mathcal{I}}$ can also be in the form of positioning information as a 1D curve circumventing the compound, the 2D region of the entire compound, or the 3D region above the compound. Because the proposed solution is discrete, we can sample viewing locations from an arbitrary r-manifold inspection region, $\boldsymbol{\mathcal{I}}$, only to compute the visibility map from these discrete locations.

After having portrayed a general framework for handling discrete approximated solutions of GCP's, in the next section we present our implementation and results for the application of mold-design.

4. Results

In this section, we present several examples of mold-design application using the presented framework. All the examples and timings presented were using software implementation of the algorithms with the aid of the IRIT modeling environment [26]. The set-cover problem was solved using both an exhaustive search and a simple greedy algorithm for comparison. In the greedy search

we added in each step the view with the largest addition to the cover which is different from other implementations such as the one used by [18] where the number of added elements are considered and not their actual impact on the total cover. In all the examples presented in this section, $\boldsymbol{\mathcal{I}}$ consists of 130 potential inspection directions, views that are spread uniformly on the unit sphere [27], augmented with the antipodals of those directions and the six views of $\pm x, \pm y, \pm z$, 266 inspection directions in all. An a priori analysis of the geometry of the input could be used in order to identify unique additional directions in the input model, such as deep holes, which could be added to the set of inspection directions, alleviating chances of incomplete coverage.

In all presented examples, binary visibility maps of size 4096×4096 are used, where black denotes an 'invisible' location, white denotes a 'visible' location, and green marks a "don't-care". Again, a don't-care bit means that this specific location is trimmed away, in that surface. Also presented in all examples, is a final pair of images in Euclidean and in parametric space. The final view of the Euclidean space shows a general view of the model with arrows that present the covering viewing set. The final view of the parametric space shows the union of the binary visibility maps. Brighter gray level depicting more views that cover that specific pixel (while black denotes a pixel that is covered by no one).

Table 1 provides some additional statistics on all the presented results. The percentage of coverage which appears in all the examples of this work including Table 1 reflects the ratios between visible pixels and the total number of pixels (i.e. the sum of white and gray pixels divided by white, gray and black pixels, while ignoring don't-care pixels). One immediate observation that can be made from the results presented in Table 1 is that the 4096^2 pixels are vector-collapsed to around a hundred-thousand different (collapsed) vectors as part of the preprocessing stages in all presented examples. Another observation is the difference between the exhaustive and the greedy set-cover search. The former presents better results whereas the latter doesn't achieve as good results but requires significantly less processing time.

In Figure 2, a sphere with four depressions, $\mathcal{C}$, is presented. Not surprisingly, four views are required to cover it all and as can be seen in Figure 2, a coverage of 99.997% is achieved using an exhaustive search of four views. The greedy algorithm as can be seen in Table 1 achieves over 99% as well, yet not as good a result as the exhaustive search. The comb-like shapes in the four bottom parametric domains are due to the fact that the surface becomes singular at these points and degenerate triangles are purged away.

Figure 3 presents a model of a cup, $\mathcal{C}$. The first two views for the greedy search, (a) and (b), inspect the cup from the top and bottom and are able to capture over 95% of the model. However, the region of the body below the handles is not captured and additional views are required, (c) and (d), to achieve 99.827% of the coverage. Still, those regions aren't fully covered. The exhaustive search as seen in Table 1 achieves an even higher 99.995% coverage overcoming this deficiency. It is also shown in Table 1 that an exhaustive search of three views achieves almost as good result as well, while the best three views in the greedy search are far behind. Note that in this example of the cup, the visibility maps of the handles have disconnected components at times, a case this work allows.

The third example, in Figures 4 and 5, is of the famous Utah teapot, $\mathcal{C}$, with its interior added. This interior presents negative slopes. That is, the interior is *not* completely visible from a top view. This specific example consists of a model with eight trimmed surfaces. An exhaustive search of five views achieves a 99.738% coverage. Examining Table 1 we can see that an exhaustive search of four views doesn't achieve a good enough result due to the negative slopes of the interior. The greedy search results are less satisfactory, requiring six views to achieve similar results to the exhaustive search of five views.

Figure 6 presents our last example. A fairly complex 2D section (an outline of the letter 'r') is swept along an arc. While the greedy algorithm achieves poor results for this model, as seen in Table 1, the exhaustive search of four views displayed here achieves a coverage of over 99%. Obviously a more extensive exhaustive search of five or six views can be conducted to achieve an even better cover.

Although the algorithm as presented finds the minimal set of directions, a question remains. Given two adjacent mold parts, where will the parting line be? When the visibility maps of two adjacent mold parts overlap, the parting line can be anywhere inside that overlapping domain. In the infrequent cases when the different visibility maps are only tangent to each other, the parting line is simply this tangent boundary line.

5. Extension and Future Work

In this work, we have employed a reduction of any GCP with proper parameterization to a set-cover over vectors of bits. A critical question emerges as to the accuracy of the computation. Given a resolution of the visibility map (e.g. 4096^2 in Section 4) and an expected error of about half a pixel, what will be the error in the original model, in Euclidean space?

The error introduced by this visibility mapping can be estimated by the Jacobian of the parameterization of $\mathcal{C}$. The first order partial derivatives of $\mathcal{C}$ clearly govern the ratios between changes in the domain and the range. Hence, bounds on these derivatives can provide the desired answers. It should be further noted that given a piecewise polynomial/rational parametric surface, its derivatives are also piecewise polynomial/rational and hence these bounds can be easily established for B-spline

Example (Figure Number)	Number of polygons	Visibility col-lapsed vector	Visibility maps creation time (hh:mm:ss)	Pre-processing time - collapsing (mm:ss)	Number of views	Set-cover			
						Exhaustive		Greedy	
						time (hh:mm:ss)	Cover (%)	time (mm:ss)	Cover (%)
Spherical shape (2)	734353	55011	04:03:44	02:51	4	03:17:26	99.997	00:01	99.318
A cup (3)	183044	162441	02:16:26	04:11	3	00:08:50	99.712	00:04	97.933
A cup (3)	183044	162441	02:16:26	04:11	4	09:52:35	99.995	00:04	99.827
Utah Teapot (4)	1649758	136010	07:02:18	01:37	4	08:40:29	95.573	00:06	92.870
Utah Teapot (5)	1649758	136010	07:02:18	01:37	5	432:54:48	99.738	00:06	98.466
Utah Teapot (4)	1649758	136010	07:02:18	01:37	6	—	—	00:06	99.785
An arc model (6)	157246	111246	01:39:48	02:17	4	06:30:26	99.219	00:01	94.729

Table 1: Additional statistics on the mold-design examples presented in this section. See also Figures 2 to 6. The exhaustive search of six views of the Utah Teapot was too expensive to compute and is therefore not shown.

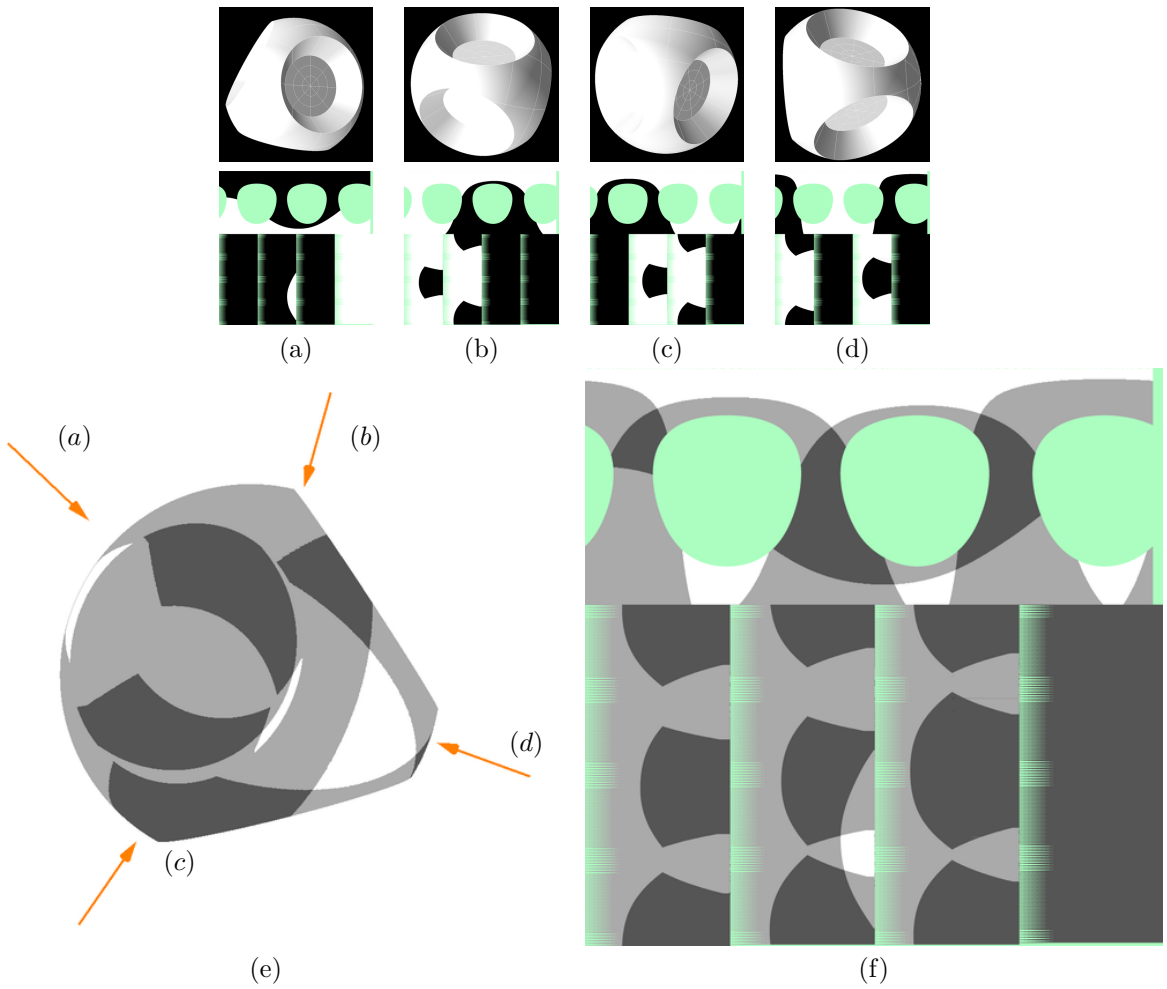

Figure 2: A sphere with four depressions, $\mathcal{C}$. The displayed results are for an exhaustive search of four views. (a) to (d) show, at the top row, the selected views in Euclidean space whereas the bottom row shows the visibility maps of all trimmed surfaces, with the sphere at the top and the four depression below it. (e) and (f) show general final views of the model with the four views merged together. The total coverage for the four views is 99.997%

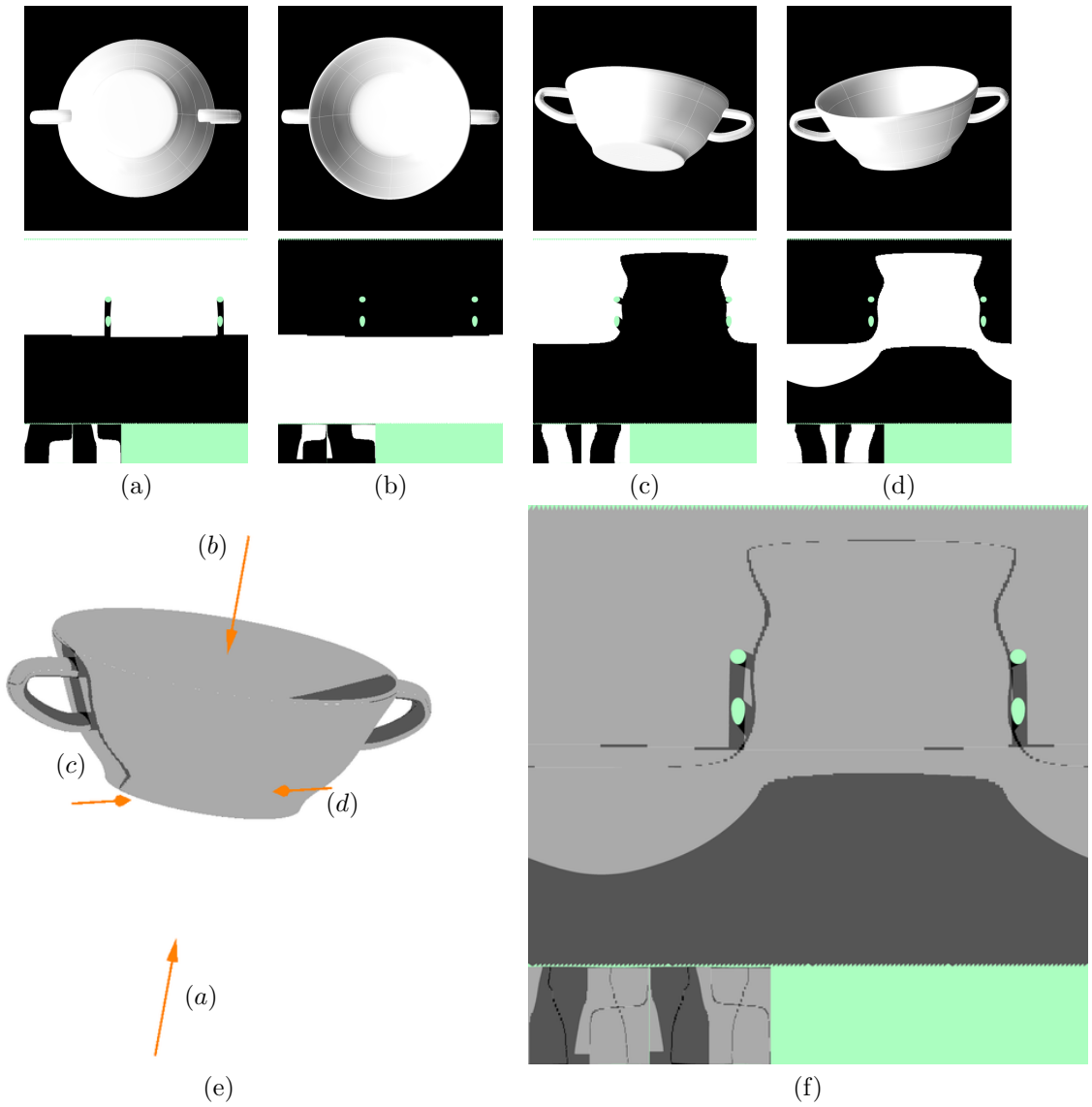

Figure 3: A cup model, $\mathcal{C}$. The displayed results are for a greedy search. (a) to (d) show, at the top row, the selected views in Euclidean space whereas the bottom row shows the visibility maps of all trimmed surfaces with the body (inside and outside) at the top and the two handles below it. (e) and (f) show general final views of the model with the four views merged together. The following coverages are achieved as we add viewing locations from (a) to (d): 49.902% 95.513% 97.933% 99.827%.

and NURBS freeforms. Similarly, given a polygonal mesh with UV parameterization, the first order derivatives can be bound from the ratios between finite differences of the XYZ values and UV values, at the vertices of each triangle in the mesh. Using bounds on the Jacobian, one can also map the percentages of the coverage in the parametric domain which are used through out this article with a similar bound on the percentages of the coverage in the Euclidean space.

A related concern that is relevant, for example, for security applications is the distance of the guard from the inspected region. A simple partial remedy can be found in constraining the visibility map computation to some maximal distance. That is, an event of a distance larger than a designated limit will tag the inspected pixel 'invisible', regardless. A more precise solution can also embed this distance factor into the mapping from $\mathcal{C}$ to its visibility map, taking into account the perspective distortion. Also related is the angular deviation of the normal of $\mathcal{C}$ at the pixel and the view direction, possibly reducing the effective exposed differential area element to the guard. Ideally, the surface normal of $\mathcal{C}$ is along the line of sight. On the other extreme, if they are orthogonal, extraction of the molded piece will be difficult. Here again, locations in $\mathcal{C}$ may be tagged 'invisible' based on the angular relation between the normal of $\mathcal{C}$ at that location and viewing direction.

In some cases, $\boldsymbol{\mathcal{I}}$ may be given in a discrete form. However, another source for error can stem from the fact that we sample $\boldsymbol{\mathcal{I}}$, if $\boldsymbol{\mathcal{I}}$ is continuous. Given some compound to guard, $\boldsymbol{\mathcal{I}}$ might be a curve circumventing the compound. If an airport is to be inspected, its entire

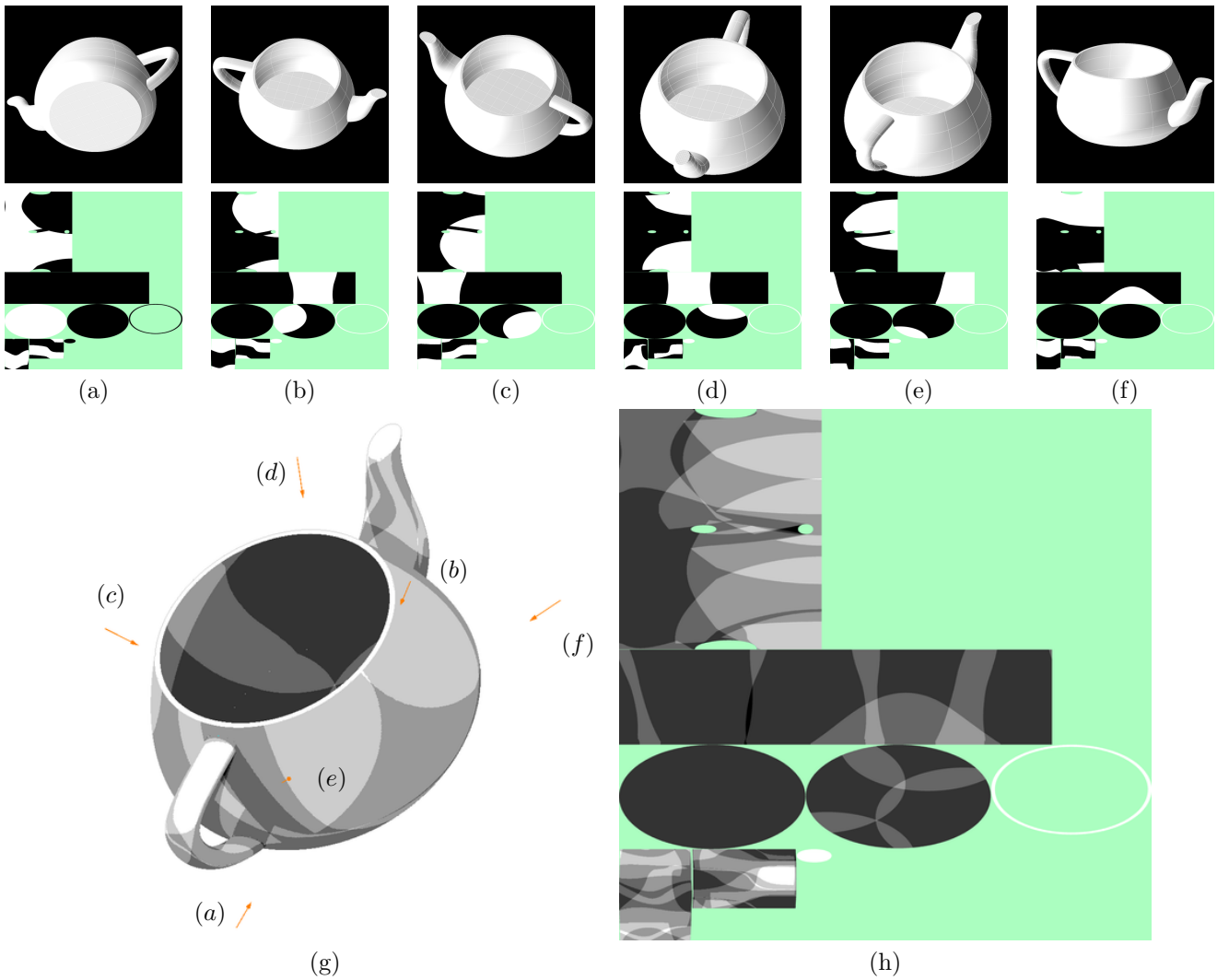

Figure 4: The Utah Teapot with its interior having negative slopes, $\mathcal{C}$. The displayed results are for a greedy search. (a) to (f) show, at the top row, the selected views in Euclidean space whereas the bottom row shows the visibility maps of all trimmed surfaces as (top to bottom, left to right): outside body, inside body, outside base, inside base, top body ring, spout, handle, spout's cover. (g) and (h) show a general view of the teapot with all six views merged together. The coverings gained by adding additional views from (a) to (f) are 37.631%, 69.504%, 83.591%, 92.870%, 98.966%, and 99.785%. Compare with Figure 5.

2D area can serve as locations from which to inspect. In the regular mold-design problem, the entire unit sphere of directions can be employed. Proper and/or optimal selection of view directions has already been considered, for example in [18, 17], and can clearly benefit the results presented in the work.

While the presented framework resorts to sampling $\boldsymbol{\mathcal{I}}$ if $\boldsymbol{\mathcal{I}}$ is continuous, there is no restriction on what a view-location denotes. For example, consider a case where one can compute the visibility map of $\mathcal{C}$ for a moving camera. That is, one can delineate a model for all its visible parts, from at least one camera position along its path. Then, that moving camera (and its single visibility map) can be equally used as one inspection location in the set-covering stage, much like a stationary camera.

A very challenging question in the context of GCP's is how can one handle GCP's continuously, for either $\boldsymbol{\mathcal{I}}$ or $\mathcal{C}$ or both, if at all. Beyond the theoretical question of handling set-covering problems for continuous subsets, how can one even build the visibility maps continuously? One simple option is to build the visibility maps for (trimmed) surfaces as trimmed surfaces, as in [23]. That is, trimming curves will serve as boundaries between the visible and invisible portions. How to handle set-covering over these continuous domains is as yet a completely open question. Alternatively, one can hope that since the GCP is inherently continuous, a whole different approach could be exploited, not using the discrete set-cover solution at all.

As can be seen in Section 4, a major portion of the computation was devoted to creating and pre-processing the visibility maps and solving exhaustive set-covers over 1D vectors of bits. In this work, software implementation was used but it is clear that GPU hardware can be employed to achieve major improvements to these bottlenecks. Indeed the resolution of 4096^2 needs also to

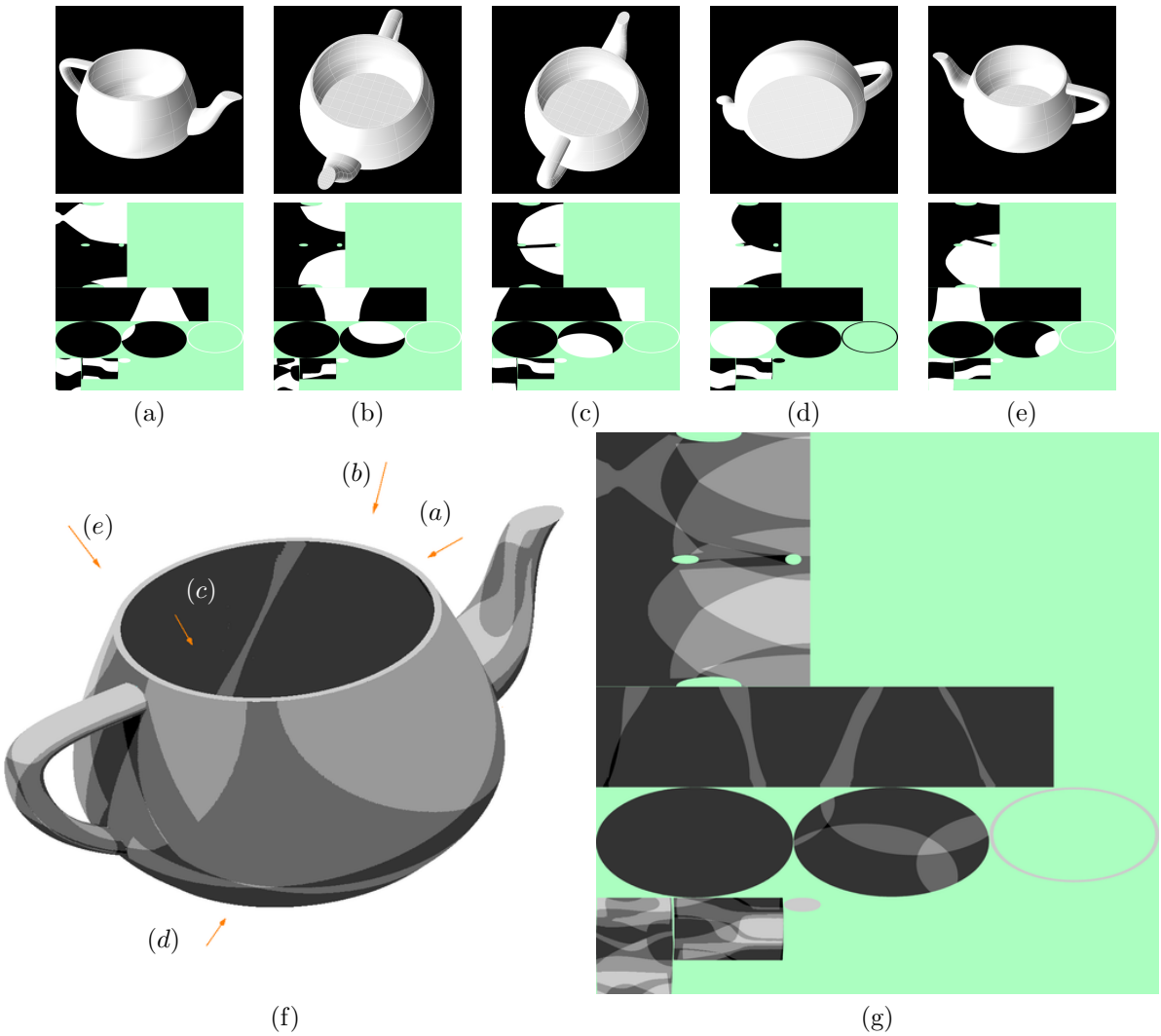

Figure 5: The Utah Teapot with its interior having negative slopes, $\mathcal{C}$. The displayed results are for an exhaustive search of five views. (a) to (e) show, at the top row, the selected views for five views exhaustive search in Euclidean space whereas the bottom row shows the visibility maps of all trimmed surfaces as (top to bottom, left to right): outside body, inside body, outside base, inside base, top body ring, Spout, Handle, Spout's cover. (f) and (g) show a general view of the teapot with all five views merged together. The total coverage for the five views is 99.738%. Compare with Figure 4.

be improved, expecting in practice to process models of size $O(10^2)$ millimeters in an accuracy of microns. It is our expectation that the growth rate in the number of collapsed vectors will be sub-linear with respect to the resolution but this postulation is yet to be examined. Moreover and while the exhaustive set-cover is of exponential complexity, one can hope that the use of GPU's can make its use practical for more than very few guards.

As stated earlier, in the presented examples we have considered 266 visibility maps. The greedy algorithm scans all visibility maps for the one that improves its local result the most. Therefore, the complexity of the greedy algorithm will only show a linear growth as the number of visibility maps increases. In contrast, the exhaustive algorithm examines all possible visibility map-tuples of up to a prescribed size. Hence, the complexity will now grow exponentially with the increase in number of visibility maps. That said, the presented framework can benefit from any algorithm that does better selection of candidates for inspection locations.

The discretization nature of the suggested framework may yield small gaps along the boundaries of adjacent visibility maps where only part of the region represented by a pixel is actually seen. In other words, the framework as presented, has a hard time ensuring 100% coverage. This difficulty can be partially addressed by applying some small dilation of the invisible regions in the visibility maps. Similarly and as already stated, one can also eliminate pixels that represent visible regions following some optimization functions such as distance from the guard or inspection angle.

In this work, we did not force the visibility maps to be connected, a condition that ensures every visibility map represents a single mold part. Pre-processing every visibility map so that only its largest connected component remains in the set-cover analysis, can introduce this

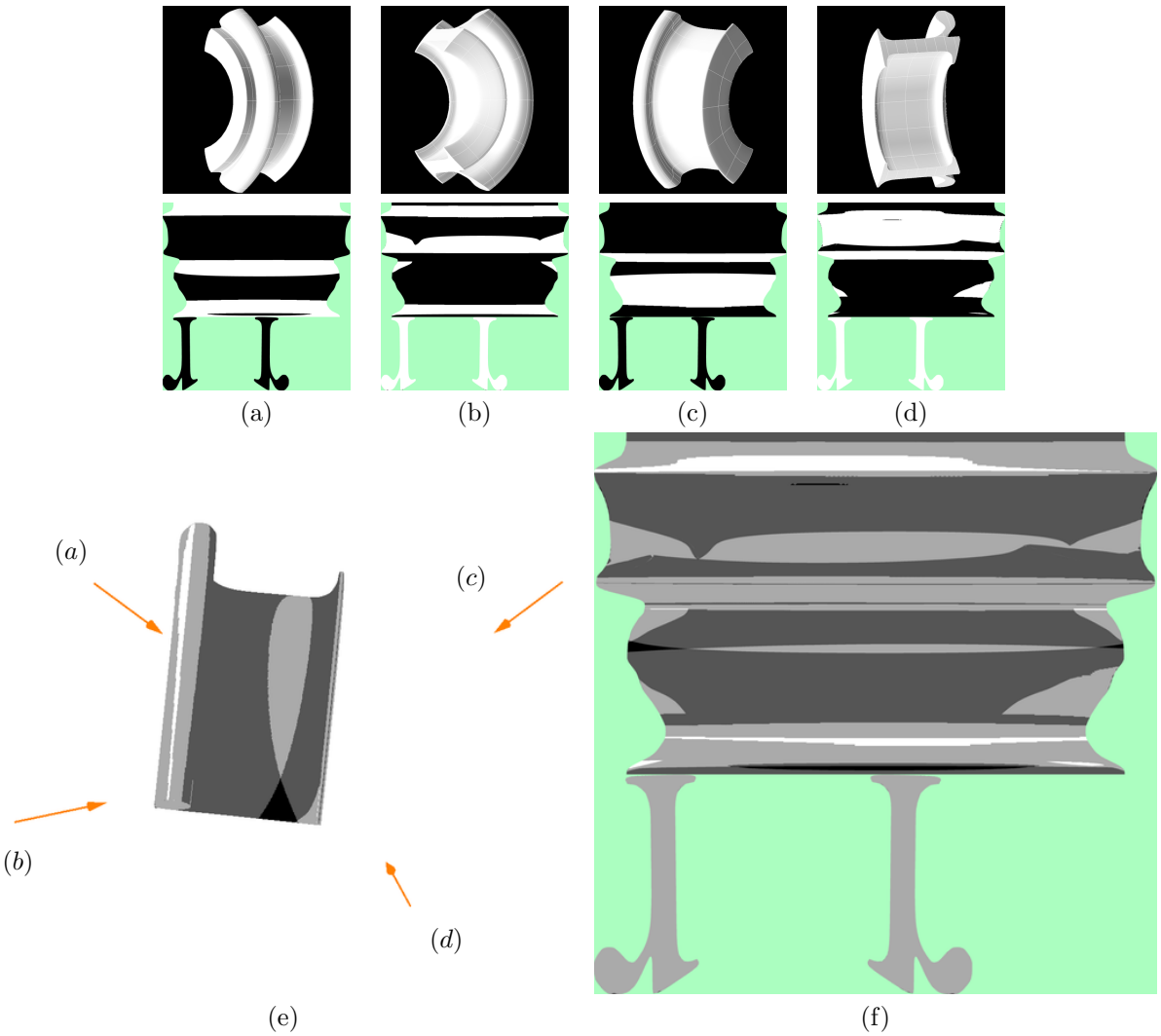

Figure 6: An arc model, $\mathcal{C}$. The displayed results are for an exhaustive search of four views. (a) to (d) show, at the top row, the selected views in Euclidean space whereas the bottom row shows the visibility maps of all trimmed surfaces with the rounded arc surface at the top. (e) and (f) show general views of the model with the four views merged together. The total coverage for the four views is 99.219%.

constraint, possibly allowing easier assembling of the 3D mold parts. Alternatively, each visibility map can be replaced by several visibility maps, one for each connected component of the original visibility map, though this will also increase the computation time.

This last continuity question raises a larger question of what is a good mold, in mold design context, and what is a good coverage, in general. Assuming one can define an objective function that captures the knowledge of what is a good coverage, how can we employ that objective function in the presented framework? Another example can stem from the overlaps between adjacent visibility maps. Overlapping regions are typically a measure of redundancy and large overlaps hint on a large amount of redundancy. Can one exploit this redundancy information to optimize the inspection directions? Can one perturb the inspection directions so as to create better overlaps and hence better parting lines (that reside in the overlapping zones)?

Finally, it is obvious that other geometric covering problems can benefit from the proposed scheme. Additional examples can include the art-gallery problem, security and surveillance, illumination design, coverage of cellular antennas, etc.

6. Conclusions

In this work, we presented a simple framework that is able to handle continuous geometric covering problems (GCP) by reducing them to a discrete set-cover problem over vectors of bits. Due to the simplicity of the approach, it is also very general, drawing on a low common denominator of geometric covering problems in any dimension, and only requiring a parameterization for the inspected domain.

That said, one has to remember we only offer an approximated discrete solution due to several reasons: the continuous inspection space is discretized, the continuous object is represented using discrete visibility maps,

and the set-covering solution is possibly only approximated as well.

Acknowledgments

The authors are grateful to the anonymous reviewers for their valuable comments and to Eyal Guthmann and Eyal Posener that helped in the implementation of the presented algorithm.

The research leading to these results has received partial funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA grant agreement PIAP-GA-2011-286426, and was supported in part by the Technion Vice President for Research Fund - Glasberg-Klein research fund.

References

[1] J. O'Rourke, Art gallery theorems and algorithms, Oxford University Press Oxford, 1987.
URL http://maven.smith.edu/~orourke/books/ArtGalleryTheorems/art.html

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to algorithms (3rd ed)., The MIT press, 2009.

[3] S. McMains, X. Chen, Determining moldability and parting directions for polygons with curved edges, in: International mechanical engineering congress and exposition. ASME, Anaheim, CA, 2004, pp. 359–368.

[4] S. McMains, X. Chen, Finding undercut-free parting directions for polygons with curved edges, Journal of computing and information science in engineering 6 (2006) 60.

[5] G. Elber, R. Sayegh, G. Barequet, R. R. Martin, Two-dimensional visibility charts for continuous curves, in: Shape Modeling and Applications, 2005 International Conference, IEEE, 2005, pp. 206–215.

[6] L. Chen, S. Chou, T. Woo, Parting directions for mould and die design, Computer-Aided Design 25 (12) (1993) 762–768.

[7] J. Madan, P. Rao, T. Kundra, Die-casting feature recognition for automated parting direction and parting line determination, Journal of Computing and Information Science in Engineering 7 (2007) 236.

[8] R. Khardekar, G. Burton, S. McMains, Finding feasible mold parting directions using graphics hardware, Computer-Aided Design 38 (4) (2006) 327–341.

[9] G. Elber, X. Chen, E. Cohen, Mold accessibility via gauss map analysis, in: Shape Modeling Applications, 2004. Proceedings, IEEE, 2004, pp. 263–272.

[10] H. Ahn, M. De Berg, P. Bose, S. Cheng, D. Halperin, J. Matoušek, O. Schwarzkopf, Separating an object from its cast, Computer-Aided Design 34 (8) (2002) 547–559.

[11] X. Chen, S. McMains, Finding all undercut-free parting directions for extrusions, Geometric Modeling and Processing-GMP 2006 (2006) 514–527.

[12] X. Chen, S. McMains, Finding all undercut-free parting directions for solids of revolution, in: ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference. Las Vegas, Nevada, USA, 2007, pp. 489–500, proceedings of IDETC/CIE 2007. September 4-7.
URL http://www.me.berkeley.edu/~mcmains/pubs/DETC07ChenMcMains.pdf

[13] H. Ahn, S. Bae, S. Cheng, K. Chwa, Casting an object with a core, Algorithms and Computation (2005) 40–49.

[14] H. Ahn, O. Cheong, R. Van Oostrum, Casting a polyhedron with directional uncertainty, Computational Geometry 26 (2) (2003) 129–141.

[15] R. Singh, J. Madan, Optimal selection of parting line for die-casting, in: 2nd National conference on Precision Manufacturing, 2010.

[16] P. Chakraborty, N. V. Reddy, Automatic determination of parting directions, parting lines and surfaces for two-piece permanent molds, Journal of Materials Processing Technology 209 (5) (2009) 2464 – 2476.
URL http://www.sciencedirect.com/science/article/pii/S0924013608004883

[17] A. Priyadarshi, S. Gupta, Geometric algorithms for automated design of multi-piece permanent molds, Computer-Aided Design 36 (3) (2004) 241–260.

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

[19] S. Stoyan, Y. Chen, Multi-piece mold design based on linear mixed-integer program toward guaranteed optimality, in: Manufacturing Automation (ICMA), 2010 International Conference on, 2010, pp. 112 –119.

[20] 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.
URL http://www.sciencedirect.com/science/article/pii/S0010448509001213

[21] M. Liu, K. Ramani, Computing an exact spherical visibility map for meshed polyhedra, in: Proceedings of the 2007 ACM symposium on Solid and physical modeling, SPM '07, ACM, New York, NY, USA, 2007, pp. 367–372.
URL http://doi.acm.org/10.1145/1236246.1236299

[22] K. Suthunyatanakit, E. L. Bohez, K. Annanon, A new global accessibility algorithm for a polyhedral model with convex polygonal facets, Computer-Aided Design 41 (12) (2009) 1020 – 1033.
URL http://www.sciencedirect.com/science/article/pii/S0010448509002140

[23] J. Seong, G. Elber, E. Cohen, Simultaneous precise solutions to the visibility problem of sculptured models, Geometric Modeling and Processing-GMP 2006 (2006) 451–464.

[24] S. J. Eidenbenz, (in-)approximability of visibility problems on polygons and terrains, Ph.D. thesis, Technisch Wissenschaften ETH Zurich, Zurich (2000).

[25] J. Foley, A. van Dam, S. Feiner, J. Hughes, Computer Graphics: Principles and Practice, second edition in C, Addison-Wesley Professional, 1995.

[26] G. Elber, Irit 11.0 user's manual, http://www.cs.technion.ac.il/~irit, technion (2012).

[27] R. H. Hardin, N. J. A. Sloane, W. D. Smith, http://www2.research.att.com/~njas/coverings/index.html (1997).