Finding the minimum illuminating direction set for a polyhedron

Guangyu Hou

Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/etd

Part of the Computer Sciences Commons

Recommended Citation

Hou, Guangyu, "Finding the minimum illuminating direction set for a polyhedron" (2019). Graduate Theses and Dissertations. 17466.

https://lib.dr.iastate.edu/etd/17466

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Finding the minimum illuminating direction set for a polyhedron

by

Guangyu Hou

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Yan-Bin Jia, Major Professor
Matthew C. Frank
David Fernandez-Baca

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this thesis. The Graduate College will ensure this thesis is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University
Ames, Iowa
2019

Copyright © Guangyu Hou, 2019. All rights reserved.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>iii</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>v</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>vi</td>
</tr>
<tr>
<td>CHAPTER 1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>CHAPTER 2. LITERATURE REVIEW</td>
<td>4</td>
</tr>
<tr>
<td>CHAPTER 3. ILLUMINATION STRATEGY</td>
<td>6</td>
</tr>
<tr>
<td>1. Boundary Tracing Visibility Algorithm</td>
<td>7</td>
</tr>
<tr>
<td>2. Visibility Polygons Construction</td>
<td>11</td>
</tr>
<tr>
<td>3. Overlay of Visibility Polygons</td>
<td>16</td>
</tr>
<tr>
<td>4. Minimum Set Cover Solution</td>
<td>19</td>
</tr>
<tr>
<td>CHAPTER 4. IMPLEMENTATION</td>
<td>22</td>
</tr>
<tr>
<td>CHAPTER 5. CONCLUSIONS</td>
<td>26</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>28</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1-1 Five axis milling of an impeller ................................................................. 1

Figure 1-2 The minimum illuminating direction set for the concave region of letter C. ................................................................. 2

Figure 1-3 2D and 3D visibility example ................................................................... 3

Figure 1-4 Formulating a set cover problem from the visibility polygons’ subdivision................................................................. 3

Figure 2-1 Cube projection of sphere adopted by S2Geometry ........................................ 5

Figure 3-1 Point and face visibility ........................................................................ 8

Figure 3-2 Boundary tracing algorithm for a pair of faces ....................................... 9

Figure 3-3 Contact pair and sliding planes ................................................................. 9

Figure 3-4 Half spaces intersection for non-visibility ................................................. 11

Figure 3-5 Six sliding planes of a pair of triangles ..................................................... 11

Figure 3-6 Half-planes intersection of oriented lines created by sliding planes......... 12

Figure 3-7 The scope of obstacles for a polyhedral face .......................................... 13

Figure 3-8 Non-visibility polygons of a base face (in red) against five obstacle faces (in yellow). ........................................................................ 14

Figure 3-9 A collection of non-visibility polygons (in green) for a polyhedral face (in red); ........................................................................ 14

Figure 3-10 Visibility polygons (in white) and non-visibility polygons (in shadow) after considering all obstacles. ........................................ 15

Figure 3-11 Convex hull separated concave clusters ............................................... 15

Figure 3-12 One iteration of overlay between two polygons ...................................... 18

Figure 3-13 Overlay of multiple spherical polygons ............................................... 18

Figure 3-14 A greedy algorithm that solves the minimum set cover problem .......... 21
Figure 4-1 Visibility polygons of polyhedron "Cube with slot" ................................. 23
Figure 4-2 Visibility polygons of a U shape bracket. .................................................... 23
Figure 4-3 Visibility polygons of four different polyhedra. .......................................... 24
Figure 4-4 The complete subdivision of visibility polygons for polyhedron "Cube with slot" .................................................................................................................. 25
Figure 4-5 The minimum illuminating direction set solution for polyhedron "Cube with slot" ................................................................................................................ 25
ACKNOWLEDGMENTS

I would like to thank my committee chair, Dr. Yan-Bin Jia, and my committee members, Dr Matthew Frank and Dr. David Fernandez-Baca, for their guidance and support throughout the course of this research.

In addition, I would also like to thank my friends, colleagues, the department faculty and staff for making my time at Iowa State University a wonderful experience.
ABSTRACT

The minimum illuminating direction set cover problem for a polyhedron seeks the minimum cardinality set of 3-D directions that illuminate the entire polyhedron surface. This thesis introduces a new algorithm for solving such problem. The algorithm includes four steps: (1) Computing sliding planes; (2) Constructing visibility polygons; (3) Conducting overlay of polygons on the unit sphere and (4) Applying greedy algorithm to solve a set cover problem. Results have shown the algorithm gives correct answers to a set of polyhedra. Because the visibility polygons results are exact, the solution of the minimum illuminating direction set cover problem is accurate, though might not be optimal.
CHAPTER 1. INTRODUCTION

Multi-axis Computer Numerical Control (CNC) machining is a precision manufacturing process for making parts with complex geometry. A critical problem in CNC machining is the planning of a tool’s access directions. A tool’s access direction determines if a part surface is reachable and thus machinable. Reducing the total number of tools’ access directions required for making a part reduces the re-fixturing of a part in the machine. For 3 and 4 axis CNC machines, such reduction is critical because re-fixturing a part introduces precision error.

Figure 1-1 Five axis milling of an impeller [1]

The objective of this thesis is to find the minimum set of tool access directions so that a part’s surfaces are completely machinable.

To simply the problem, we make three assumptions. First, we assume any positive tool access angle is valid (an access angle is the angle between the tool and the surface); Second, the volume of the tool is ignored thus a tool is thin as a line. Third, the part’s geometry is modeled as a polyhedron. Under this simplification, choosing tool access directions is equivalent to choosing directions of parallel rays. Thus, the objective is simplified to the following: Finding the minimum set of parallel rays’ directions so the entire polyhedron surface can be illuminated. Figure 1-2 shows an example of this problem.
Figure 1-2 The minimum illuminating direction set for the concave region of letter C. On the right, five different curve segments are illuminated by five parallel light rays respectively.

Visibility depicts the reachability of light rays to the surface of an object. Since its close relations with accessibility, visibility is of great importance to many applications that need to access the surfaces of a 3D part. For example, the cutting tool in CNC machining [2-5]; the contact probe in Coordinate-measuring machine (CMM) [6, 7]; the laser scanner used to reconstruct 3-D models [8-10] and the parting line determination of molding [11, 12]. Recently, visibility has also been used to solve the accessibility of cutting tools in an additive-subtractive hybrid system [13, 14]. In such a system, materials are built up and removed through multiple iterations, which causes the accessibility of a cutting tool to change dynamically.

Quantitatively speaking, visibility is essentially a set of three-dimensional directions. Because a direction is a vector whose length is ignored, it can be best characterized on a unit sphere on which each point defines a unique direction (In mathematics, a unit sphere is the set of points of distance 1 from a fixed central point, where a generalized concept of distance may be used). Accordingly, visibility is represented as a set of points on a unit sphere, the boundaries of which form polygon(s) on a unit sphere. The nature of visibility determines that such polygons are simple polygons without holes (proven in Chapter 3). Therefore, visibility can be quantified by a set of simple polygons on a unit sphere (Figure 1-3).
Figure 1-3 2D and 3D visibility example

Once all visibility polygons are obtained for the set of faces on a polyhedron (represented on a unit sphere), they form a subdivision of the unit sphere. We can investigate which subdivided polygon is mapped to by a maximum number of faces. Such a polygon has the best illuminating capability of the polyhedron (in terms of number of faces). If we keep choosing such maximum polygons from the subdivision, we end up with a greedy algorithm to solve the minimum illuminating direction set cover problem (Figure 1-4).

Figure 1-4 Formulating a set cover problem from the visibility polygons’ subdivision. Top: Visibility polygons mapped to by faces. Bottom: Faces mapped to by visibility polygons.
CHAPTER 2. LITERATURE REVIEW

The machining configuration problem is addressed by many researchers through solving a visibility set problem [15]. Visibility determination has received considerable attention. Chen and Woo conducted the seminal work that introduced the Gaussian Map [16]. The basic idea is to compute a dual image of the Gaussian map on a unit sphere. Their idea has been applied to many applications in 4 and 5 axis planning [17]. Suh and Kang introduced a global visibility map generation method by discretizing the visibility sphere into spherical triangles and applying an occupation test to obtain visibility [18]. Spitz and Requicha computed accessibility (closely related to visibility) using depth-buffer and projections in computer graphic hardware [19]. Li and Frank [20] introduced a method to compute non-visibility by occlusion computation between a pair of polygonal faces. Liu and Ramani [21] introduced a similar pairwise occlusion method. The difference is that the occlusion region is obtained from the spherical convex hull of extremal stabbing lines of two faces. Our previous work on approximate visibility used slices cut from the 3D model and discretized the visibility sphere into a set of meridians [4].

After visibility is determined, Tang and Liu used a central projection to project visibility obtained from different faces to a shared plane, for example, $z = 1$ plane [22]. Gupta et al used the same central projection to solve the minimum setups problem for 4 and 5 axis CNC machines [23]. Such projection has some disadvantages. Because central projection is discontinuous along the equator, visibility polygons that cross the equator will be projected to either two planes ($z = 1$ and $z = -1$) or in one plane but separated; such polygons become unbounded. Special handling is required for this separation and unboundedness. Hu, Tang et al used Mercator projection (one kind of cylindrical projection)
for visibility [24]. Mercator projection is discontinuous along the $180^\circ$ meridian. Similarly, the polygons across the $180^\circ$ meridian and two poles are separated. Liu and Ramani sampled the visibility sphere into discrete and pre-defined points. They built a Boolean visibility matrix by querying yes/no visibility on these sample points [25]. Due to sampling, the solution of the minimum illuminating direction set is unlikely optimal. In this thesis, we use S2Geometry library from Google [26, 27] that adopts cube projection. Such projection contains no seams or singularities. It also facilitates Boolean operations of polygons on a unit sphere. This avoids the complexity to handle discontinuity in central or Mercator projection. It also uses snap rounding (a robust geometric computing technique) to eliminate topological errors as mentioned in the work by Ha and Yoo [28].

Figure 2-1 Cube projection of sphere adopted by S2Geometry
CHAPTER 3. ILLUMINATION STRATEGY

To find the minimum illuminating direction set of a polyhedron, the visibilities of its faces are analyzed. Polyhedra containing only triangular faces (i.e. stereolithography file format) are used throughout in the thesis, in spite that polyhedra containing convex polygonal faces of more than three edges are theoretically applicable.

To solve for the exact boundaries of visibility, an exact visibility algorithm for polyhedra comprised of only convex polygonal faces is used [20]. What it generates are visibility polygons for each polyhedral face. The definition of visibility for a face and the algorithm detail will be introduced in the later section. Although the unit spheres on which we represent visibility differ their origins from face to face, when solving for a common illuminating direction (which equals a set of parallel lines), origin differences can be ignored. Therefore, all visibility polygons share the same unit sphere. In this thesis, the center of the polyhedron being analyzed is set as the origin of the unit sphere.

To simplify phrasing, the word “polygons” later in this section refers solely to visibility polygons on the unit sphere, not the faces of a polyhedron. Also, the word “regions” later in this section refers solely to any points/lines/polygons or combined on a unit sphere.

The visibility algorithm’s output – spherical polygons grouped by faces is, collectively, a mapping $M_1$ from faces to polygons. The domain is a set of faces. The codomain is a set of computed polygons on the same unit sphere. The minimum illuminating direction set problem is equivalently to find the minimum number of polygons whose inverse mapping cover all the faces. Therefore, the inverse mapping $M_2$ from polygons to faces is required. In order for the domain of $M_2$ to be complete, in other words, in order for the
domain to contain as many elements as possible so each element is mapped to a distinct set of faces in the codomain, the original polygons needs to be subdivided so the codomain of $M_1$ contains only disjoint elements/polygons. Geometrically, any polygon in the codomain of $M_1$ should not overlap with each other. This can be done by an overlay of all the polygons. The subdivision resulted from the overlay essentially subdivide each original polygon into several disjoint polygons. The detail of the overlay process is provided in the later section. Once the inverse map is obtained, we can formulate a minimum set cover problem and use greedy algorithm to solve it. A greedy algorithm is chosen because up until now it is still the best possible polynomial time approximation algorithm for set cover up to lower order terms [29].

The above mapping $M_1$ has the following properties. Because a visibility region might be mapped to by multiple faces, the mapping is non-injective or not one-to-one. Because any visibility region must be mapped to by some faces, the mapping is also surjective or onto (imagining illuminating the polyhedron from a direction with parallel light beams, some faces, with its entire area will be lighted). Similarly, the inverse mapping $M_2$ from polygons to faces is non-injective because a face might be mapped to by multiple regions. $M_2$ is not surjective because it is possible that some faces are not visible at all.

Each of the procedure mentioned above is explained in separate sections below.

1. **Boundary Tracing Visibility Algorithm**

As polyhedra are of the interest of this thesis, the visibility of their faces needs to be determined. The chosen visibility algorithm uses a pairwise occlusion method between two distinct faces of the polyhedron. The face being analyzed for visibility is called base, while
the other is called obstacle. The occlusion is done by tracing the boundary of the obstacle with a 3-D light beam emitted from the base.

Intuitively, (1) A point is visible from a direction if and only if the ray emitting from the point in that direction does not collide with any obstacle. (2) The base is visible from a direction if all points in the base are visible from that direction. (3) A 3-D light beam is a collection of rays emitted from points in the base in the same directions (Figure 3-1). Therefore, the base is visible from a direction if and only if the 3-D light beam of the base does not collide with any obstacle. To find the exact boundary between collision and non-collision, one can trace the obstacle boundary with a 3D light beam. The movement of the 3-D light beam gives the boundary of collision, or equivalently the non-visibility of base due to obstacle (Figure 3-2). The next step is to determine of the movement of tracing.

![Figure 3-1 Point and face visibility](image-url)
The movement of the tracing is in fact a set of sliding movements of the 3-D light beam against the obstacle. Each sliding movement occurs on a plane determined by a contact pair. The contact pair is composed of one vertex or edge of the obstacle boundary and one side edge or side face of the 3-D light beam (or equivalently, a vertex or edge of the base (Figure 3-3)). Such planes are called sliding planes. A sliding plane is tangent to both base and obstacle and separates them into two complementary half-spaces. Also, a sliding plane subdivides the direction space (a hemisphere) of 3-D light beam into two quarter-spaces: In
one, the 3-D light beam can move arbitrarily without collision (I); In the other, the 3-D light beam will collide with the obstacle in some directions (II). Notice there are multiple sliding planes for a pair of faces. Thus, there are multiple divisions of the space. Because any direction in quarter-space I is collision free, the union of \( I_i \) spaces \( \bigcup I_i \) is the entire collision free space for the 3-D light beam. Accordingly, the complementary space \( \bigcup \overline{I_i} = \cap \overline{II_i} \) is the entire collision space or non-visibility region (Figure 3-4).

In conclusion, the half-space intersection of a set of sliding planes generated for a pair of faces (oriented to quarter-space II) defines the non-visibility of one face due to the other.

The property that (1) a sliding plane separates the base and obstacle into two half-spaces and (2) a sliding plane is tangent to both faces are used to determine sliding planes. Notice for a pair of faces, the distinction between base and obstacle is trivial when determining sliding planes because they share the same set of sliding planes. We can simply denote a pair of faces as \( \text{face}_1 \) and \( \text{face}_2 \). The construction of sliding planes is equivalent to identify the contact pairs from the two faces so that the plane created by such contact pair is (using previous property) (1) tangent to both faces and (2) separates the two faces into two separate half-spaces. The details of the sliding plane determination procedure is out of the scope of this thesis and can be found in [20].
Figure 3-4 Half spaces intersection for non-visibility. $S_i$ denotes sliding plane $i$; $I_i$ denotes visible half-space $i$; $II_i$ denotes the complementary half-space of $I_i$.

2. Visibility Polygons Construction

Using the previous boundary tracing visibility algorithm, we can obtain sliding planes for a pair of faces. An example is given in Figure 3-5.

Figure 3-5 Six sliding planes of a pair of triangles. Contact pairs (edges and vertices) that determine sliding planes are marked with blue and red respectively.

Because the visibility polygon(s) of a specific base face is at most a hemisphere (and is oriented by the face normal), we can in fact project those polygons on the
sphere to a plane that is parallel to the base by a central projection (call this plane local \( z = 1 \) plane). In this manner, we convert the half spaces (of a plane) to half planes (of a line). We do such implicit projection because Boolean operations of polygons on a plane is simpler than that on a unit sphere. For example, if base is on the XY plane (as shown in Figure 3-5), we can choose \( z = 1 \) plane to for the projection. Each sliding plane intersect with the \( z=1 \) plane with a line. The half planes intersection of these oriented lines gives the non-visibility polygon in 2-D (Figure 3-6(a)). We can easily project this 2D non-visibility polygon back to the unit sphere (Figure 3-6 (b) and (c)).

Now, we can compute a polyhedral face’s non-visibility due to another polyhedral face. Because a face’s visibility is usually occluded by multiple other faces, we need to repeat the non-visibility computation on all these faces (an example is shown in Figure 3-8). The union of these non-visibility polygons gives the complete non-visibility region for a base face. The complement of the union gives the visible region (Figure 3-9).

Notice the obstacles for a polyhedral face, in this thesis, are restricted to those from the same polyhedron. Obstacles on other polyhedra are not considered.
Figure 3-7 The scope of obstacles for a polyhedral face

Here are some properties of the complete visibility region and non-visibility region of a face: (1) The visible region of a face might contain multiple disjoint polygons. Each polygon is simple without any hole. To prove the latter, suppose a visibility polygon has a hole. The visibility polygon is now in a ring shape. Recall the definition of face visibility, a 3-D light beam must sweep along the ring shape without collision. The swept volume effectively cuts the 3D space into two isolated subspaces (call them interior and exterior). For the hole to be a non-visibility region, there must be some faces of the polyhedron acting as obstacle in the interior subspace. Because the base face is in the exterior subspace, if there is a polyhedral face in the interior subspace, some faces must cross the ring for a polyhedron to be connected which is a contradiction. (2) The non-visibility region of a face can contain multiple disjoint polygons. Each polygon can contain multiple holes of level at most one (i.e. at most one nested loop). This is because the hole of a non-visibility polygon is a visibility polygon, using previous conclusion, the hole cannot contain any more holes. (3) Non-visibility polygons must be unbounded. This is because a bounded non-visibility polygon is itself a hole of an exterior visibility polygon, which is a contradiction. In Figure 3-10, an example of visibility and non-visibility regions are drawn to illustrate these properties.
Figure 3-8 Non-visibility polygons of a base face (in red) against five obstacle faces (in yellow). Non-visibility Polygons are in blue. (a) Five pairs of faces to compute non-visibility. (b) The non-visibility polygons for the corresponding face pair in isometric view. (c) Same as (b), but in front view.

Figure 3-9 A collection of non-visibility polygons (in green) for a polyhedral face (in red); The union of non-visibility polygons (in purple) and the visibility polygon (in blue). On the right is the visibility polygon projected to a unit sphere.
Figure 3-10 Visibility polygons (in white) and non-visibility polygons (in shadow) after considering all obstacles.

It is worth noting that when picking obstacle faces from the polyhedron, one can rule out many faces that will not contribute to the non-visibility of the base first. These invalid faces include the following:

- Faces that are on the same plane of base.
- Faces that are in the negative space of the base plane.
- Faces that are back facing the base, or equivalently base is in the negative space of such face’s plane (oriented by face norm).
- Faces that are on the convex hull of the polyhedron.
- Faces that are not in the same concave cluster with the base. A concave cluster contains faces (1) not on the convex hull of the polyhedron and (2) connected to each other (via graph traverse). An example is shown in Figure 3-11.

Figure 3-11 Convex hull separated concave clusters. Convex hull is in green and three concave clusters are shown in red, gray and yellow.
Now we can compute the visibility region for every face on the polyhedron. The visibility region, as mentioned above, consists of one or more simple polygons represented on the plane or on the unit sphere. If represented on the plane, because each face has its own local $z = 1$ plane, we will need to project these polygons to some shared plane for later process. This is possible as mentioned in the Literature Review. We can project all polygons to the $z = 1$ plane of the polyhedron (similarly the $z = -1$ plane, both are central projection). Any spherical polygon should fit into this projection although polygons that cross the equator will be split into two unbounded parts, one in the upper plane and one in the lower plane. This might need special handling of the unboundedness and the separation issues. However, in this thesis, we choose to represent visibility polygons on the unit sphere and conduct Boolean operations directly on spherical polygons using S2Geometry.

3. Overlay of Visibility Polygons

At this stage, visibility polygons of all faces are represented on the same unit sphere. These polygons might overlap, creating smaller regions. Let’s define a congruent region to be a spherical region in which all directions maps to the same set of faces. To pick a direction that illuminates the maximum number of faces is equivalent to pick a congruent region that intersect the maximum number of visibility polygons. Therefore, a subdivision of the unit sphere induced by the visibility polygons is required. Such subdivision is achieved by overlays of visibility polygons. The following is the overlay process.

Suppose initially we label each polygon by the face (that maps to this polygon). Whenever a polygon overlaps with another polygon, an intersection polygon is created and labeled with faces from both polygons. The two difference polygons are also created and labeled by their original faces respectively. This process effectively subdivides the original
polygons and increase the size of total polygons by one. And we continue to do this until there is no overlap of polygons anymore. Intuitively, the overlay process is like re-categorizing. It aims to distinguish polygons, so each has a distinct set of labels (faces). This essentially creates a mapping from polygons to their incident faces.

In this thesis, the overlay process is conducted between a pair of polygons each time. A subdivision of previous polygons is maintained over each iteration of overlay. The subdivision contains all previous subdivided polygons. No one overlaps with others. To record a polygon’s visible faces. Two maps are maintained throughout. The first map maps face id to a set of polygon ids. The second map, which is an inverse map, maps polygon id to a set of face ids.

Starting the subdivision with a set of disjoint polygons from face zero:

Loop: Take in the set of polygons of the next face. Compute overlay: For each pair of polygons A and B (one from the current subdivision and one from the next face), their intersection and two difference polygons are created. The intersection polygon is labeled by union A and B’s faces. The (A-B) difference polygon is labeled with A’s faces and (B-A) difference polygon is labelled with B’s faces. After that, the original polygons A and B are removed from the subdivision. Updates the two maps to reflect the change. An example is given in Figure 3-12 to show one iteration of overlay.

Eventually, the overlay algorithm generates a subdivision of polygons and two maps recording relations between subdivided polygons and faces. An example of spherical polygons overlay is shown in Figure 3-13.
Figure 3-12 One iteration of overlay between two polygons

Figure 3-13 Overlay of multiple spherical polygons. On the left: polygons from the same face are colored the same. In the top middle, a polygon covering the north pole is chosen; On the top right, the chosen polygon is subdivided. Subdivided polygons are drawn on spheres of slightly different radius to differentiate them. In the bottom middle and on the bottom right: Similar to the top one, but a different polygon is chosen.
4. **Minimum Set Cover Solution**

Recall that the goal is to find the minimum illuminating direction set of the polyhedron. Because of the subdivision, all directions in the same polygon illuminates the same set of faces. Thus, it is sufficient to find the minimum illuminating polygon set for the polyhedron. One can apply other criteria as for which directions to choose in the polygon set solution (one direction per polygon).

The overlay algorithm generates a map that maps polygons to faces. Thus, the minimum illuminating polygon set problem is essentially a set cover problem in the following format:

- The universe $U$ is the set containing all polyhedral faces $\{f_1, ..., f_n\}$.
- The collection $S$ contains $m$ subsets of $U$, $m$ equals to the number of polygons in subdivision. Each set, labeled by a polygon id, consists of face ids it illuminates.
- The goal is to identify the smallest sub-collection of $S$ whose union equals to the universe.

This is an optimization version of the set cover problem. It is known to be NP-hard. We use the greedy algorithm to solve it since it is the best-possible polynomial time approximation algorithm for set cover.

The greedy algorithm first iterates through the “polygon to faces” map (defined as Map2) and find the polygon with maximal faces count, say $p_I$. It adds $p_I$ to the solution set. Meanwhile Map2 must be updated so no other polygons map to the faces $p_I$ maps to. For this purpose, we query the “face to polygons” map (defined as Map1). For each face $p_I$ maps to,
find all polygons mapping to it and remove these polygons’ link to the face in Map2. After all links are removed, the faces $p_i$ maps to are removed from Map1. Namely,

$$\forall f_j \in Map2(p_i)$$

$$\forall p_k \in Map1(f_j)$$

Remove link $(p_k \rightarrow f_j)$ in Map2

Remove $f_j$ in $Map1$

After updating Map2, we iterate through Map2 again to find the polygon with maximal faces count. This loops until all faces in Map1 is removed which marks the end of the greedy algorithm. If all polygons in Map2 have been removed while Map1 is still non-empty, then these faces in Map1 are not visible by any polygon, the algorithm will stop too. An example of executing this greedy algorithm is shown in Figure 3-14.

It has been shown that the approximation ratio for greedy algorithm of set cover problem is $\ln n - \ln \ln n + \Theta(1)$, where $n$ is the size of the universe [30].

Now, the minimum illuminating direction set problem is solved. The solution is a set of polygons on the unit sphere. Each polygon contains a set of directions. One needs to pick one direction in each polygon as a final step. It is worth mentioning that, for the purpose of machining, the boundary of the polygon is not favored because it might either render the tool to collide with an obstacle (considering a tool has volume), or it can cause an acute angle between the tool’s access direction and a polyhedral face. The centroid of the polygon is usually the preferred choice since it can avoid both problems and keeps a max clearance to obstacles.
A greedy algorithm that solves the minimum set cover problem. Pick the polygon that renders the maximal number of faces visible in each iteration. (a) The subdivision of three polygons, each is from a different face. (b) The two maps for the subdivision. (c) Pick the polygon that sees the most faces (if there is a tie, pick the first one). (d) Remove faces that are already seen in Map2. (e) Pick the best polygon again. (f) The final solution is obtained when Map1 contain no more faces.
CHAPTER 4. IMPLEMENTATION

The program was implemented in C++ under Microsoft Visual Studio 2015. Three open source libraries are used. libQGLViewer is used for display [31]. The Computational Geometry Algorithms Library (CGAL) is used for 3D geometry construction, predicates and Boolean operation of 2D polygons [32]. S2Geometry is used for Boolean operations of spherical polygons [26].

The overall procedure of our illumination strategy is as follows:

• For each pair of faces on the polyhedron, sliding planes are computed using the visibility algorithm in [20].

• For each face of the polyhedron, the set of eligible obstacles is determined. For each obstacle, non-visibility polygon is generated from corresponding sliding planes and represented on local $z = 1$ plane.

• For each face, non-visibility polygons generated from different obstacles are combined. The complementary of the union gives the visibility polygon(s).

• For each face, the visibility polygon(s) represented on local $z = 1$ plane are mapped to the same unit sphere. This marks the end of visibility computation.

• For all polygons on the unit sphere, an overlay is conducted to create a subdivision of the unit sphere. Faces to polygons (and vice versa) information are stored in a map data structure.

• For the subdivision (consists of a set of polygons), a set cover problem is formulated and solved using greedy algorithm. For the set cover problem, the
collection has size m, where m is the number of polygons in the subdivision. The universe has size n, where n is the size of the polyhedron.

The following are some visibility results from several different polyhedra. Since faces on the convex hull always have visibility of a hemisphere, they are not shown here.

Figure 4-1 Visibility polygons of polyhedron "Cube with slot". (a) Overlay of visibility polygons from all faces. (b-f) Visibility polygons for faces not on the convex hull (the face being analyzed is marked in red).

Figure 4-2 Visibility polygons of a U shape bracket. (a) Overlay of visibility polygons from all faces. (b-d) Visibility polygons for some example faces (the face being analyzed is marked in red).
Figure 4-3 Visibility polygons of four different polyhedra. (a) A face at the bottom of a flower-like-hole of a cube (isometric and top view). (b) A face on the wall of a flower-like-hole of a cube. (c) A face on the wall of a through hole of a cylinder (two views). (d) A face on the back of the “Stanford bunny” polyhedron. (a-d) Face being analyzed is marked red.

Figure 4-4 shows a complete subdivision of visibility polygons for the polyhedron “Cube with slot”. The visibility polygons (as shown in Figure 4-1, plus hemispheres for faces on the convex hull) effectively divided the unit sphere into 15 different polygons. Each polygon illuminates the same set of faces.

The minimum illuminating polygon set solution for polyhedron “Cube with slot” is shown in Figure 4-5. The solution contains two directions, the size of which is optimal. Notice, neither of the directions is at the centroid of any polygon. Rather, they are at the intersection points of these polygons. This is because visibility polygons are assumed to be closed. The intersection point shares the visibility of its neighboring polygons. Therefore, for any polygon, there will always be a point that has a better visibility. This means our algorithm will always choose points over polygons. This might not be ideal. Future work can be done to add constraints to the greedy algorithm so that not all directions in the solutions are points.
Figure 4-4 The complete subdivision of visibility polygons for polyhedron "Cube with slot". Blue polygons are the result of overlay process. The arrow represents a direction from the centroid of the polygon to the origin. Notice, the set of faces each blue polygon illuminates are shown in yellow on the polyhedron (otherwise gray). Recommended reading order: From left to right, from top to bottom.

Figure 4-5 The minimum illuminating direction set solution for polyhedron "Cube with slot". (a) The two green arrows represent solution for the set cover problem. (b) Direction 1 could illuminate all faces of the polyhedron except one on the back. (c) Direction 2 covers the back face that direction 1 fail to illuminate.
CHAPTER 5. CONCLUSIONS

A new algorithm has been designed and implemented to solve the minimum illuminating direction set cover problem for a polyhedron. The algorithm includes four steps: (1) Computing sliding planes; (2) Constructing visibility polygons; (3) Conducting overlay of polygons on the unit sphere and (4) Applying greedy algorithm to solve a set cover problem. Results have shown that all four steps of the algorithm have been implemented successfully and the algorithm gives correct answers to a set of polyhedrons. Because the visibility polygons are exact, the solution of the minimum illuminating direction set problem is accurate, though might not be optimal.

Future work

First, the time complexity of the overlay process of a set of polygons can be improved by using the divide-and-conquer strategy and DCEL data structure. We can divide the set of polygons S into two sets S₁ and S₂, compute their subdivisions recursively, and then compute the overlay of the subdivision from S₁ and subdivision from S₂. As a reference, for n = n₁+n₂, where n₁ is the complexity of subdivision Sub₁, n₂ is the complexity of subdivision Sub₂, the overlay of Sub₁ and Sub₂ takes $O(n \log n + k \log n)$ time.

Second, before sliding planes computation, we can pre-process the polyhedron to trim off more invalid obstacles. One idea is to project potential obstacle faces onto the base face’s hemi-sphere and merge them. Then the original obstacles of the base are replaced with the merged faces. The expectation is that the merging will reduce the number of obstacles in
consideration. However, pre-processing itself has introduced computation overhead. Thus, the use of pre-processing must be balanced.

Third, consider allowing objects from outside the polyhedron to be treated as obstacles. In the machining scenario, there are always supporting structures that connects the part and the platform (e.g. chucks) because parts must be fixtured in machining. These fixtures cannot be machined accidentally. Thus, they can become external obstacles.
REFERENCES


