Computing tool accessibility of polyhedral models for toolpath planning in multi-axis machining

by

Guangyu Hou

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Industrial Engineering

Iowa State University

Ames, Iowa

2019

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Dr. Matthew Frank, for his excellent and consistent guidance, understanding, patience, and providing me with a wonderful atmosphere for doing research. I would like to thank my committee members, Dr. Frank Peters, Dr. John Jackman, Dr. James Oliver and Dr. Chris Harding, for their guidance and support throughout the course of this research.

In addition, I would like to thank my colleagues Shuangyan Lei, Ashish Joshi, Siqi Zhu, Niechen Chen, Prashant Barnawal, Michael Hoefer, David Peiffer, Alicia Guzman, Esra'a Abdel-All and all other colleagues and friends in the RMPL lab for their kindly help in the development of this dissertation. I would also like to thank the department faculty and staff for making my time at Iowa State University a wonderful experience.

Finally, I would like to thank my parents and my brother for their consistent encouragement, respect and love.

ABSTRACT

This dissertation focuses on three new methods for calculating visibility and accessibility, which contribute directly to the precise planning of setup and toolpaths in a Computer Numerical Control (CNC) machining process. They include 1) an approximate visibility determination method; 2) an approximate accessibility determination method and 3) a hybrid visibility determination method with an innovative computation time reduction strategy. All three methods are intended for polyhedral models.

First, visibility defines the directions of rays from which a surface of a 3D model is visible. Such can be used to guide machine tools that reach part surfaces in material removal processes. In this work, we present a new method that calculates visibility based on 2D slices of a polyhedron. Then we show how visibility results determine a set of feasible axes of rotation for a part. This method effectively reduces a 3D problem to a 2D one and is embarrassingly parallelizable in nature. It is an approximate method with controllable accuracy and resolution. The method's time complexity is linear to both the number of polyhedron's facets and number of slices. Lastly, due to representing visibility as geodesics, this method enables a quick visible region identification technique which can be used to locate the rough boundary of true visibility.

Second, tool accessibility defines the directions of rays from which a surface of a 3D model is accessible by a machine tool (a tool's body is included for collision avoidance). In this work, we present a method that computes a ball-end tool's accessibility as visibility on the offset surface. The results contain all feasible orientations for a surface instead of a Boolean answer. Such visibility-to-accessibility conversion is also compatible with various kinds of facet-based visibility methods.

Third, we introduce a hybrid method for near-exact visibility. It incorporates an exact visibility method and an approximate visibility method aiming to balance computation time and accuracy. The approximate method is used to divide the visibility space into three subspaces; the visibility of two of them are fully determined. The exact method is then used to determine the exact visibility boundary in the subspace whose visibility is undetermined. Since the exact method can be used alone to determine visibility, this method can be viewed as an efficiency improvement for it. Essentially, this method reduces the processing time for exact computation at the cost of introducing approximate computation overhead. It also provides control over the ratio of exact-approximate computation.

CHAPTER 1. INTRODUCTION

1.1 Multi-axis CNC Machining

Milling is a machining process that uses rotary cutters to remove material from a workpiece. The cutter is advancing in a direction that is perpendicular to the axis of the cutter (Fig. 1-1a). The capability of making parts to precise dimensions and shapes and creating complex features has made milling one of the most commonly used manufacturing processes in industry. The advent of Computer Numerical Control (CNC) in the 1950s, by which a machining tool's movement is controlled by computers, has upgraded traditional milling machines to machining centers with accurate control and complex mechanical systems, making milling much more automated and precise (Fig. 1-1b). Based on the kinematic capability of the machine, CNC machines are categorized into 3-axis, 4-axis and 5-axis. 3-axis CNC machines have 3 linear axes. 4-axis CNC machine provide one extra freedom by adding a rotary axis. 5-axis CNC machines introduce two rotary axes which, if not for a machine's rotation limits, can pose the tool in arbitrary orientations (Fig. 1-1c).

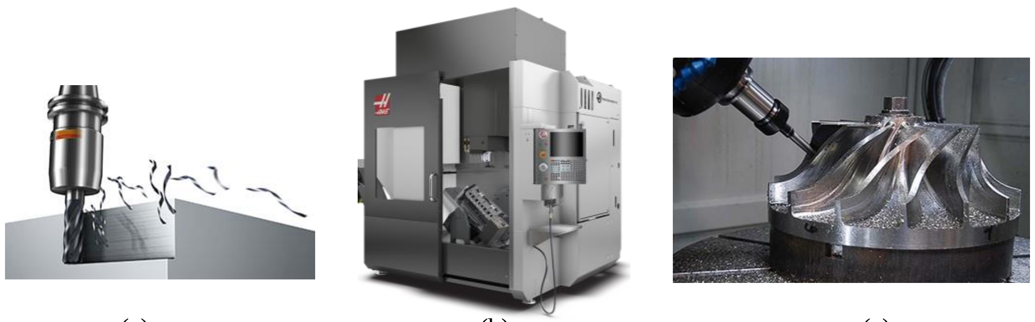
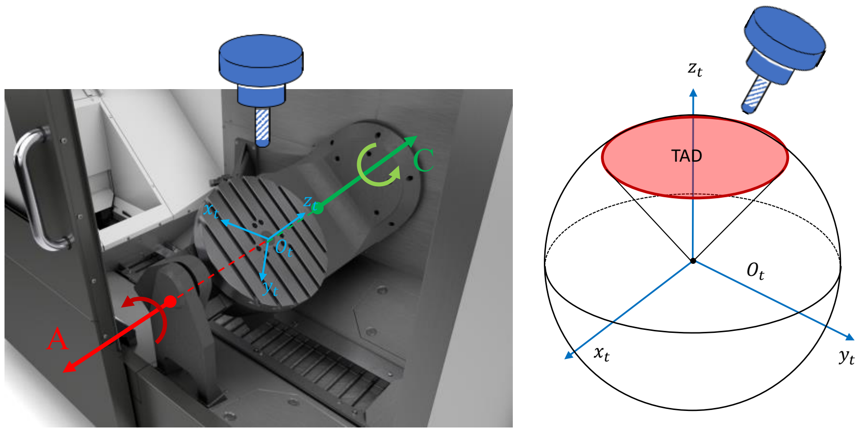


Fig. 1-1 Multi-axis CNC milling and machine centers [1-3]

Tool Access Directions

A workpiece must be fixated on the machine mounting table before it can be machined. Tool Access Directions (TAD) are, in the mounting table coordinates system, all the directions the table can be approached by the tool (definition and part of the subfigures are from [4]). Since 5-axis CNC machines represent the most general case of tool access scenario, we use them as the example to illustrate TAD.



(a) A-C type 5-axis CNC machines (b) Tool Access Directions

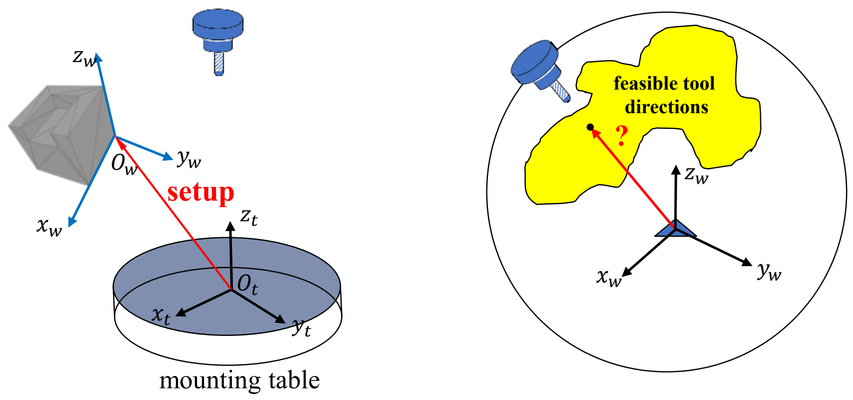
Fig. 1-2 Five-axis trunnion table and its tool access directions [5]

There are many different configurations of rotary axes for 5-axis CNC machines. One common configuration is the A-C table where the platter (mounting table) can rotate around its center line (C-axis) by 0-360 degrees (Fig. 1-2a). This platter is on another rotary axis (A-axis) that can rotate in a range specified by the machine limits. The cutter can move linearly in X, Y or Z axis. Under such configuration, in the mounting table coordinate system, the direction space where the tool can orient (i.e. TAD) is a spherical cap (Fig. 1-2b). TAD defines a machine's reorientation capability.

Setup Planning

As machining is a subtractive manufacturing process, material is gradually removed from a workpiece (e.g. a cylindrical block) and eventually what is left is the desired part.

Before machining, a machinist must decide how the part should be oriented and positioned in the workpiece. Because the workpiece is fixated on a mounting table, it is equivalent to deciding how the part should position and orient with respect to the mounting table. Such a posture is called a setup of the part (Fig. 1-3a). The choice of the setup affects whether specific surfaces of the part can be accessed by the machine tool, considering the machine tool usually cannot approach the mounting table from all directions (namely TAD is smaller than a unit sphere). The choice of what setup to use and how many setups to use in order to completely machine a part is decided in setup planning. It is important as it determines whether all surfaces of a part can be accessible by the machine tool and, if a setup is fixed, from what directions a surface can be accessed by a machine tool (Fig. 1-3b). The latter would affect the available toolpath choices for a surface.



(a) Setup planning (b) Tool path planning

Fig. 1-3 Setup and tool path planning

1.2 Visibility Concept

To help solve the setup planning problem, the visibility concept is introduced. We define the visibility of a point as the collection of directions of a ray that is casting from this point while not colliding with any obstacle (Fig. 1-4a). Similarly, we define the visibility of a surface as the collection of directions of parallel rays that are casting from every point in this surface while not colliding with any obstacle (Fig. 1-4b). An alternative but equivalent definition is given below: We define a surface of a part as visible from a direction if and only if there exists an *infinite distant* point in that direction can see the entirety of the surface without obstruction by the part itself. Such a direction is called a visible direction. The visibility of a surface is defined as the collection of such visible directions. Because we are only concerned with polyhedral models in this dissertation, the smallest surface of a part is a polygon (e.g. a triangle). In such case, the visibility of a polygon surface (a facet) has a more intuitive definition. Suppose we extrude a 3D beam from this facet along a direction where all side edges of this 3D beam are parallel. We define the facet as visible from a direction if and only if the extruded 3D beam in this direction does not collide with any obstacle (Fig. 1-4c). The visibility of a facet is the collection of such directions.

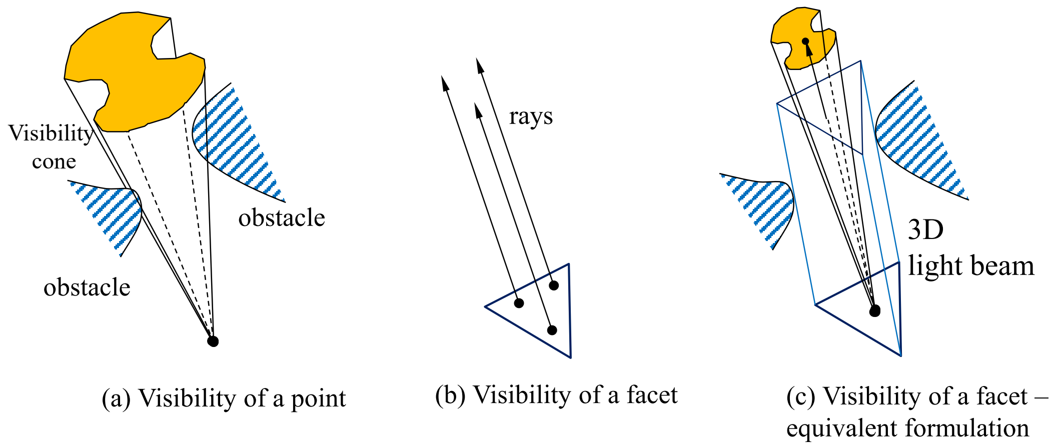

Visibility cone
obstacle
obstacle
(a) Visibility of a point
rays
(b) Visibility of a facet
3D light beam
(c) Visibility of a facet – equivalent formulation

Fig. 1-4 Definition of visibility

Visibility is usually represented on a unit sphere as spherical points because they are essentially a set of directions. Because the boundary of these points creates spherical polygon(s), we use the term visibility polygon(s) to refer to the visibility's exact shape.

Visibility is used in many technological fields where there is a requirement that an object's surfaces be accessed from various directions by tools specific to that field. Such technologies include Coordinate Measurement Machines (CMMs), where visibility determines the accessibility of the contact probe [6]; CNC machining, where visibility serves as the necessary condition for the accessibility of the cutting tools [7-9] and also where accessibility has been addressed by the “C-Space” method [10]; Laser scanning, where visibility helps determine a sufficient set of the scanning orientations [11] and molding design, where visibility is used to determine parting lines [12-14]. Also, visibility is widely used in the field of computer graphics, where shadow computation, global illumination, point-based rendering and view-dependent mesh-simplification are good examples [15, 16]. It is necessary to point out, although visibility is not explicitly addressed in some literature tackling CNC process planning, it is the prerequisite and efficiently narrows down the design space of more advanced planning strategies. For example, optimization of the toolpath smoothness [17] and optimization of the workpiece setups for 5-axis CNC machining for energy-saving purpose [18]. Moreover, with the advancement of AM technologies, innovative methods using multidirectional deposition processes are emerging [19]. Unlike conventional AM processes that use a single building direction, multidirectional deposition has increased degrees of freedom and therefore visibility will play an important role in facilitating the process planning of these new systems.

1.3 Tool Accessibility Concept

Tool accessibility of a part’s surface is the collection of directions from which a tool can access the entirety of the surface without colliding with the part. It is different from visibility in that it includes the body of the tool in collision requirements instead of a line as in visibility.

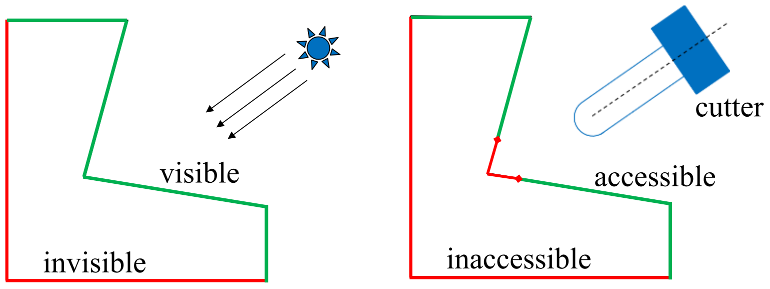


Fig. 1-5 Visibility vs. Accessibility

Tool accessibility is based on a more accurate modelling of a machining tools' movement than visibility. The limitation of using visibility for feasible tool orientations is obvious: a surface that is visible from a direction is not necessarily accessible by the tool (Fig. 1-5). This is because visibility only provides accessibility for the center line of the tool. Thus, the other space that the tool occupies could easily collide with the part. As such, visibility is in fact an overestimate of feasible tool orientations. Accordingly, the setups derived from visibility results do not necessarily expose the entire part surface to the cutting tool; an incomplete cutting situation. To solve this problem, the tool body must be considered in collision, as is in accessibility. However, the two concepts are also related. One can treat visibility as an extreme case of tool accessibility where the tool is infinitely thin. In fact, accessibility can be derived from visibility for ball-end tools, as will be shown in Chapter 4.

1.4 Motivation

Using Accessibility in Setup Planning

Intuitively, the more complex the part geometry is, the more difficult it is to make; accessibility of a part is one of the useful indicators of difficulty. It provides a requirement of tool orientations for the machine to match. Because a machine has its orientation limits, defined by Tool Access Directions (TAD), it is therefore straightforward that we want to match a part's accessibility with a machine's TAD. Because of the setup, a part's accessibility is in different coordinate system from a machine's TAD ((Workpiece Coordinate System (WCS)) versus Table Coordinate System (TCS)). We can compare them by transforming the TAD into the WCS (Fig. 1-6, some subfigures are from [4]). In WCS, the overlapped region represents the tool directions that are required by the part's surface and also satisfied by the machine. We can see that one specific setup leads to one specific orientation of TAD in WCS. Changing the setup, the TAD may overlap with accessibilities from different surfaces. Therefore, we can determine whether a setup renders some surfaces accessible or whether a collection of setups render all part surfaces accessible.

In summary, a part's accessibility results can help determine its feasible setups which is why we are interested in accessibility (and visibility) computation methods.

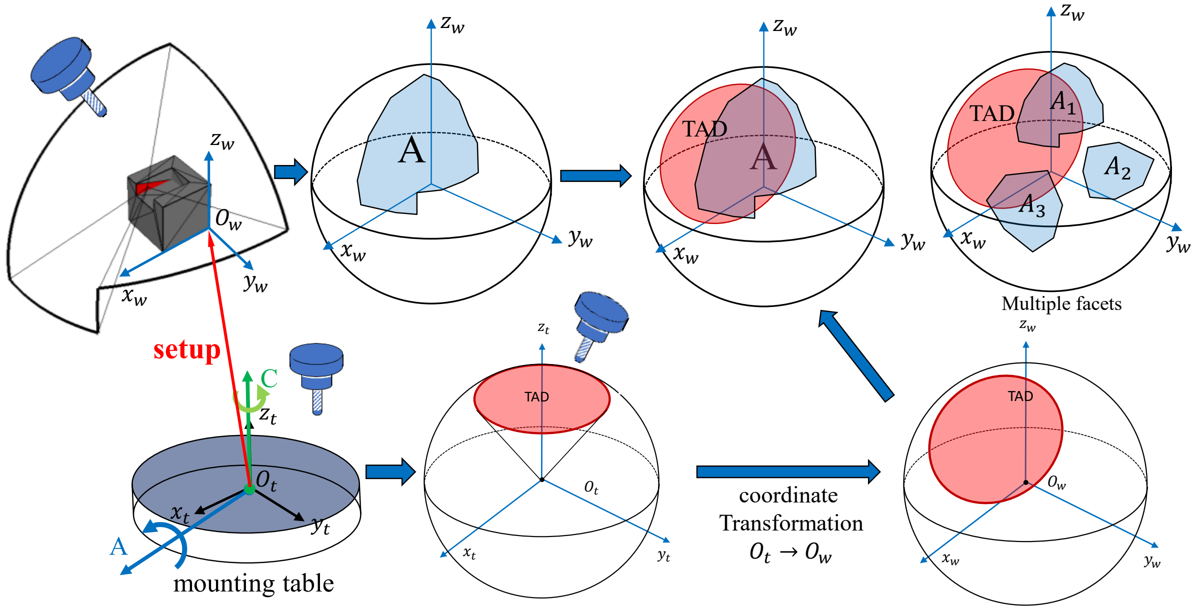


Fig. 1-6 A part's accessibility and a machine's Tool Access Directions

As part models are becoming increasingly complicated today, the efficiency of computing visibility and accessibility is emphasized. An efficient visibility/accessibility determination method will also facilitate rapid prototyping processes that use CNC machining, 3D printing or both. Therefore, it is highly desirable that we can compute visibility and accessibility under reasonable time cost while maintaining some degree of accuracy.

1.5 References

[1] Sandvik, 2019, "Milling tools,"https://www.sandvik.coromant.com/en-gb/products/pages/milling-tools.aspx, p. Solid milling tools and exchangeable tips.

[2] Haas-Automation, 2019, "5-axis Universal Machines,"https://www.haascnc.com/machines/vertical-mills/universal-machine.html, pp. 5-axis Universal Machines.

[3] Martínez, Q., 2018, "CNC machining workshop in Valencia,"http://umesal.com/taller-de-mecanizado-cnc-en-valencia/, p. The most efficient machining.

[4] Hu, P., Tang, K., and Lee, C.-H., 2013, "Global obstacle avoidance and minimum workpiece setups in five-axis machining," Comput Aided Design, 45(10), pp. 1222-1237.

[5] Haas-Automation, 2019, "UMC-750,"https://www.haascnc.com/machines/vertical-mills/universal-machine/models/umc-750.html#gsc.tab=0, pp. 5-axis trunnion table.

[6] Kweon, S., and Medeiros, D. J., 1998, "Part orientations for CMM inspection using dimensioned visibility maps," Comput Aided Design, 30(9), pp. 741-749.

[7] Wang, N., and Tang, K., 2007, "Automatic generation of gouge-free and angular-velocity-compliant five-axis toolpath," Comput Aided Design, 39(10), pp. 841-852.

[8] Balasubramaniam, M., Sarma, S. E., and Marciniak, K., 2003, "Collision-free finishing toolpaths from visibility data," Comput Aided Design, 35(4), pp. 359-374.

[9] Suh, S.-H., and Lee, J.-J., 1998, "Five-Axis Part Machining With Three-Axis CNC Machine and Indexing Table," Journal of Manufacturing Science and Engineering, 120(1), pp. 120-128.

[10] Morishige, K., Takeuchi, Y., and Kase, K., 1999, "Tool Path Generation Using C-Space for 5-Axis Control Machining," Journal of Manufacturing Science and Engineering, 121(1), pp. 144-149.

[11] Elber, G., and Zussman, E., 1998, "Cone visibility decomposition of freeform surface," Comput Aided Design, 30(4), pp. 315-320.

[12] Fu, M. W., 2008, "The application of surface demoldability and moldability to side-core design in die and mold CAD," Comput Aided Design, 40(5), pp. 567-575.

[13] Chen, L.-L., Chou, S.-Y., and Woo, T. C., 1995, "Partial visibility for selecting a parting direction in mold and die design," J Manuf Syst, 14(5), pp. 319-330.

[14] Priyadarshi, A. K., and Gupta, S. K., 2004, "Geometric algorithms for automated design of multi-piece permanent molds," Comput Aided Design, 36(3), pp. 241-260.

[15] Bittner, J., and Wonka, P., 2003, "Visibility in computer graphics," Environ Plann B, 30(5), pp. 729-755.

[16] Zach, C., and Karner, K., 2003, "Progressive compression of visibility data for view-dependent multiresolution meshes," Wscg'2003, Vol 11, No 3, Conference Proceedings, pp. 546-553.

[17] Lu, Y., Ding, Y., and Zhu, L., 2016, "Smooth Tool Path Optimization for Flank Milling Based on the Gradient-Based Differential Evolution Method," Journal of Manufacturing Science and Engineering, 138(8), pp. 081009-081009-081011.

[18] Xu, K., and Tang, K., 2016, "Optimal Workpiece Setup for Time-Efficient and Energy-Saving Five-Axis Machining of Freeform Surfaces," Journal of Manufacturing Science and Engineering, 139(5), pp. 051003-051003-051016.

[19] Song, X., Pan, Y., and Chen, Y., 2015, "Development of a Low-Cost Parallel Kinematic Machine for Multidirectional Additive Manufacturing," Journal of Manufacturing Science and Engineering, 137(2), pp. 021005-021005-021013.

CHAPTER 2. LITERATURE REVIEW

2.1 Visibility Computation

There has been a considerable amount of work addressing the problem of visibility computation. Among them, the seminal work in this field is attributed to Chen and Woo [1] where a Gaussian Map was described. The basic idea is to compute a dual image of the Gaussian map on a unit sphere. Later, methods using the Gaussian Map were then applied to compute setup orientations for 4 and 5 axis machining [2, 3]. However, visibility obtained from a Gaussian map is local, ignoring the fact that the visibility of a designated surface might be occluded by other surfaces. Therefore, such methods are limited to the visibility of certain features of the component [4]. Suh and Kang obtained a global visibility map by discretizing the visibility sphere into spherical triangles and using an occupation test to obtain the visibility cone [5]. One drawback of this method is that it uses the centroid to approximate the triangle, resulting in an approximated visibility. Besides, the occupation test leads to inefficient computation due to the enumeration of discretized directions. Li and Frank introduced a boundary tracing method that computes non-visibility between a pair of polyhedral facets [6]. Though global visibility is obtained, the computation time complexity of $O(n^2)$ remains a challenge, where n is the number of facets of the model.

The methods to compute visibility can be classified into two categories: approximate methods and exact methods. An approximate method calculates visibility in finite resolution at a reasonable computational cost. On the other hand, an exact method calculates the exact visibility image using sophisticated geometry, usually at a relatively more expensive computational cost.

Among the approximate category, there is one type of method that uses the hidden surface removal technique. They create two discretized surfaces, one for the part and one for the visibility space. Then, a mapping from the discrete surface of the part to the discrete space of visibility is built. One way to build such a mapping is by ray-casting [7]. Some others use a Z-buffer method, created through graphics hardware [8]. Another approximate method makes use of the slice geometry, where the 3D part is sliced into a set of 2D cross sections. The visibility of the 3D part is then derived from the visibility of the 2D slices [9-11]. The drawbacks of the approximate methods include inaccuracy and incompleteness of visibility results. The approximation in computation might lead to an underestimate or overestimate of visibility and the discretization might render the results incomplete.

Within the exact category, Dhaliwal et al. proposed a method that essentially conducts an occlusion calculation between a pair of triangular facets [12]. Liu and Ramani extended the work to the occlusion calculation between a pair of convex facets [13]. A similar method by Li and Frank computes pairwise occlusion using a boundary tracing technique [6]. These approaches share a common bottleneck: due to the pairwise occlusion computation, the algorithm's time complexity is quadratic to mesh size. Besides, the union operation among all spherical polygons (pairwise occlusion results) before generating the final visibility is an expensive computational task. To improve the latter's efficiency, Liu and Ramani further extended their work by introducing the use of Minkowski sum [14].

2.2 Tool Accessibility Computation

Miller investigated the application of surface accessibility on the visual effect of shading [15]. In which, accessibility is defined as radius of a sphere which may touch a surface point and not intersect any surface. The results highlight the inaccessible surface by a spherical probe. However, the method only gives a Boolean answer to whether a surface is accessible; it does not provide actual access orientations. Elber proposed a method to determine the inaccessible surface induced by flat-end tools in 5-axis machining [16]. However, the method is restricted to inaccessibility of convex surfaces due to other check surfaces and does not provide feasible tool orientations for the accessible surfaces. Tang et al. proposed a surface offset/upper envelope method that solves gouging for 3-axis multi-surface Numerical Control (NC) machining [17]. Kim et al. proposed a triangular mesh offset algorithm for tool path planning of generalized cutters in NC machining [18]. However, both Tang and Kim's work focuses on 3-axis NC machining where tool orientation is not considered. Xu et al. proposed a method that determines feasible tool orientations for each predefined cutter contact (CC) point in 5-axis NC machining [19]. However, CC points and CC paths must be provided first. This is not desirable if we want to optimize the tool path choice. This method works better in the case where a CC path is already determined, and CC points are used to generate gouge-free and collision-free tool orientations. Alternatively, it can be used to verify if a given toolpath is gouge and collision free.

2.3 Research Problem and Objectives

Research Problem

After reviewing the literature, there does not exist a visibility method that has the capacity to incorporate both exact and approximate visibility computations. Such flexibility enables the balancing between processing time and accuracy. Besides, there is no work that derives accessibility directly from visibility. With such a method, we can adopt various existing visibility methods for accessibility.

Research Objectives

To solve these research problems, the objective of this dissertation is to develop three new computational methods:

1. A method that computes the approximate **visibility** map efficiently.
2. A method that computes the tool **accessibility** map based on visibility results.
3. A method that computes the **near-exact** visibility map while balancing processing time and accuracy using an approximate visibility method for preprocessing.

2.4 References

[1] Chen, L. L., and Woo, T. C., 1992, "Computational Geometry on the Sphere with Application to Automated Machining," J Mech Design, 114(2), pp. 288-295.

[2] Tang, K., Woo, T., and Gan, J., 1992, "Maximum Intersection of Spherical Polygons and Workpiece Orientation for 4-Axis and 5-Axis Machining," J Mech Design, 114(3), pp. 477-485.

[3] Chen, L. L., Chou, S. Y., and Woo, T. C., 1993, "Separating and Intersecting Spherical Polygons - Computing Machinability on 3-Axis, 4-Axis and 5-Axis Numerically Controlled Machines," Acm T Graphic, 12(4), pp. 305-326.

[4] Chen, L.-L., Chou, S.-Y., and Woo, T. C., 1993, "Parting directions for mould and die design," Comput Aided Design, 25(12), pp. 762-768.

[5] Suh, S. H., and Kang, J. K., 1995, "Process Planning for Multiaxis Nc Machining of Free Surfaces," Int J Prod Res, 33(10), pp. 2723-2738.

[6] Li, Y., and Frank, M. C., 2007, "Computing non-visibility of convex polygonal facets on the surface of a polyhedral CAD model," Comput Aided Design, 39(9), pp. 732-744.

[7] Tarbox, G. H., and Gottschlich, S. N., 1995, "Planning for Complete Sensor Coverage in Inspection," Comput Vis Image Und, 61(1), pp. 84-111.

[8] Spitz, S. N., and Requicha, A. A. G., 2000, "Accessibility analysis using computer graphics hardware," Ieee T Vis Comput Gr, 6(3), pp. 208-219.

[9] Frank, M. C., Wysk, R. A., and Joshi, S. B., 2006, "Determining setup orientations from the visibility of slice geometry for rapid computer numerically controlled machining," J Manuf Sci E-T Asme, 128(1), pp. 228-238.

[10] James Stewart, A., 1999, "Computing visibility from folded surfaces," Computers & Graphics, 23(5), pp. 693-702.

[11] Suthunyatanakit, K., Bohez, E. L. J., and Annanon, K., 2009, "A new global accessibility algorithm for a polyhedral model with convex polygonal facets," Comput Aided Design, 41(12), pp. 1020-1033.

[12] Dhaliwal, S., Gupta, S. K., Huang, J., and Priyadarshi, A., 2003, "Algorithms for Computing Global Accessibility Cones," Journal of Computing and Information Science in Engineering, 3(3), pp. 200-209.

[13] Liu, M., and Ramani, K., 2007, "Computing an exact spherical visibility map for meshed polyhedra," Proceedings of the 2007 ACM symposium on Solid and physical modeling, ACM, Beijing, China, pp. 367-372.

[14] Liu, M., Liu, Y. S., and Ramani, K., 2009, "Computing global visibility maps for regions on the boundaries of polyhedra using Minkowski sums," Comput Aided Design, 41(9), pp. 668-680.

[15] Miller, G., 1994, "Efficient algorithms for local and global accessibility shading," Proceedings of the 21st annual conference on Computer graphics and interactive techniques, ACM, pp. 319-326.

[16] Elber, G., 1994, "Accessibility in 5-axis milling environment," Comput Aided Design, 26(11), pp. 796-802.

[17] Tang, K., Cheng, C. C., and Dayan, Y., 1995, "Offsetting surface boundaries and 3-axis gouge-free surface machining," Comput Aided Design, 27(12), pp. 915-927.

[18] Kim, S.-J., and Yang, M.-Y., 2005, "Triangular mesh offset for generalized cutter," Comput Aided Design, 37(10), pp. 999-1014.

[19] Xu, X. J., Bradley, C., Zhang, Y. F., Loh, H. T., and Wong, Y. S., 2002, "Tool-path generation for five-axis machining of free-form surfaces based on accessibility analysis," Int J Prod Res, 40(14), pp. 3253-3274.

CHAPTER 3. COMPUTING THE GLOBAL VISIBILITY MAP USING SLICE GEOMETRY FOR SETUP PLANNING

A paper published as *Hou G, Frank MC. Computing the Global Visibility Map Using Slice Geometry for Setup Planning. ASME. J. Manuf. Sci. Eng. 2017;139(8):081006-081006-11. doi:10.1115/1.4036423.*

Guangyu Hou[1]
Department of Industrial and Manufacturing System Engineering,
Iowa State University,
3023 Black Engineering,
Ames, IA, 50011
e-mail: houes@iastate.edu

Matthew C. Frank
Department of Industrial and Manufacturing System Engineering,
Iowa State University,
3023 Black Engineering,
Ames, IA, 50011
e-mail: mfrank@iastate.edu

Abstract

This paper introduces a new method that uses slice geometry to compute the Global Visibility Map (GVM). Global Visibility Mapping is a fundamentally important process that extracts geometric information about an object which can be used to solve hard problems; for example, the setup and process planning in CNC machining. In this work, we present a method for creating the GVM from slice data of polyhedron models, and then show how it can help determine around which axis of rotation a part can be machined. There have been various methods of calculating the GVM to date, tracing back to the well-known seminal methods that use Gaussian Mapping. Compared to the considerable amount of work in this field, the proposed method has an advantage of starting from feature-free models like STL files and has adjustable resolution. Moreover, since it is built upon slicing the model, the

[1] Corresponding author.

method is embarrassingly parallelizable in nature, thus suitable for high-performance computing. Using the GVM obtained by this method, we generate an axis of rotation map to facilitate the setup planning for 4-axis CNC milling machines as one implementation example.

Keywords: *Global Visibility Map (GVM), Slice Geometry, Axis of rotation, Setup Planning, CAD/CAM, 4-axis CNC Machining, Parallel Computing*

3.1 Introduction

Conceptually speaking, visibility is the quantified measurement of the extent that a geometric entity (point, surface, object, *etc.*) can be viewed or accessed from a distance considering other entities as obstacles. Specifically, it is a set of all possible lines of sight that are able to reach the geometric entity. Since only the orientations of these lines of sight matter to describe visibility, visibility is often depicted on a unit sphere, using a portion of the spherical surface to denote the visible area, as shown in Fig. 3-1. If the center of the sphere is connected to the boundary of the spherical visible area with radii, the resultant 3D geometry is referred to as a *visibility cone* in some literature.

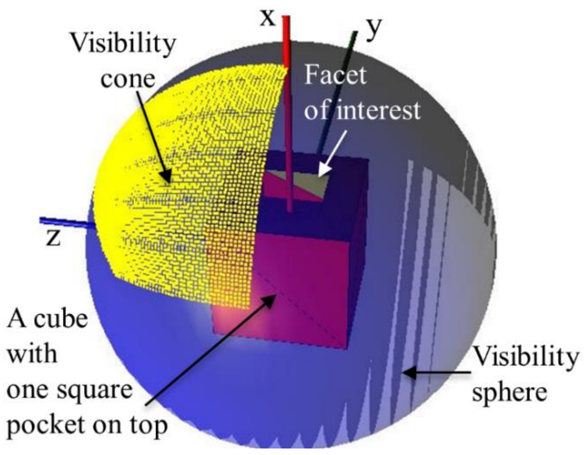


Fig. 3-1 An example showing the visibility cone of a facet on the pocket of the cube

Visibility is used in many technological fields, especially where the geometry of the surface is critical to their applications. Such technologies include Coordinate Measurement Machines (CMMs), where visibility determines the accessibility of the contact probe [1]; CNC machining, where visibility serves as the necessary condition for the accessibility of the cutting tools [2-4] and also where accessibility has been addressed by the “C-Space” method [5]; Laser scanning, where visibility helps determine a sufficient set of the scanning orientations [6] and molding design, where visibility is used to determine parting lines [7-9]. Also, visibility is widely used in the field of computer graphics, where shadow computation, global illumination, point-based rendering and view-dependent mesh-simplification are good examples [10, 11]. It is necessary to point out, although visibility is not explicitly addressed in some literature tackling CNC process planning, it is the prerequisite and efficiently narrows down the design space of more advanced planning strategies. For example, optimization of the toolpath smoothness [12] and optimization of the workpiece setups for 5-axis CNC machining for the energy-saving purpose [13]. Moreover, with the advancement of AM technologies, innovative methods using multidirectional deposition processes are emerging [14]. Unlike conventional AM processes that use a single building direction, multidirectional deposition has increased degrees of freedom and therefore visibility will play an important role in facilitating the process planning of these new systems.

3.2 Related Work

There has been a considerable amount of work addressing the problem of visibility computation. Among them, the seminal work in this field is attributed to Chen and Woo [15] where a Gaussian Map was described. The basic idea is to compute a dual image of the Gaussian map on a unit sphere. Later, methods using the Gaussian Map were then applied to compute setup orientations for 4- and 5- axis machining [16, 17]. However, visibility obtained from a Gaussian map is local, ignoring the fact that the visibility cone of a designated entity might be occluded by other surfaces. Therefore, such methods are limited to the visibility of certain features of the component [18]. Suh and Kang [19] obtained a global visibility map by discretizing the visibility sphere into spherical triangles and used an occupation test to obtain the visibility cone. One drawback of this method is that it uses the centroid of the triangle to approximate the triangle, resulting in an approximated visibility. Also, the occupation test leads to an inefficient computation. Li and Frank [20] introduced an approach that computes visibility by an occlusion computation between a pair of polyhedral facets. Though global visibility is obtained, the time complexity of $O(n^2)$ remains a challenge, where n is the number of facets.

The methods to compute visibility can be classified into two categories; approximate solution methods and exact solution methods. The approximate solution methods attempt to capture most of the visibility area at a reasonable computational cost. On the other hand, exact solution methods compute the exact visibility image using sophisticated geometry, usually at an expensive computational cost.

Among the approximate solutions category, hidden surface removal methods create two discretized surfaces, one for the component and one for the visibility space. Then, a mapping from the discrete surface of the component to the discrete visibility space is built. One technique to build such a mapping is to use ray-casting [21], while others use a z-buffer method, created through graphics hardware [22]. Another approximation method makes use of the slice geometry, where the 3D component is sliced into a set of 2D cross sections. The visibility of the 3D component is then derived from the visibility of the 2D slices [23-26]. One drawback of the approximated methods is their impaired visibility accuracy. The discretization impacts visibility accuracy and the visibility cone is often underestimated.

Within the exact solution category, Dhaliwal, Gupta et al. [27] proposed a method that essentially conducts an occlusion calculation between a pair of triangle facets. Liu and Ramani [28] extended the work to the occlusion calculation between a pair of convex facets. A similar method by Li and Frank [20] computes pairwise occlusion using a boundary tracing approach. These approaches share a common bottleneck: due to pairwise occlusion computing, the algorithm's time efficiency is driven by the size of the mesh. Also, the union operation among all spherical polygons before generating the GVM is an expensive computational task. To improve efficiency in the latter problem, Liu and Ramani [29] further extended their work by introducing the Minkowski sum.

3.3 Methodology

The proposed method assumes a triangulated model (STL) as input. Thus, the objective is to determine the visibility cone of each triangular facet. The visibility of any polyhedron surface is easily derived from its constituent facets; a simple intersection. The method is an approximate solution method in that the visibility will only be evaluated at a finite number of positions: the unit visibility sphere is discretized into a set of intensely distributed points located at the intersections of evenly spaced longitude and latitude lines (Fig. 3-2). The angular spacing between both longitudinal and latitudinal lines is set to one degree by default. Thus, a total of 64,442(=179*360+2) points are

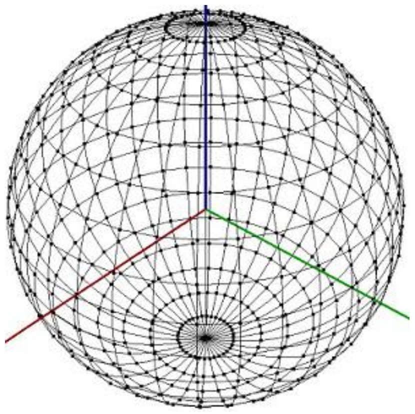


Fig. 3-2 The discretization of the visibility sphere

sampled to represent the entire visibility sphere. As the resolution is controlled by the number of sampling points, it is easily adjustable.

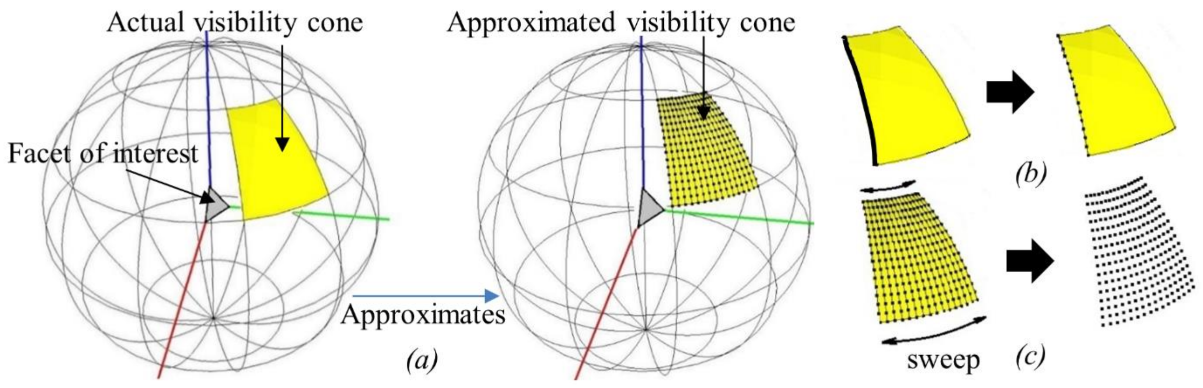


Fig. 3-3 An example of a 3D visibility cone built up by a set of 2D visibility arcs. (a) an actual visibility cone is approximated by a set of arcs/points, (b) a visibility arc is represented by a set of visibility points, (c) a sweep of the visibility arcs yields the visibility cone. In this paper, the continuous visibility arcs are showed by a set of points

To compute the visibility cone, the method uses the fact that, as an approximation, a 3D visibility cone can be discretized into a finite set of 2D visibility arcs (Fig. 3-3a). Thus, the task is further simplified to compute a set of visibility arcs. In general, the visibility cones can appear anywhere and may exist disconnected in multiple places on the sphere, so do the constituent visibility arcs. Therefore, the visibility arcs should be evaluated in places that cover the entire sphere. To do this, the 2D visibility arcs are chosen to be evaluated on 360 uniformly

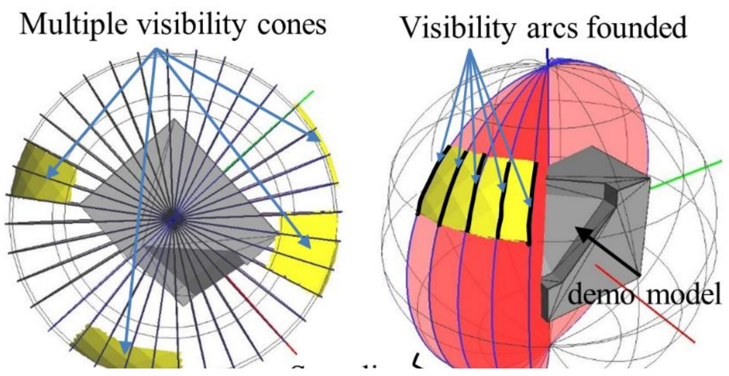


Fig. 3-4 The distribution of sampling longitude lines where visibility arcs are evaluated. (a) top view showing evenly spaced sampling longitude lines captures multiple visibility cones, (b) isometric view showing five visibility arcs of a visibility cone discovered by the densely angular spaced sampling longitude lines

spaced longitude lines throughout the sphere (Fig. 3-4a). Because of this dense sampling, most visibility cones can be accurately captured (Fig. 3-4b). This sampling scheme of visibility arcs matches with the former discretization of the visibility sphere.

With the visibility arcs' sampling scheme determined, the next critical step is to actually compute the visibility arcs on the sampling longitude lines, namely determining the visible portion of the longitude lines. It is obvious that each visibility arc resides on the plane defined by the longitude line. When such plane rotates, the visibility arcs should change accordingly. The solution to the problem can then be simplified into two phases. The first phase is to compute the 2D visibility arcs on each sampling plane and the second phase is to assemble these 2D visibility arcs from all sampling planes.

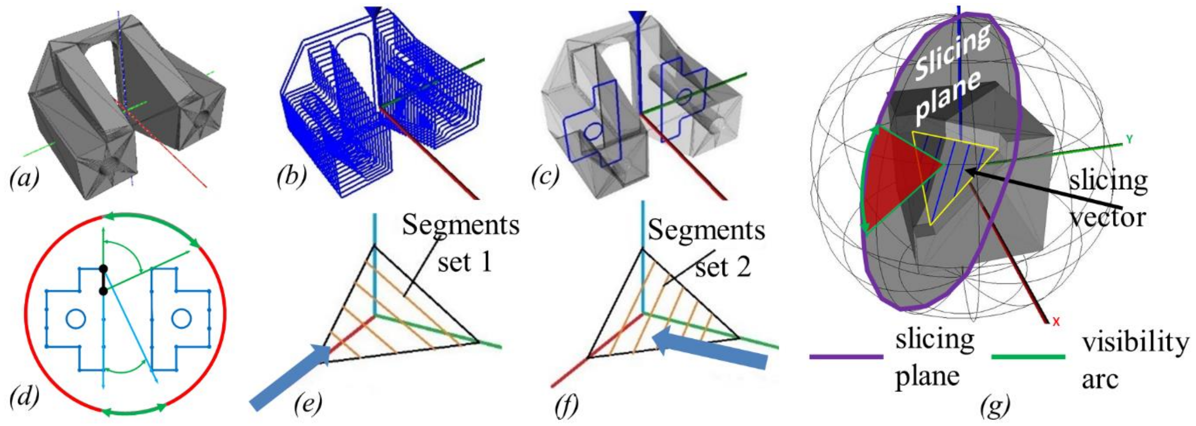


Fig. 3-5 The slice-based visibility computation showed by an example. (a) an example part, (b) the slice geometry, (c) one slice for demonstration, (d) visibility computation for a segment on the slice chain, (e-f) example where sliced segments changes with the slicing direction, (g) example showing the visibility arc resides on the plane perpendicular to the slicing direction

To obtain the 2D visibility arcs, this paper utilizes the visibility algorithm developed by Frank et al. [23], and further extends it to a facet-based visibility algorithm. The earlier version of this visibility algorithm represents the 3D object as multiple 2D slices. It assumes the viewer resides on a plane parallel to those 2D slices. It then computes the visibility within such plane for all segments in the slices (Fig. 3-5a-d). The reason that the visibility scope is restricted in a plane is that in applications like 4-axis CNC milling machine, once the 4th axis (rotation axis) is chosen, the accessibility of the cutting tools is restricted in a plane (in workpiece coordinate system). Thus, such visibility is sufficient to tell whether the component is visible from the cutting tool.

In the algorithm mentioned above, the plane where visibility arcs lie is perpendicular to the pre-defined slicing direction (Fig. 3-5g). To apply it to the method in this paper, where we evaluate visibility arcs on planes containing longitude lines, the slicing vectors must lie in the XY-plane. Consequently, when we rotate the slicing vector in the XY-plane with a one-degree interval, the visibility arcs can then be computed on uniformly angular-spaced longitude lines as planned earlier. However, segment-based visibility exposes its limitations when we try to assemble such results: because the sliced segments of a given facet changes while slicing direction changes (Fig. 3-5e and Fig. 3-5f), the two visibility results cannot combine directly. To solve this, we need a fixed geometry whose visibility result can be divided and computed in different planes separately and combined in 3D afterward. This requirement leads to the facet-based visibility.

In this paper, the visibility of a facet is defined as the intersection of visibility of its containing segments – a necessary condition (Fig. 3-6). Such definition is an approximation: since in the ideal situation, the number of containing segments should reach infinity to be exact. In fact, because of the finite number of segments, this definition assumes the visibility bounds remains constant for a certain distance in the slicing direction (Fig. 3-6a). Therefore, the visibility bounds of a segment are equivalent to the visibility bounds of a corresponding area.

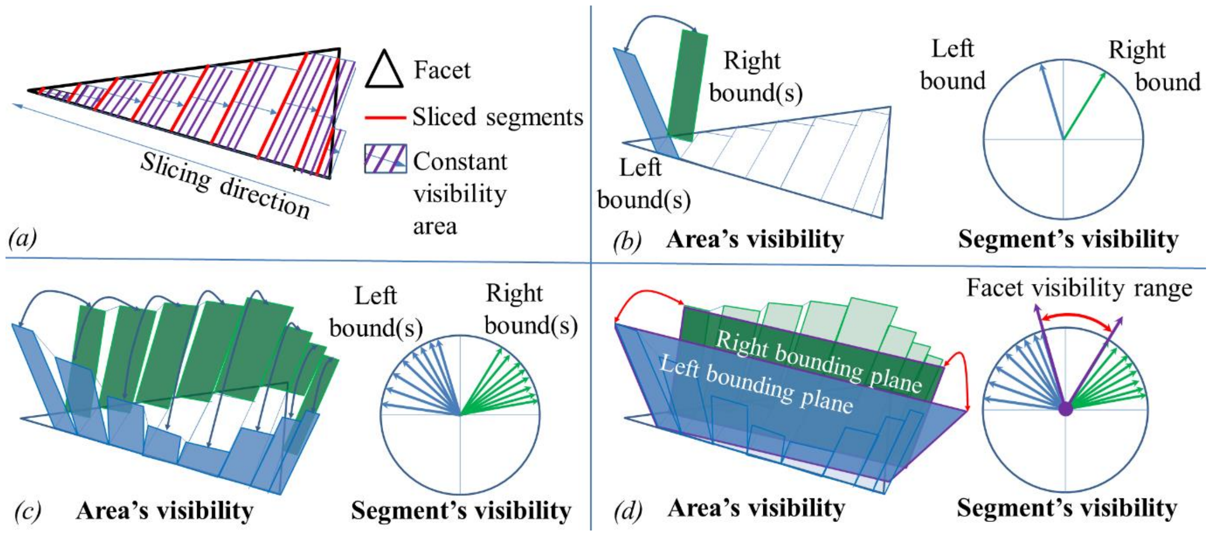


Fig. 3-6 The definition of facet visibility. (a) an example showing segment's visibility is constant for a distance due to approximation, (b-d) the formation of the facet visibility by intersecting visibility ranges of its containing segments

Using a facet as the visibility source enables us to combine visibility arcs resulting from different sampling planes. The facet remains fixed while the slicing plane (or vector) rotates. Nevertheless, such facet-based visibility erases certain visibility conveyed by segments due to the intersection operation. For example, if one out of ten segments is invisible and the other nine are visible, the intersection of visibility indicates the facet is invisible. Clearly, visible area, no matter the proportion, is ignored due to the intersection. To retain as much visibility as possible, it is necessary to reduce the difference of visibility among segments in a facet. In general, the smaller the facet, the less likely this kind of visibility loss occurs since segments in a smaller facet have more similar geometric surroundings. However, smaller facets lead to a denser mesh, thus a higher computational cost. A threshold should be set to balance the trade-off. To estimate the visibility loss due to intersection operation, we introduce the "weighted visibility." Weighted visibility is defined as the product of 2D visibility range and the area it is on. Thus, before the intersection operation, the aggregate weighted visibility of a facet is

$$\sum_{i=1}^{n} \theta_i \cdot A_i \tag{1}$$

Where n is the number of segments in a facet; θ_i, A_i are the 2D visibility range and the effective area of segment i, respectively. After the intersection operation, the weighted visibility of a facet is $\theta_{intersect} \cdot A_{facet}$. Therefore, the ratio

$$\boldsymbol{\rho} = 1 - \frac{\theta_{intersect} \cdot A_{facet}}{\sum_{i=1}^{n} \theta_i \cdot A_i} \tag{2}$$

is used to define the loss of visibility due to the intersection. In our implementation, we deem a facet's visibility is "under representative" if $\rho > 0.2$. If the total area of under representative facets is greater than 5% of the model's surface area, a global mesh refinement can be implemented. The choice of the thresholds 0.2 and 5% are empirical values with the consideration of computational capability. Theoretically, the smaller the thresholds, the better the result is expected (closer to real visibility). However, to keep the size of the mesh (thus the computational cost) to an

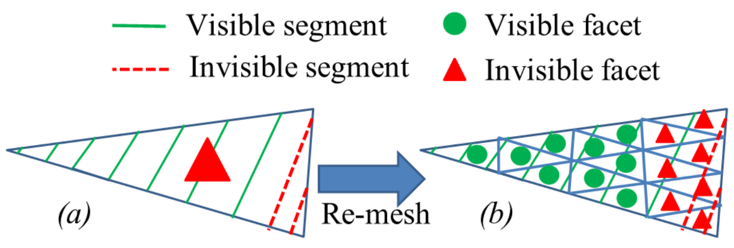


Fig. 3-7 An example of improved visibility after mesh refinement. (a) visibility loss ratio of the original facet is 1.0, (b) visibility loss ratio of the equivalent facets after refinement is 0.36. Assuming visibility range θ is constant for all visible segment

acceptable value (e.g. less than 20,000 facets), the proposed thresholds have been found to be reasonable. An example is given in Fig. 3-7 to show that mesh refinement reduces the amount of visibility loss due to visibility conversion from segment to facet.

Now consider a facet-based visibility arc is obtained in the YZ-plane using a slicing vector aligned with the X-axis. Rotating the slicing vector around the Z-axis gradually at a 1° interval through 179° will correspondently create visibility arcs around the globe (Fig. 3-8b-d). The union of these visibility arcs reveals the complete visibility cones (Fig. 3-8e). Note that for each individual slicing direction, visibility arcs are generated for all facets simultaneously. Thus, after the rotation, visibility cones are created for all the facets, namely the Global Visibility Map (GVM). In fact, the starting and ending slicing vector can be randomly chosen as long as they sweep 179 degrees. In this paper, the rotation is around the Z-axis and counterclockwise. The X-axis is the starting vector and the slicing vectors are chosen at 0, 1, 2 … 179 degrees (Fig. 3-8a).

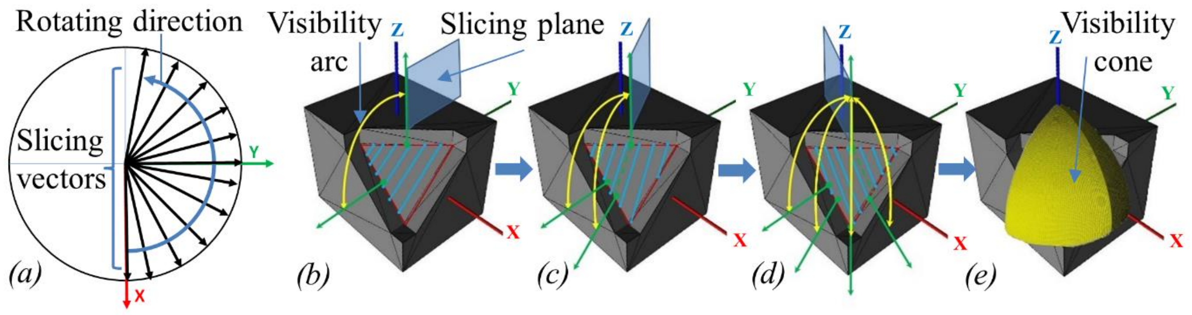


Fig. 3-8 Examples showing the union of visibility arcs from different slicing planes. (a) the orientations of slicing vectors, (b–d) visibility arcs generated from different slicing planes, (e) the union of visibility arcs from 180 sampling planes results in a complete visibility cone

The parallel planes challenge

One of the challenges using slice geometry is to address the “missing plane” problem caused by the model planes that are parallel to the slicing planes. These parallel planes are ignored in the slicing process. For example, in 3D printing, a part’s physical planes parallel to the slicing planes are always approximated by the cross-sections near them (less than a slice spacing above or below). Similarly, parallel planes fail to be captured and counted in the visibility computation since no segment is generated to represent those planes. Specifically, such deficiency may lead to an incomplete visibility cone that misses some constituent visibility arcs. One solution is to use a second set of slicing planes to slice the parallel planes specifically [30]. The second set of slicing planes (B-planes) are chosen to be perpendicular to the first set of slicing planes (A-planes), thus also perpendicular to the unsliced facets (Fig. 3-9). This ensures the facets missed by A-planes are properly sliced by B-planes. Here, we define “A-segments” as the segments sliced by A-planes and “B-segments” as the segments sliced by B-planes. The visibility computation for B-segments is different from that of A-segments. First, an obstacle chain has to be found for B-segments. Second, since a B-segment belongs to no chain in the parallel plane, it can be accessed from both sides.

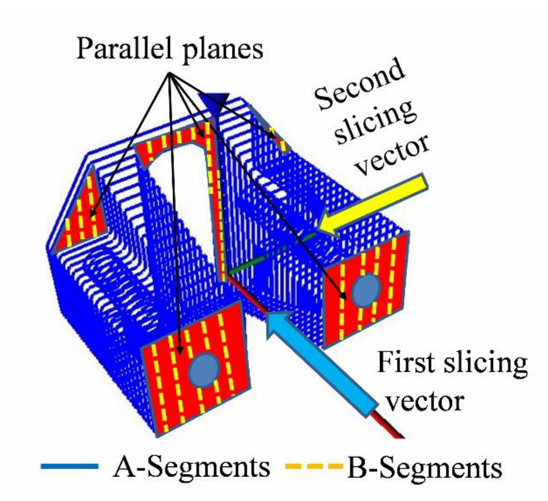


Fig. 3-9 The use of two sets of slices (perpendicular to each other) to solve the missing parallel planes (in red) problem

In this paper, we introduce a new method to compute the visibility of facets on parallel planes. Instead of using B-segments, which could be too many if the slicing interval is small, we use the three edges of the triangular facet. An example part is shown in Fig. 3-10a. The non-visibility for a triangle with respect to one obstacle chain is computed as the maximum angle difference of six bounding rays sent from the three vertices (Fig. 3-10c). Since the vertices of a triangle are shared by its neighboring triangles, the bounding rays are also shared (Fig. 3-10b). Ultimately, the task is simplified to obtaining the bounding rays of each vertex on the parallel plane with respect to the obstacle chains. Compared to B-segments, this method greatly saves computational time and is inherently an exact method (since we no longer approximate a triangle's visibility by its containing segments).

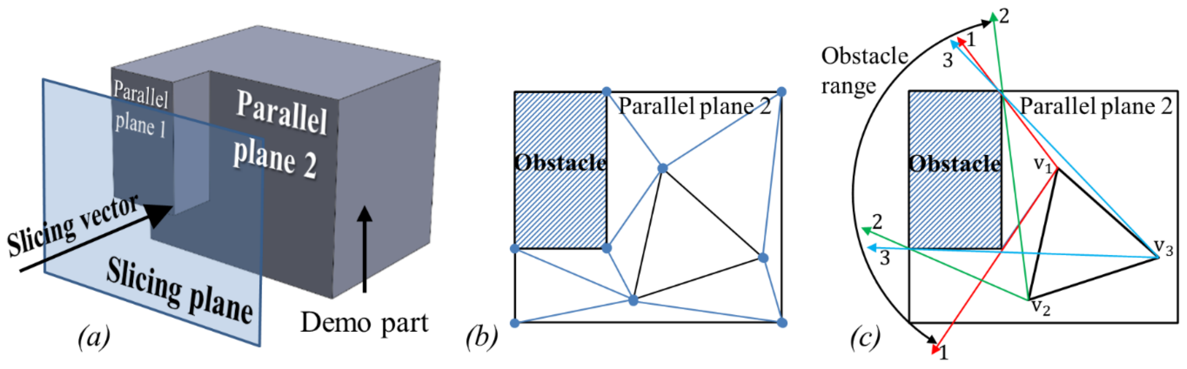


Fig. 3-10 An example of parallel plane visibility computation using the three edges of a triangle; (a) a demonstration part with its parallel planes marked, (b) a triangular facet on a parallel plane showing its vertices are shared by its neighboring triangles, (c) obstacle range obtained by finding the maximum angle difference among six bounding rays. Rays of the same color come from the same vertex.

Though efficient, this method should be implemented carefully considering the complex boundary cases. Ideally, the triangle in the parallel planes should always be on the exterior of the obstacle chain. In other words, none of the triangle's three vertices should be in the obstacle chain (as in Fig. 3-11a-e). However, in the boundary cases where at least one vertex is on the obstacle chain (Fig. 3-11b-d), due to numerical error in slicing (to obtain the obstacle chain), some of the triangle's vertices could get into the obstacle chain by some small value Δ (Fig. 3-11*f*-*h*). In such case, we call this vertex "false" interior with respect to the obstacle chain. “False” interior vertices lead to the problem that the bounding rays sending from such vertex will be erroneously computed. As can be seen in Fig. 3-11i, if the triangle’s vertex is on the exterior of the obstacle chain, the bounding rays are defined as rays of min and max accumulated angle (the triangle’s vertex to each vertex on the obstacle chain forms a ray). However, for a “false” interior vertex, the rays of min and max accumulated angle fail to capture the obstacle chain’s boundary. For example, in Fig. 3-11j, if we start counting angle from p_1 and traverse the obstacle chain counterclockwise, then after visiting all the vertices, the min and max rays will go through p_1 and p_0 respectively, which is an incorrect boundary. To solve this problem, all “false” interior vertices must be corrected. A vertex is identified as a “false” interior if it is contained by an outer obstacle chain meanwhile quite near the chain. It must be near because otherwise it could be an exterior vertex surrounded by the obstacle as seen in Fig. 3-11e. The extent of nearness is defined by the constant ϵ_1 (namely if $\Delta < \epsilon_1$, the vertex is near the chain). ϵ_1 should be large enough to tolerate numerical errors and small enough so that the vertex does not cross the obstacle and become an exterior vertex contained by the inner obstacle chain (Fig. 3-11k). In this paper, ϵ_1 is set to 10^{-6} because it is very rare that a part has a wall feature thinner than 10^{-6} inch and 10^{-6} detects most “false” interior vertices in implementation. Once “false” interior vertices are found, either the vertex is offset to the exterior of the outer obstacle chain or such triangle is offset inward by a small value ϵ_2 depending on how many of the three vertices are deemed “false” interior (Fig. 3-11k and Fig. 3-11ℓ). The choice of ϵ_2 has more freedom, where ϵ_2 should be greater than zero to make the vertex exterior, but small enough so that the angles of bounding rays are computed as if from a vertex on the obstacle chain. In practice, setting ϵ_2

to 10^{-6} has achieved both goals in implementation.

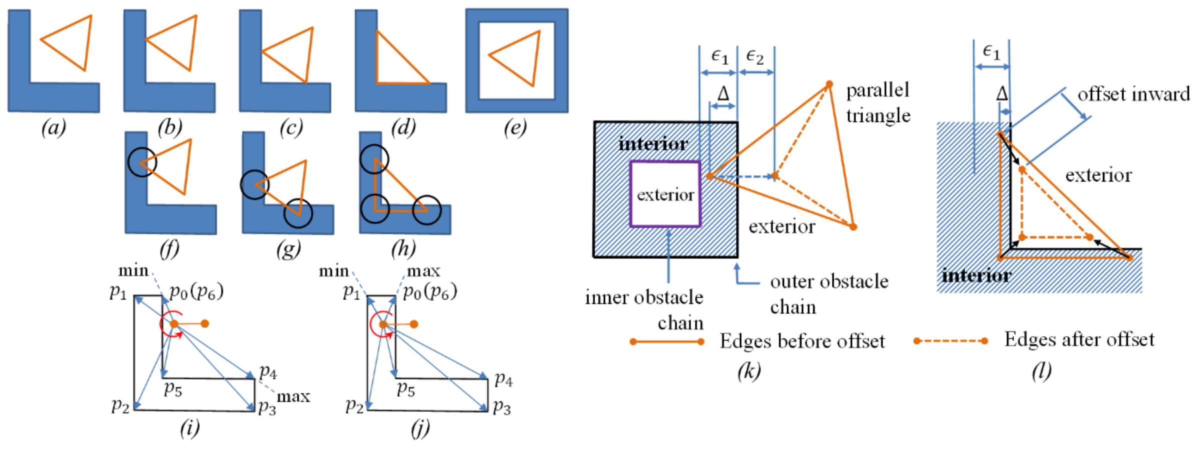


Fig. 3-11 The possible position relations of a triangle facet and its obstacle chains together with the solution to numerical error. The triangle's edges are in yellow. The obstacle chain is in blue. The triangle (a) is out of-, (b) has one vertex on-, (c) has two vertices on-, (d) has three vertices on-, (e) is contained by-, the obstacle chain. (f-h) due to numerical error, at least one vertex is contained in the obstacle chain, (i) the rays of min and max accumulated angle give the correct bounding rays if vertex is on the exterior, (j) the rays of min and max accumulated angle give the wrong bounding rays if vertex is in the interior, (k) offset the triangle's "false" interior vertex to the exterior by a small value ϵ_2, (l) offset the triangle inward to its centroid by a small value. The vertex is deemed "false" interior if $\Delta < \epsilon_1$.

3.4 Axis of Rotation Map

In CNC machining, it is necessary to determine a set of "setups" with which to fixture an object. Simple rectangular parts may require as few as one setup; however, in multiple setups, or even multi-axis setups, the problem is more challenging. For this work, we show how a GVM can lead to possible "axes of rotation" for an object. That is, if a 4th axis rotary (indexer) was available, what are the possible rotations of the part geometry about that axis yielding a potentially feasible solution? An example is given in Fig. 3-12 to show the correspondence between a visibility point in GVM and feasible axes of rotation. The rule is that a visibility point in GVM corresponds to axes on a great circle in the plane perpendicular to the visibility vector. A detailed explanation of this property has been properly shown in

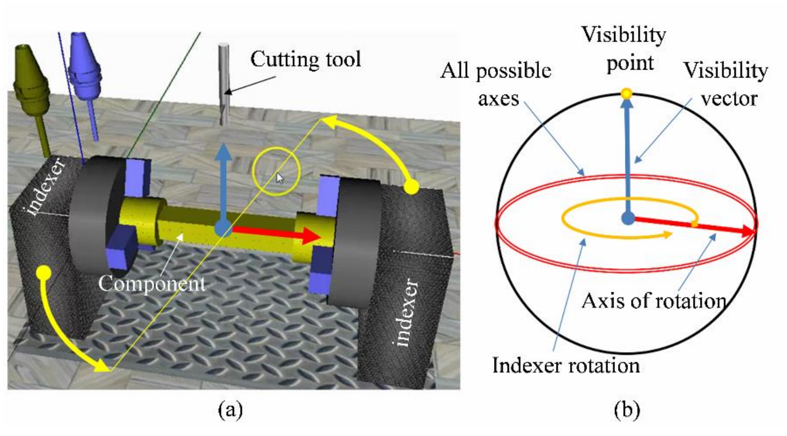


Fig. 3-12 The correspondence between a visibility point and feasible axes of rotation. (a) a real 4-axis CNC machine setup, (b) the feasible axes of rotation are any axes perpendicular to the visibility vector. Assuming the component is fixed while two indexers rotates to align with different axes

[31]. A simple explanation begins by noting that the cutting tool is *always* perpendicular to the axis of rotation. If the visibility vector is also perpendicular to the axis of rotation, then they are co-planar. Thus, the cutting tool can align with the visibility vector by a rotation. Accordingly, the tool and visibility point coincide, leading to the facet being seen by the tool. Granted, further analysis would need to ascertain whether the part workpiece can be clamped, whether the swept diameter is reachable, and whether the machine's envelope/travel can handle the part in a particular orientation. However, the mapping of "potential" axes of rotations is invaluable, as one could avoid re-clamping and re-positioning the workpiece if a singular axis could be found. If not, the mapping could still be valuable in determining a minimum set of axes with which a part can be machined. Given the GVM, such an axis of rotation map could be generated [31]. This axis of rotation map provides a feasible design space of the rotation axes for 4-axis CNC milling machines.

For a facet, the feasible axis of rotation provides how the facet should be oriented to align with the CNC machine's 4th axis for it to be visible from the cutting tool. The first step is to find the corresponding axes of rotation for each visibility point on the visibility cone. As shown in Fig. 3-13*a*, a region of feasible axes of rotation is generated for one visibility cone in the GVM. Note that a facet might have multiple visibility cones. Thus, its corresponding axes of rotation region could be very complex (e.g. in the shape of a union of multiple rings). Clearly, if an axis of rotation is shared by all the facets of the model, it is the axis that makes the entire model visible – a feasible axis for the model. In other words, intersecting the axes regions obtained from all facets of the model yields the feasible axes for the model (Fig. 3-13b shows an example of axes region intersection between two facets).

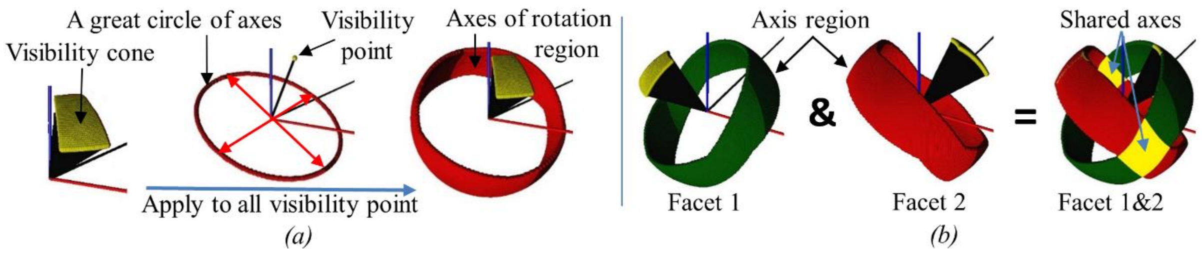


Fig. 3-13 The procedure of finding feasible axes of rotation from the GVM. (a) find corresponding axes of rotation region of a visibility cone, (b) an example showing intersecting axes' regions of two facets to obtain the share axes of rotation region

In summary, the GVM provides the map from facets to visibility cones. Using the rule in Fig. 3-13*a* provides the map from the visibility cones to the axes of rotation. A concatenation of the two maps generates the "facets to axes of rotation" map. To see how many facets each axis reveals, a reverse mapping on the "facets to axes of rotation" map is generated (resulting in an "axes to facets" map shown in Fig. 3-14a). For each axis, the reverse mapping collects every facet that maps to it in the original map. It is different from an inverse function because our map is one-to-many. It is also convenient to see the surface area each axis reveals by summing the corresponding facet areas. Such axes to visible surface areas mapping is the "axes of rotation" map for the model where the visible surface area is normalized by the model's total surface area (thus with range [0,1]). For display purpose, color is used to show the visibility of each axis, where red and blue denotes the highest and lowest visibility respectively, an example is given in Fig. 3-14b. As such, all axes of rotation are represented as 3D dots on the upper hemisphere. The color of the 3D dot indicates the visibility of the corresponding axis. Only the upper hemisphere is used because axes are symmetric about the origin.

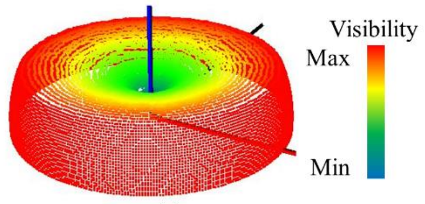
Map: facets → axes

$$\begin{bmatrix} f_1\\ f_2\\ f_3\\ \vdots\\ f_n\end{bmatrix} \begin{matrix}\rightarrow\\ \rightarrow\\ \rightarrow\\ \rightarrow\\ \rightarrow\end{matrix} \begin{bmatrix} axis_1 & axis_2 & \cdots & axis_k\\ axis_3 & axis_5 & \cdots & axis_l\\ axis_4 & axis_1 & \cdots & axis_a\\ & \vdots & \ddots & \vdots \\ axis_7 & axis_5 & \cdots & axis_b\end{bmatrix}$$

Reverse map: axes → facets

$$\begin{bmatrix} axis_1\\ axis_2\\ axis_3\\ \vdots\\ axis_m\end{bmatrix} \begin{matrix}\rightarrow\\ \rightarrow\\ \rightarrow\\ \rightarrow\\ \rightarrow\end{matrix} \begin{bmatrix} f_1 & f_3 & \cdots & f_c\\ f_1 & f_5 & \cdots & f_d\\ f_2 & f_6 & \cdots & f_e\\ & \vdots & \ddots & \vdots \\ f_7 & f_9 & \cdots & f_g\end{bmatrix}$$

(a)



(b)

Fig. 3-14 The formation of the "axes of rotation" map. (a) facets to axes map and its reverse map (For each axis, the reverse mapping collects every facet that maps to it in the original map), (b) an example of "axes of rotation" map where visibility is the normalized visible surface area showed by gradient color

3.5 Implementation

The method in this paper was implemented using the C++ programming language in Visual Studio 2010 on a PC of the following configuration: Intel Quad-Core i5-2300 CPU @2.8GHz, 6GB RAM running Windows 7/64bit on an SSD. The display of the models, GVM and the axis of rotation map were executed in OpenGL. A flowchart is used to illustrate the procedure of GVM computation (Fig. 3-15a). Together with the flowchart, Fig. 3-15b is used to illustrate some of the most critical steps in GVM computation.

As can be seen from the flowchart, GVM computation involves a loop structure. Using slice geometry, visibility computations among different slicing directions are completely independent of each other. This makes computing the GVM an embarrassingly parallel procedure. To take advantage of this, the code was written to support OpenMP, an API for shared-memory multiprocessing programming.

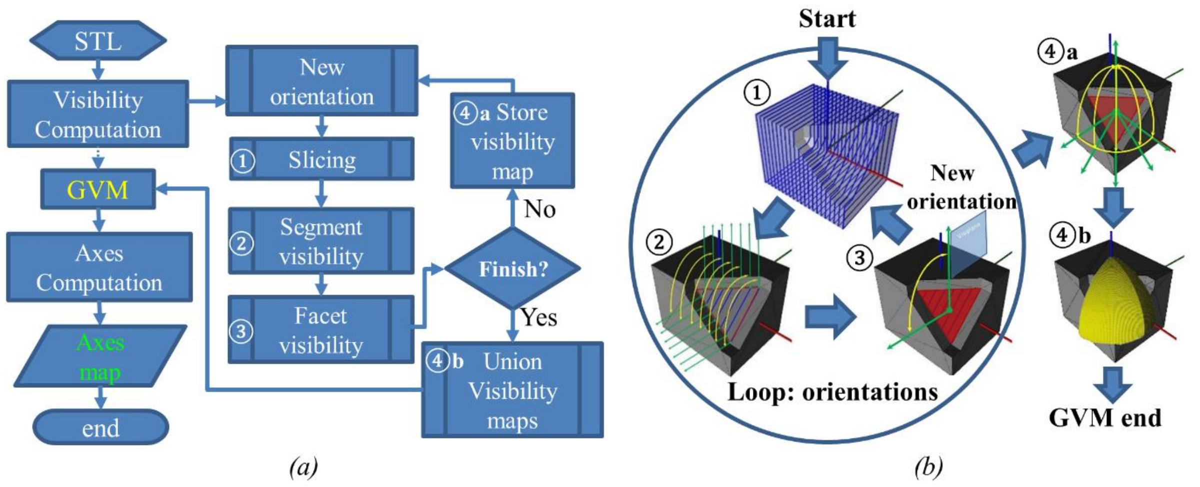


Fig. 3-15 An overview of the computational procedure (a) the flowchart showing entire computational flow, (b) the corresponding detailed illustration showing five critical steps in the flowchart

The program was tested with a set of models (Fig. 3-16). Model 1 is a cube with one cylindrical pocket; Model 2 is a cube with three connecting square pockets; Model 3 is a cube with one square pocket; Model 4 is a ring with the engraved text “RMPL”; Model 5 is a bracket with two through-holes; Model 6 is a toy jack. The facets of interest are denoted as yellow dots in model 1-4 as they are too small. The facets of interest are indicated with black arrows while the visibility cones are represented by a set of yellow points.

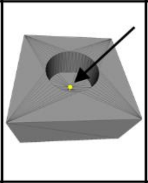
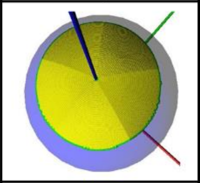
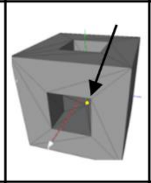
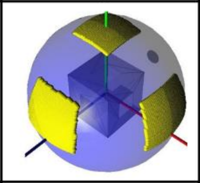
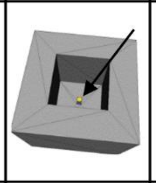
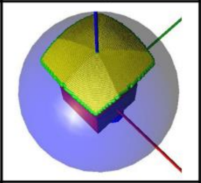
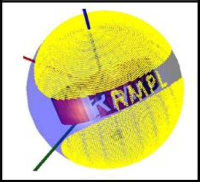
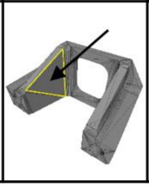
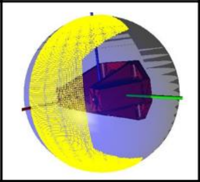
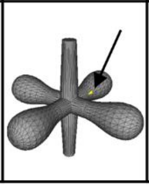
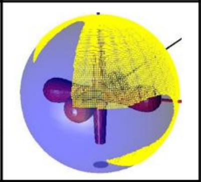
#	Model	Visibility Cones	#	Model	Visibility Cones	#	Model	Visibility Cones
1			2			3		
4			5			6		

Fig. 3-16 The resulting visibility cones for various models

The concave cylindrical pocket of model 1 is specially designed so that the radius is equal to the depth. Therefore, the visibility cone is expected to have a 90-degree opening angle. The program collects each point at the boundary of the visibility cone and gives a resulting opening angle of 90°~94°. Similarly, the square pocket of part 3 has the depth to width ratio of 0.8 whose visibility cone is expected to have an opening angle in the range 51°~60°. Program results show the opening angle is 54°~63°. The error between the expected value and program results can be attributed to the following reasons; 1) The facet investigated is not small enough to be treated as a point; 2) Since it is an STL mesh, the cylindrical pocket of model 1 does not have continuous curves; 3) the visibility computation has errors (which will be discussed later). The result for model 2 shows the method's ability to capture multiple visibility cones, while the results for model 4 shows the method's ability to capture small features (text lettering). Results for model 5 and 6 show the method's performance on representative industrial components.

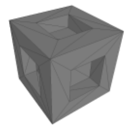
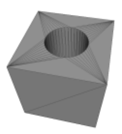
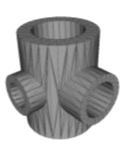
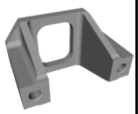
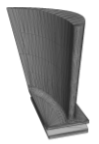
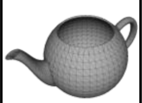
Part name	Sample1	Sample2	Sample3	Sample4	Sample5 [32]	Sample6 [33]	Sample7 [34]	Sample8 [35]
Part preview								
# of facets	60	176	1152	2088	3174	7922	10668	26402
Dimension (inch)	2.0x2.0x2.0	2.0x2.0x2.0	3.7x3.5x4	2.8x3.3x2.3	1.2x0.7x1.9	4.2x2.0x2.6	3.7x4.2x4.4	3.5x2.2x3.6
Dimension (cm)	5.1x5.1x5.1	5.1x5.1x5.1	9.4x8.9x10.2	7.1x8.4x5.7	3.1x1.8x5.0	10.7x5.1x6.6	9.4x10.7x11.1	8.8x5.6x9.1
Time (s)	**6**	**10**	**25**	**56**	**47**	**220**	**410**	**727**

Fig. 3-17 The time on computing Global Visibility Map for various models. Sample parts 5-8 are from the source in [32-35]

The results in Fig. 3-17 provides a comparison of computational time among models ranging from those with simple geometry to those with freeform surfaces. The slicing interval is set to be the model's diagonal length divided by 200. This setting ensures the computational time is not affected by the dimensions of the model.

The nature of the algorithm causes its computational time be largely affected by two factors, the number of slices k and number of facets n. To see their influences, an example model has been tested under various settings of k and n. The results are shown in Table 3-1 and plotted in Fig. 3-18. Note that the k slices are evenly spaced and the mesh is subdivided at the edge's middle point to increase size. This test was conducted on a PC of the following configuration: Intel Core i7-6700HQ CPU@2.6GHz, 16GB RAM, running windows 10/64bit on an SSD. The results show the computational time is approximately linear to both k and n.

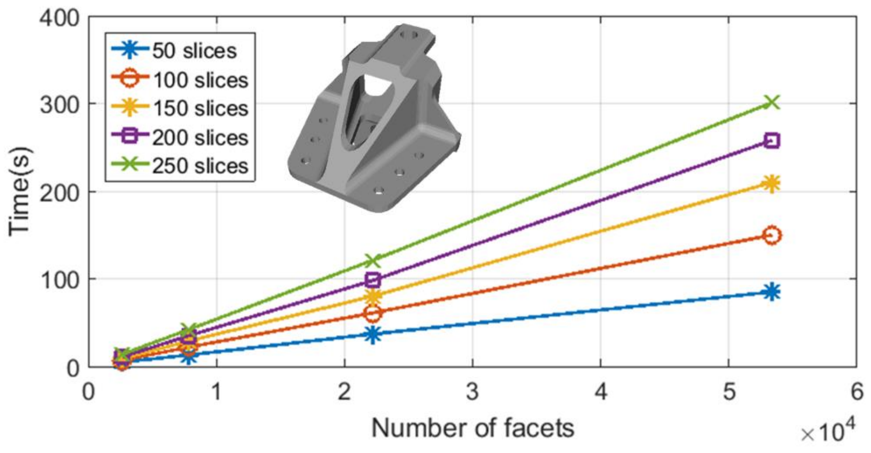


Fig. 3-18 The statistics of computational time on a various number of slices and facets for an example model. (a) the example model, (b) line chart where the number of facets is used as horizontal axis (Data from Table 3-1)

Table 3-1 The statistics of computational time(s) for the model in Fig. 3-18a under various settings of slice number k and facets number n

Slices# (k) / Facets # (n)	50	100	150	200	250
2690	5	7	9	11	14
7836	13	22	29	35	42
22194	37	61	80	98	121
53374	85	150	210	258	301

A time complexity analysis is given below, where we suppose a slice has c contours. These contours have $A = \{n_1, n_2, n_3 \ldots n_c\}$ vertices respectively. For a contour u, visibility is computed on each of its vertices. In the worst case, the visibility computation for a vertex requires traversing all vertices on its own contour (local visibility) and all vertices on its obstacle contours (global visibility). This gives the time complexity

$$O\left({n_u}^2 + \sum_{v=1\ldots c, v \neq u} n_u \cdot n_v\right).$$

Therefore, for all contours in this slice, the time complexity is:

$$O\left(\sum_{i=1\ldots c} {n_i}^2 + 2 \cdot \sum_{j=1\ldots c-1, k=j+1\ldots c} n_j \cdot n_k\right) = O\left(\sum_{i=1\ldots c} n_i\right)^2.$$

Denote $m_l = \sum_{i=1\ldots c} n_i$ which is the total number of vertices in slice l. Now, considering all slices, the computational complexity is:

$$O\left(\sum_{\forall l} {m_l}^2\right).$$

Suppose there are k slices and $\overline{m}$ is the root mean square (generalized mean with exponent 2) of the set $\{m_l\}$ (the set of vertices numbers of slices), then the above time complexity can be rewritten as:

$$O(k \cdot \overline{m}^2).$$

Now imagine an arbitrarily shaped polyhedron whose surface area is S and number of facets is n. The average area for a triangular facet is $\frac{S}{n}$ (assuming a homogeneous meshing of the model). Then, the characteristic edge length of an average facet is on the order of $\sqrt{\frac{S}{n}}$.

Meanwhile, the perimeter of an average slice is on the order of $\sqrt{S}$. Therefore, on average, the number of facets a slice may cross is $\frac{\text{average perimeter of a slice}}{\text{characteristic edge length of a facet}}$, which is on the order of $\left(\frac{\sqrt{S}}{\sqrt{\frac{S}{n}}}\right) = \sqrt{n}$. Consider the number of facets a slice crosses as roughly the number of vertices in a slice, we have $\bar{m} \approx t \cdot \sqrt{n}$, where t is some constant coefficient. Accordingly, the time complexity for visibility computation on all slices is:

$$O(k \cdot n).$$

Until now, the analysis only considers the time spent on one slicing direction. Suppose we conduct visibility computation in h different slicing directions, then the overall time complexity is:

$$O(k \cdot n \cdot h).$$

In this paper, we have a fixed number of slicing directions (h $=$ 180). This time complexity is in accordance with the test results (Fig. 3-18).

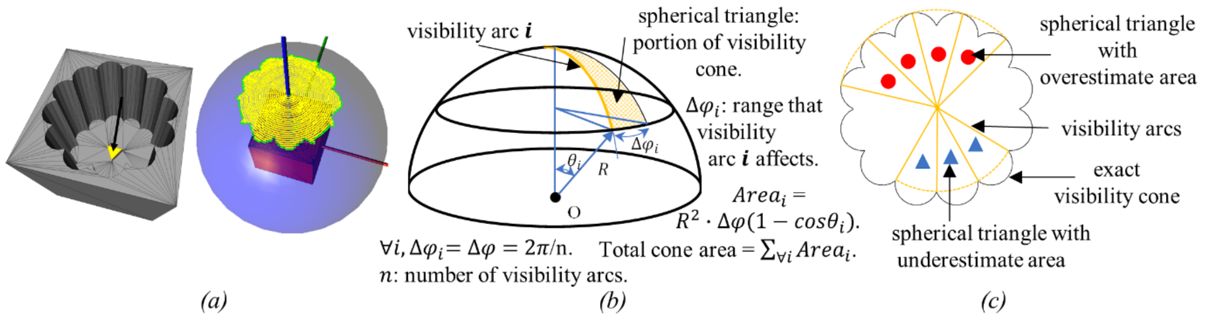
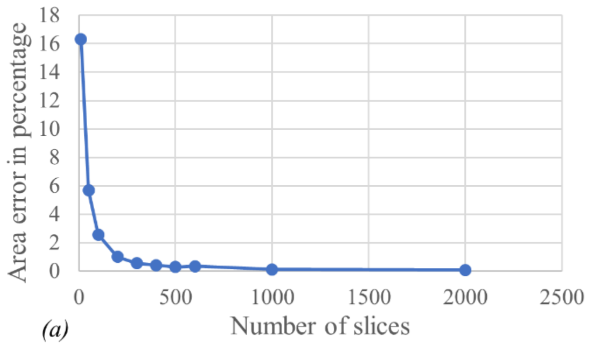

visibility arc $\boldsymbol{i}$
spherical triangle: portion of visibility cone.
$\Delta\varphi_i$: range that visibility arc $\boldsymbol{i}$ affects.
$\Delta\varphi_i$
θ_i
R
O
$Area_i = R^2 \cdot \Delta\varphi(1 - cos\theta_i)$.
$\forall i, \Delta\varphi_i = \Delta\varphi = 2\pi/\text{n}$.
Total cone area $= \sum_{\forall i} Area_i$.
n: number of visibility arcs.
spherical triangle with overestimate area
visibility arcs
exact visibility cone
spherical triangle with underestimate area
(a)
(b)
(c)

Fig. 3-19 The method to obtain total area of the visibility cone from discrete visibility arcs; (a) example part and visibility cone, (b) total area of visibility cone is computed as the area summation of spherical triangles determined by corresponding visibility arcs, (c) example showing the spherical triangle determined by the visibility arc could either overestimate or underestimate the portion of the cone area

The accuracy of this method is affected by two major factors: the number of slices and the number of slicing directions. The number of facets does not affect accuracy per se, although number of facets affects the geometric approximation of the original CAD model. Also, the visibility will be closer to point visibility for smaller facets. Since a facet is approximated by its containing segments in visibility computation, putting more evenly spaced segments in a facet gives a more accurate representation. Increasing the number of slices effectively increases the number of segments generated for each facet. Hence, a better visibility result is expected. The other major approximation in this method is the use of a finite number of slicing directions. This causes that only a subset of the visibility cone is covered by the visibility arcs. Hence, increasing the number of slicing directions (thus number of visibility arcs) is expected to improve visibility results. To facilitate the accuracy estimation, we use the area of the visibility cone as the indicator. In this test, the area of a visibility cone is computed as the area summation of a set of spherical triangles where the spherical triangles are determined by the corresponding visibility arcs (Fig. 3-19). To see how the cone area is affected by the number of slices k and number of slicing directions m, the sample from Fig. 3-19a is tested with various settings of k and m (Fig. 3-20). Please note, the analytical solution of this cone area is approximated by a nearly converged result which all other results are compared to. The results show that with increasing number of slices, the cone area decreases monotonically and trends to converge. The cone area decreases because the more segments a facet has, the more restricted the visibility (consider the intersection operation). When the number of segments in a facet is considerably large, the visibility for a facet will hardly change since the segments populate much of the area of the facet. Therefore, the visibility converges with increasing slice number. On the other hand, the results also show that with increasing number of slicing directions, the cone area becomes more accurate. The number of slicing directions determines how many visibility arcs are used to approximate the cone. The more visibility arcs, the better the geometry of the cone is captured. Hence, the visibility improves with increasing slicing directions. Depending on the positions of the visibility arc, it could either overestimate or underestimate a certain portion of cone area (Fig. 3-19c). Thus, both positive and negative errors exist in the results. However, in general, the absolute errors tend to decrease as the number of slicing directions increases, which is expected for arbitrary shaped cones.



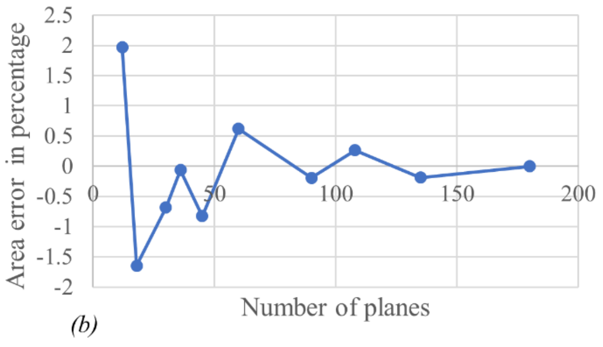


Fig. 3-20 The results showing areas of visibility cone for (a) errors of areas of visibility cone versus the number of slices, (b) errors of areas of visibility cone versus the number of slicing directions (Data from Table 3-2)

Table 3-2 The results of visibility cone area (assuming $R = 1$) for the model in Fig. 3-19a under various settings. All area results are compared to the nearly converged result under the setting (180 slicing directions & 10,000 slices)

Number of slice directions = 180											
# of slices	10	50	100	200	300	400	500	600	1000	2000	**10000**
cone area	0.7493	0.6807	0.6606	0.6506	0.6476	0.6468	0.646	0.6463	0.6449	0.6446	**0.6441**
error (%)	16.3	5.7	2.6	1.0	0.54	0.42	0.29	0.34	0.12	0.078	**0**
Number of slices = 10,000											
# of planes	12	18	30	36	45	60	90	108	135	**180**	
cone area	0.6568	0.6335	0.6397	0.6497	0.6387	0.6481	0.6429	0.6458	0.6429	**0.6441**	
error (%)	2.0	-1.6	-0.68	-0.06	-0.82	0.62	-0.19	0.26	-0.18	**0**	

Once the GVM is available, an axis of rotation map can be generated. In Fig. 3-21, three different cubes with one, two and three orthogonal pockets plus a cube with a cylindrical through-hole were tested. Note in Fig. 3-21a-c and e, only axes that make the model 100% visible are shown in red; whereas, in Fig. 3-21d, axes with incomplete visibility are also shown. In addition, the distance from the 3D dots to the origin is scaled by visibility (thus, an axis' visibility is reflected by a dot's color as well as a dot's distance to the origin). The results show that cubes with one pocket and two pockets find feasible axes of rotation. They can be machined by a 4-axis CNC milling machine. However, cubes with three orthogonal pockets are at most 96% visible for a 4-axis CNC machine. Thus, it cannot be machined fully. Such results actually reflect the general case where features like pockets inflict strong restrictions on machinability. Compared to a cube with one square pocket, a cube with a cylindrical through-hole finds more feasible axes (Fig. 3-21e). This is because the through-hole can possibly be reached from two opposite directions, making it more accessible. The cube with a through-hole is designed in a way that the diameter of the cylinder is half of the cube's edge length. The expected opening angle of this axes region can be derived from simple geometry (26.57°). The result from the program is 26°.

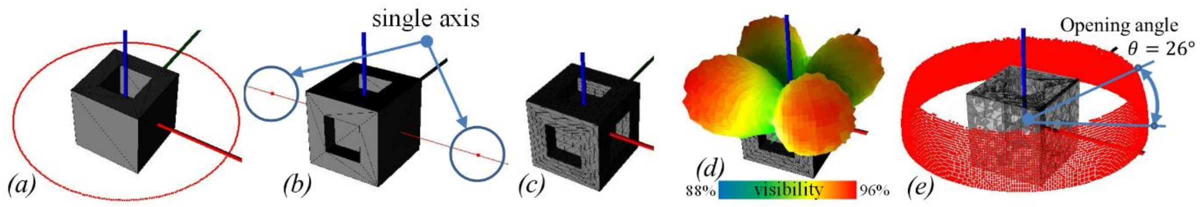


Fig. 3-21 Examples of the computed feasible axes of rotation and the "axis of rotation" map. (a) a cube with one pocket has feasible axes on a great circle, (b) a cube with two orthogonal pockets has only one feasible axis, (c) a cube with three orthogonal pockets has no feasible axis, (d) the axis of rotation map for a cube with three orthogonal pockets, (e) a cube with a cylindrical through-hole has a feasible axes region of a ring shape

Additionally, the axis of rotation region that reveals more than 99% of surface area is shown for two industrial parts in Fig. 3-22.As can be seen in both parts, in additional to the principle axes, which we normally consider, many more unconventional axes are also found to be feasible. More choices of axes imply a more manufacturable part. On the other hand, as some previous un-machinable parts may become machinable by choosing an unconventional axis now, more choices of axes also give the part designer more freedom in choosing geometries.

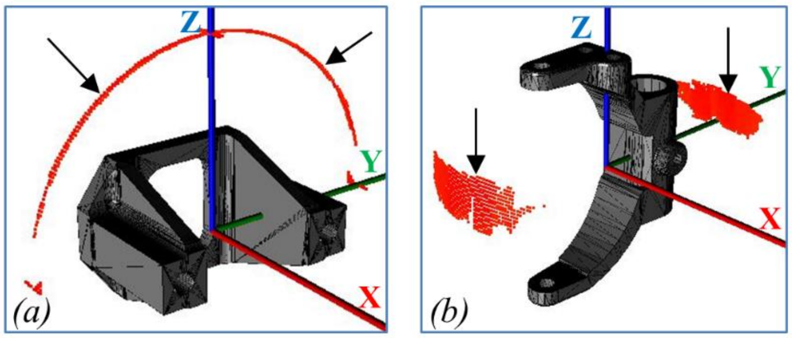


Fig. 3-22 The axes of rotation region which reveals more than 99% of component's surface area. Axes are shown in red dots. (a) an industrial bracket with two cylindrical through-holes, (b) an industrial linkage

As can be seen in Fig. 3-22a, the part's Y-axis is a feasible axis of rotation for the industrial bracket. This component was further analyzed for toolpath planning via an automated machining software called CNC-RP, developed in the Rapid Manufacturing and Prototyping Lab at Iowa State University, by selecting the part's Y-axis as the axis of rotation. The component was machined via a 4-axis HAAS VF2ss CNC mill from cylindrical aluminum stock (Fig. 3-23a). After the supports were removed, the finished part is shown in Fig. 3-23b, clearly showing that the computed axis of rotation is feasible for the machining of this bracket.

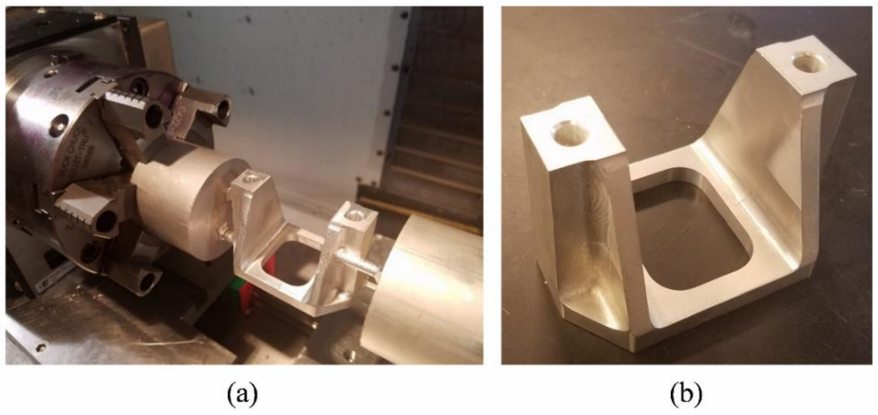


Fig. 3-23 An example of an industrial bracket machined by a 4-axis CNC machine where the axis of rotation is chosen as the part's Y-axis. (a) the bracket with supports on after CNC-RP machining, (b) The finished bracket with all supports removed

3.6 Conclusion

This paper introduces a new method to compute the Global Visibility Map (GVM). The method starts with polyhedron models and creates visibility cones by assembling visibility arcs computed from a set of planes. Parallel computing is implemented to take advantage of the embarrassingly parallel nature of the slice geometry and saves a considerable amount of computational time. Results have shown that the computed visibility cone has good accuracy and captures small features. Given the GVM, an axis of rotation map is developed to determine the feasible axes of rotation for parts to be machined by 4-axis milling. Results have shown that the computed feasible axes are valid and many more solutions in additional to the conventional ones like the principle axes are discovered. In the case that no feasible axis exists, the axis of rotation map still provides valuable information for part designers by showing the best axes for visibility. However, there are still challenges in the proposed method. One is that in actual setup planning for machining, satisfying visibility requirement is only a necessary condition, while not sufficient to obtain true accessibility since the tool's geometry should also be considered. Previous work addresses this problem for 3-axis flat end milling [36], which could be extended to 5-axis ball end milling. Also, the proposed approach is an approximated method, where the computational effort increases dramatically when the visibility resolution and accuracy requirements raise. Other than increasing the number of slices for better accuracy, one could also apply an adaptive slicing strategy [37] for a better approximation of the original model, since reducing the staircase error would also benefit visibility accuracy. Moreover, due to the facet visibility's definition, the size of a facet affects the extent its visibility is explored. In the future, effort should be made to find a tool's true accessibility on the foundation of the current visibility results. Besides, a more efficient way of building up the visibility cone from arcs other than the exhaustive method should be explored. Furthermore, we are actively looking for opportunities to extend the current method to facilitate the setup planning of advanced 5-axis CNC machines where a more comprehensive set of machining factors including the tools' orientations, lengths and diameters can be optimized.

3.7 References

[1] Kweon, S., and Medeiros, D. J., 1998, "Part orientations for CMM inspection using dimensioned visibility maps," Comput Aided Design, 30(9), pp. 741-749.

[2] Wang, N., and Tang, K., 2007, "Automatic generation of gouge-free and angular-velocity-compliant five-axis toolpath," Comput Aided Design, 39(10), pp. 841-852.

[3] Balasubramaniam, M., Sarma, S. E., and Marciniak, K., 2003, "Collision-free finishing toolpaths from visibility data," Comput Aided Design, 35(4), pp. 359-374.

[4] Suh, S.-H., and Lee, J.-J., 1998, "Five-Axis Part Machining With Three-Axis CNC Machine and Indexing Table," Journal of Manufacturing Science and Engineering, 120(1), pp. 120-128.

[5] Morishige, K., Takeuchi, Y., and Kase, K., 1999, "Tool Path Generation Using C-Space for 5-Axis Control Machining," Journal of Manufacturing Science and Engineering, 121(1), pp. 144-149.

[6] Elber, G., and Zussman, E., 1998, "Cone visibility decomposition of freeform surface," Comput Aided Design, 30(4), pp. 315-320.

[7] Fu, M. W., 2008, "The application of surface demoldability and moldability to side-core design in die and mold CAD," Comput Aided Design, 40(5), pp. 567-575.

[8] Chen, L.-L., Chou, S.-Y., and Woo, T. C., 1995, "Partial visibility for selecting a parting direction in mold and die design," J Manuf Syst, 14(5), pp. 319-330.

[9] Priyadarshi, A. K., and Gupta, S. K., 2004, "Geometric algorithms for automated design of multi-piece permanent molds," Comput Aided Design, 36(3), pp. 241-260.

[10] Bittner, J., and Wonka, P., 2003, "Visibility in computer graphics," Environ Plann B, 30(5), pp. 729-755.

[11] Zach, C., and Karner, K., 2003, "Progressive compression of visibility data for view-dependent multiresolution meshes," Wscg'2003, Vol 11, No 3, Conference Proceedings, pp. 546-553.

[12] Lu, Y., Ding, Y., and Zhu, L., 2016, "Smooth Tool Path Optimization for Flank Milling Based on the Gradient-Based Differential Evolution Method," Journal of Manufacturing Science and Engineering, 138(8), pp. 081009-081009-081011.

[13] Xu, K., and Tang, K., 2016, "Optimal Workpiece Setup for Time-Efficient and Energy-Saving Five-Axis Machining of Freeform Surfaces," Journal of Manufacturing Science and Engineering, 139(5), pp. 051003-051003-051016.

[14] Song, X., Pan, Y., and Chen, Y., 2015, "Development of a Low-Cost Parallel Kinematic Machine for Multidirectional Additive Manufacturing," Journal of Manufacturing Science and Engineering, 137(2), pp. 021005-021005-021013.

[15] Chen, L. L., and Woo, T. C., 1992, "Computational Geometry on the Sphere with Application to Automated Machining," J Mech Design, 114(2), pp. 288-295.

[16] Tang, K., Woo, T., and Gan, J., 1992, "Maximum Intersection of Spherical Polygons and Workpiece Orientation for 4-Axis and 5-Axis Machining," J Mech Design, 114(3), pp. 477-485.

[17] Chen, L. L., Chou, S. Y., and Woo, T. C., 1993, "Separating and Intersecting Spherical Polygons - Computing Machinability on 3-Axis, 4-Axis and 5-Axis Numerically Controlled Machines," Acm T Graphic, 12(4), pp. 305-326.

[18] Chen, L.-L., Chou, S.-Y., and Woo, T. C., 1993, "Parting directions for mould and die design," Comput Aided Design, 25(12), pp. 762-768.

[19] Suh, S. H., and Kang, J. K., 1995, "Process Planning for Multiaxis Nc Machining of Free Surfaces," Int J Prod Res, 33(10), pp. 2723-2738.

[20] Li, Y., and Frank, M. C., 2007, "Computing non-visibility of convex polygonal facets on the surface of a polyhedral CAD model," Comput Aided Design, 39(9), pp. 732-744.

[21] Tarbox, G. H., and Gottschlich, S. N., 1995, "Planning for Complete Sensor Coverage in Inspection," Comput Vis Image Und, 61(1), pp. 84-111.

[22] Spitz, S. N., and Requicha, A. A. G., 2000, "Accessibility analysis using computer graphics hardware," Ieee T Vis Comput Gr, 6(3), pp. 208-219.

[23] Frank, M. C., Wysk, R. A., and Joshi, S. B., 2006, "Determining setup orientations from the visibility of slice geometry for rapid computer numerically controlled machining," J Manuf Sci E-T Asme, 128(1), pp. 228-238.

[24] James Stewart, A., 1999, "Computing visibility from folded surfaces," Computers & Graphics, 23(5), pp. 693-702.

[25] Hou, Z., Li, X., Huang, Y., and Ho, S. T., 2013, "Physics of elliptical reflectors at large reflection and divergence angles I: Their design for nano-photonic integrated circuits and application to low-loss low-crosstalk waveguide crossing," Optics Communications, 287, pp. 96-105.

[26] Suthunyatanakit, K., Bohez, E. L. J., and Annanon, K., 2009, "A new global accessibility algorithm for a polyhedral model with convex polygonal facets," Comput Aided Design, 41(12), pp. 1020-1033.

[27] Dhaliwal, S., Gupta, S. K., Huang, J., and Priyadarshi, A., 2003, "Algorithms for Computing Global Accessibility Cones," Journal of Computing and Information Science in Engineering, 3(3), pp. 200-209.

[28] Liu, M., and Ramani, K., 2007, "Computing an exact spherical visibility map for meshed polyhedra," Proceedings of the 2007 ACM symposium on Solid and physical modeling, ACM, Beijing, China, pp. 367-372.

[29] Liu, M., Liu, Y. S., and Ramani, K., 2009, "Computing global visibility maps for regions on the boundaries of polyhedra using Minkowski sums," Comput Aided Design, 41(9), pp. 668-680.

[30] Joshi, A. M., 2015, "Computer aided process planning for multi-axis CNC machining using feature free polygonal CAD models," Graduate Theses and Dissertations., Iowa State University, Digital Repository@ISU.

[31] Li, Y., and Frank, M. C., 2012, "Computing Axes of Rotation for Setup Planning Using Visibility of Polyhedral Computer-Aided Design Models," J Manuf Sci E-T Asme, 134(4).

[32] Rajab, S. M., 2016, "Reverse engineer blade," grabcad.com/library/reverse-engineer-blade-1.

[33] Čapo, B., 2016, "Tea Pot," grabcad.com/library/tea-pot-11.

[34] Boose, C., 2015, "Smooth Fluted Pitcher," grabcad.com/library/smooth-fluted-pitcher-1.

[35] Pundir, N., 2016, "Fortune cookie," grabcad.com/library/fortune-cookie-1.

[36] Li, Y., and Frank, M. C., 2006, "Machinability analysis for 3-axis flat end milling," J Manuf Sci E-T Asme, 128(2), pp. 454-464.

[37] Siraskar, N., Paul, R., and Anand, S., 2015, "Adaptive Slicing in Additive Manufacturing Process Using a Modified Boundary Octree Data Structure," Journal of Manufacturing Science and Engineering, 137(1), pp. 011007-011007-011011.

CHAPTER 4. COMPUTING THE ACCESSIBILITY OF A POLYHEDRON USING A THREE-DIMENSIONAL OFFSET FOR BALL-END MILLING

Abstract

Accessibility of a surface is the set of directions that a cutting tool follows to reach the entirety of a surface from a distance without collision. Compared to visibility, where a surface is visible from a direction if a set of parallel rays in that direction can reach the entirety of it without collision, accessibility accounts for the body of the cutting tool (tool end and tool shank) in collision. Thus, accessibility is a more accurate modelling of the machining constraints. As visibility is relatively easier to compute, we propose a method that generates accessibility results from visibility on the offset surface for ball-end tools. Putting cutter locations on the offset surface avoids a tool's local gouging. Visibility results on the offset surface provides a tool's feasible orientations that are globally collision-free. A one-to-many facets mapping from part surface to offset surface is created, such that a part's surface can find its cutter locations on the offset surface. The results have shown that the tool body is effectively accounted for in collision. The method provides more than a Boolean answer of accessibility – it provides all feasible tool orientations. In this paper, we are concerned with triangular polyhedral models only, where the smallest surface is a triangle.

4.1 Introduction

Accessibility of a surface is the set of directions that a cutting tool follows to reach the entirety of this surface from a distance without collision. Accessibility recognizes a tool's body (a tool end and a tool shank) in collision. Visibility, on the other hand, simplifies the cutting tool with a line (infinitely thin tool).

Visibility for a polyhedron

Visibility for a part consists of visibility for each surface of the part. As we represent a part's surface using a triangulated (tessellated) polyhedron, each surface is a triangular facet. Thus, computing the visibility for a part is equivalent to computing the visibility for each facet of the part.

The visibility of a polyhedral facet is a set of directions. Along each direction, a set of parallel lines can reach every point in this facet without intersecting the polyhedron. A 2D example is given in Fig. 4-1. Notice, the 2D counterpart of a facet is a segment.

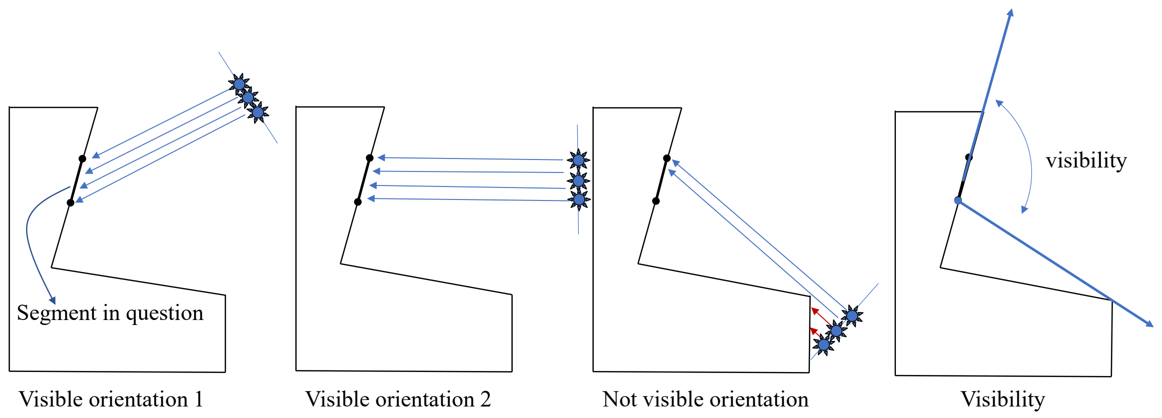


Fig. 4-1 The visibility of a segment on a polygon

Accessibility for a polyhedron

Tool accessibility of a polyhedral facet is a set of directions. Along each direction, a set of tools whose central lines are parallel can reach every point of this facet without colliding with the polyhedron. Both visibility and accessibility are widely used for setup and toolpath planning in Computer Numerical Controlled (CNC) machining. However, visibility has its limitation in that: a surface that is visible from a direction is not necessarily accessible by the tool (Fig. 4-2 leftmost subfigure). This is because visibility only provides accessibility for an infinitely thin tool; space in the tool other than the center line can easily collide with the part. This makes visibility an overestimate of actual tool accessibility. Accordingly, using visibility as accessibility is either unsafe or will render incomplete cutting of the part. To solve this problem, the tool body must be accounted for in accessibility computation.

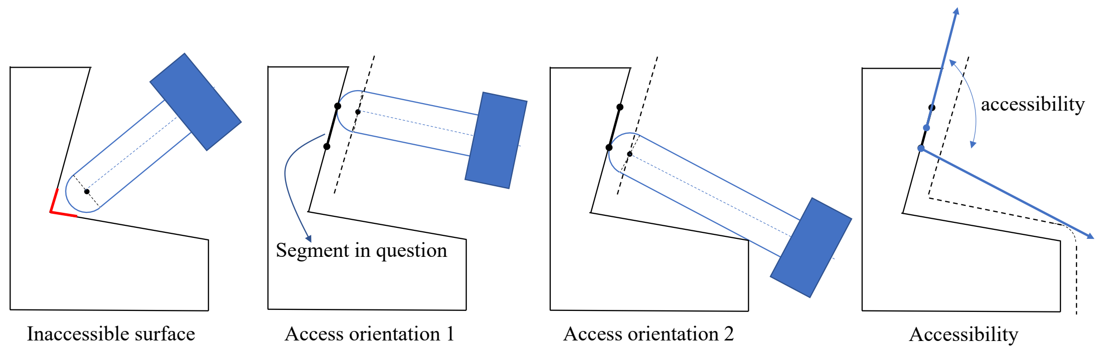
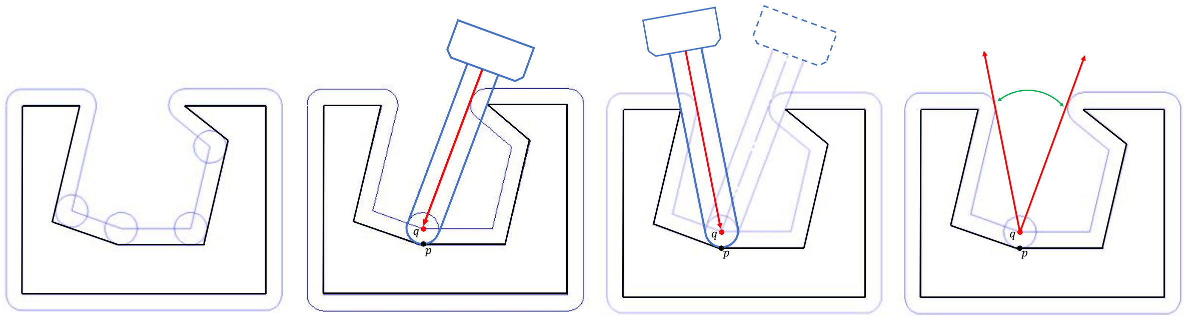


Fig. 4-2 The accessibility of a segment on a polygon

Using an Offset Surface for Ball-end Tool for Accessibility

In this paper, we propose that a tool body can be accounted for in collision by 1) using the three-dimensional (3D) offset surface of the part as the cutter locations and 2) treating the offset surface as obstacles in feasible tool orientations computation (Fig. 4-3). Cutter locations are where the center of the tool's ball-end locates. Using an offset surface as cutter locations avoids local gouging of the tool. Treating the offset surface as an obstacle in feasible tool orientations computation avoids global collision between the tool and part. Lastly, the method calculates accessibility on the part surface as visibility on the offset surface.



(a) Offset contour by tool radius (b) The rightmost feasible orientation for ball end mill on point p (c) The leftmost feasible orientation for ball end mill on point p (d) The accessibility of ball end mill on point p

Fig. 4-3 The observation that accessibility of a part surface can be derived from the visibility of its offset surface

4.2 Related Work

Miller investigated the application of surface accessibility on the visual effect of shading. In that work, accessibility was defined as radius of a sphere which may touch a surface point and not intersect any surface [1]. The results highlighted the inaccessible surface by a spherical probe. However, the method only provides a Boolean answer to whether a surface is accessible by the probe; not actual access orientations. Elber proposed a method to determine the inaccessible surface induced by a flat-end tool in 5-axis machining. However, the method is restricted to inaccessibility of convex surfaces due to other check surfaces. Also, it does not provide feasible tool orientations for the accessible surfaces [2]. Tang et al. proposed a surface offset/upper envelop method that solves gouging for 3-axis multi-surface NC machining [3]. Kim et al. proposed a triangular mesh offset algorithm for tool path planning of a generalized cutter in NC machining [4]. However, both Tang and Kim’s work focuses on 3-axis NC machining where tool orientation was not considered. Xu et al. proposed a method that determines feasible tool orientations for each predefined cutter contact (CC) point in 5-axis NC machining [5]. However, CC points and CC paths must be provided first. This is not desirable if we want to optimize the tool path choice. This method works better in the case where CC paths are already determined and used to generate gouge-free and collision-free tool orientations.

In conclusion, a method that can determine accessible and inaccessible part surfaces for ball-end cutter in 5-axis machining environment while also providing feasible tool orientations for each surface is needed. In addition, we propose a new method that can evaluate accessibility by reusing visibility results.

4.3 Methodology

4.3.1 Overview of the Methodology

The method consists of four steps (Fig. 4-4):

- The part surface is offset outward by the ball-end tool radius. The 3D offset is realized by conducting a 3D Minkowski Sum between two polyhedra: one sphere representing the tool-end and one representing the part;
- Visibility is computed on the offset surface using an approximate visibility method;
- A mapping between part surfaces and offset surfaces is generated (facets to facets mapping);
- And finally, accessibility of a part facet is obtained by intersecting visibilities of its mapped facets on the offset surface.

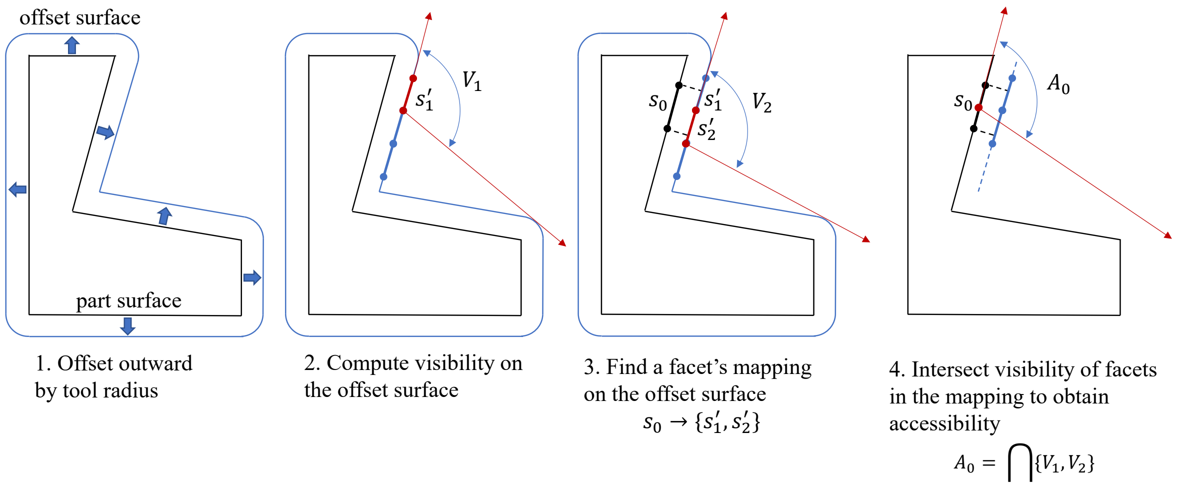

offset surface
part surface
1. Offset outward by tool radius
V_1
s_1'
2. Compute visibility on the offset surface
s_0
s_1'
s_2'
V_2
3. Find a facet's mapping on the offset surface
$s_0 \rightarrow \{s_1', s_2'\}$
s_0
A_0
4. Intersect visibility of facets in the mapping to obtain accessibility
$A_0 = \bigcap\{V_1, V_2\}$

Fig. 4-4 The four steps to compute accessibility

4.3.2 Computing the 3D Offset Surface

We first explain the rationale behind "Surface Offsetting for Accessibility" – why the use of an offset surface leads to the gouging-free and collision-free directions of ball-end tools.

Ball Centered on Offset Surface to Avoid Local Gouging

For the rest of the analysis, it is assumed we use a ball-end tool of a fixed radius. The offset surface has divided the 3D space into two subspaces for cutter locations; an inner subspace and outer subspace (subspace does not include the boundary, i.e. offset surface). The inner subspace is the collision space where a ball-end must collide with the part if it is in this subspace. The outer subspace is the collision-free space that guarantees no ball-end to part intersection. If the ball-end's center is positioned on the offset surface, it might be tangent to some part surfaces, but it will never gouge into any. Since a part surface can only be machined if the tool end is tangent to it, the offset surface becomes the only feasible space for cutter locations (locations for the ball-end's center).

A surface of the part can be machined if and only if every point in this surface can find its corresponding cutter location on the offset surface. Thus, the accessibility of a part surface can be solved on its offset surface.

Using Offset Surface as the Obstacle to Avoid Global Collision

Although placing a tool’s ball center on the offset surface avoids local gouging, this location being the travel destination of the ball-end might not be reachable due to tool-part collision. Even if it is reachable, the tool’s access directions are still to be determined.

More formally, the problem to solve is this: Given a point on the offset surface, from what directions can a tool’s ball center approach this point without any tool-part collision (Assuming the tool always approaches the part surface in a straight line and the tool is infinitely long)? Considering the tool’s approaching as a dynamic process, at any time, the travelling ball should be collision-free with the part. Notice the offset surface has created an outer subspace that guarantees no ball-part intersection and an inner subspace that guarantees ball-part intersection. Thus, the trace of the travelling ball (a ray) must lie within the outer subspace. This is equivalent to shooting a ray from said point without colliding with the inner subspace. In fact, this is effectively a visibility problem; finding the set of rays which start from a fixed point on the offset surface that do not interest with the offset surface (Notice the offset surface is the boundary of the inner subspace).

In conclusion, the visibility results of a point on the offset surface (Cutter Location (CL) point) provide directions that render the ball-end tool globally collision free (i.e. tool shank does not collide with the part). Meanwhile, the visibility results also provide the feasible tool orientations of a part point corresponding to this CL point.

Using the 3D Minkowski Sum for Surface Offset

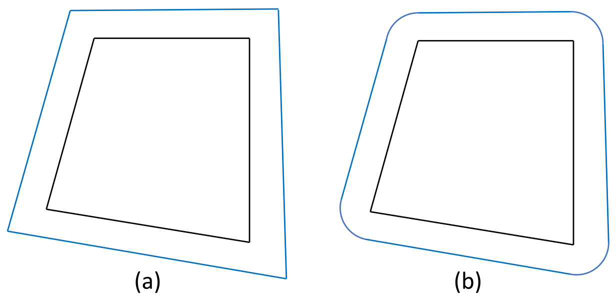


Fig. 4-5 Comparison between (a) Simple offset and (b) a Minkowski Sum with disk

To simulate ball-end tool machining (assuming the ball-end's radius is R), we require the offset surface to have the following property: First, any sphere of radius R centered on the offset surface does not collide with the part except for tangentially touching; Second, the offset surface must contain the exact sphere-part collision space, no larger or smaller. The first one guarantees there is no part gouge if cutter locations are on the offset surface. The second one guarantees an accurate boundary of the sphere-part collision space which affects the accuracy of visibility computation where the offset surface is treated as an obstacle.

The commonly used offset strategy in most CAM software (offset face-by-face, then extend and trim) does not satisfy the second requirement (Fig. 4-5a). However, the surface generated by a 3D Minkowski Sum of the part and a sphere of radius R does meet the requirement (Fig. 4-5b). Minkowski sum of two geometry objects is the space occupied by both objects when traversing one object along every point in the other object (Fig. 4-6). The 3D Minkowski Sum of a part and a sphere simulates the process of traversing a ball-end in the interior and on the surface of the part. The sum is the union of the spaces where tool-part collision occurs. Therefore, the interior of the sum is the part-tool collision space where the exterior of the sum is the part-tool collision-free space.

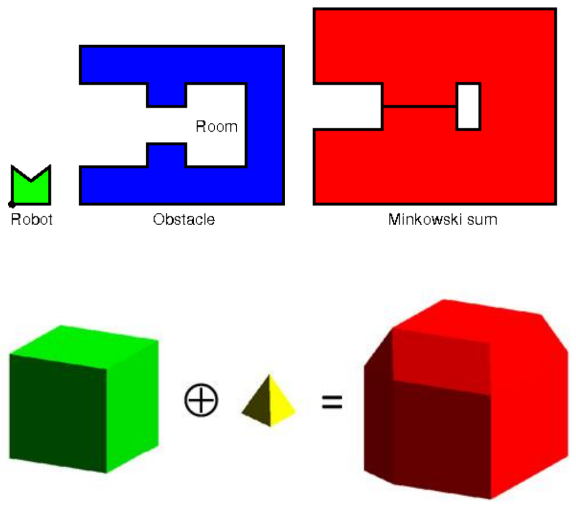


Fig. 4-6 2D and 3D Minkowski Sum [6, 7]

There are many computer programs that can compute the 3D Minkowski Sum, a commonly used one is by CGAL [6]. However, the time complexity of its implementation is $O(m^3n^3)$ making it a very slow process, where m and n are the complexities of the two input polyhedra. Complexity of a polyhedron is the sum of its vertices and edges. Lien has proposed a much faster algorithm to compute 3D Minkowski Sum using collision detection, though some results might be problematic (low dimensions) [8]. Li has proposed a fast voxel-based 3D Minkowski Sum using GPU [7]; however, the results are not exact. In this work, we choose OpenSCAD, an open source software based on Constructive Solid Geometry (CSG) and a CGAL kernel that provides basic Boolean operations of solids and 3D Minkowski Sum [9]. Though it is somewhat computationally expensive, we prefer the accuracy of the offset surface in this work.

Inaccessible Part Surfaces Determination

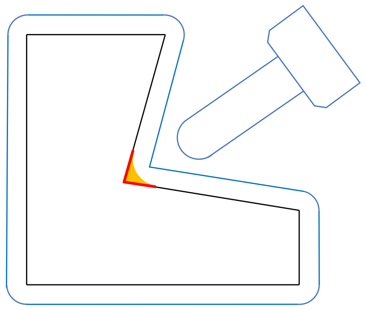


Fig. 4-7 Inaccessible surface (red) and residual material (yellow)

We first find the part surfaces that are completely inaccessible by the ball-end cutter to save computation time for the facet mapping from part surface to offset surface. Inaccessible surfaces can be determined using a two-offsetting technique (Fig. 4-8):

1. Offset the part surface outward by distance R (using 3D Minkowski Sum); the offset surface defines Cutter Locations (CL) on the surface (Fig. 4-8b).
2. Offset the CL surface inward by distance R (using 3D Minkowski Sum); the resulting surface defines the Maximal Cut (MC) surface (Fig. 4-8c).
3. Subtract the solid represented by the MC surface by the part to get residual volume; which represents the material left by Maximal Cut (Fig. 4-8c).
4. Calculate the part surfaces that are tangent to the residual material; these part surfaces are not accessible (Fig. 4-8d).

Note that in OpenSCAD, there is no Minkowski Sum that goes inward. To achieve the same effect of inward offsetting, we can subtract a model from a bigger cube (big enough to cover the model). The resulting solid has an inner surface that is the exterior surface of the model. Since it is an exterior surface, we can then conduct Minkowski Sum on it which effectively realizes inward offsetting of the model.

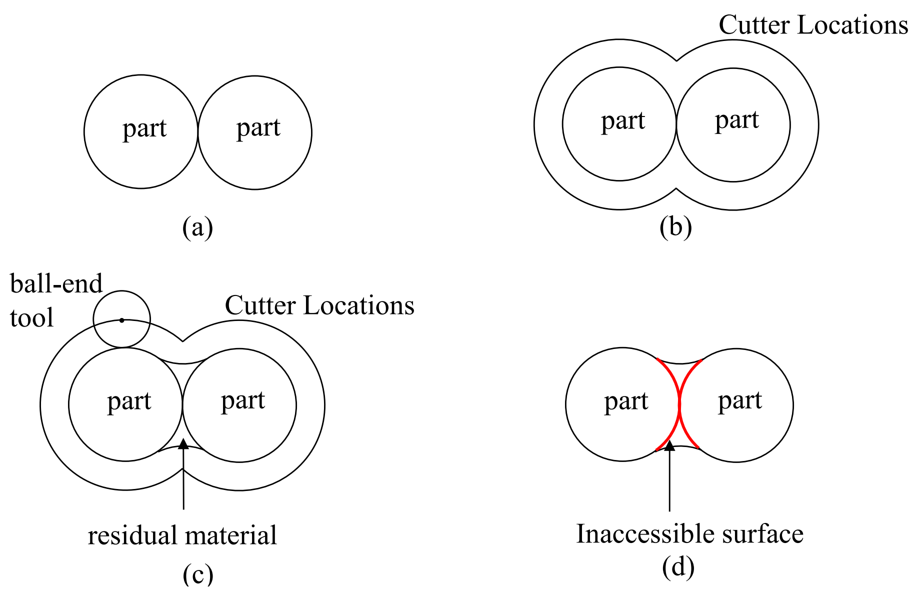


Fig. 4-8 The two-offsetting method to find inaccessible surfaces

4.3.3 Mapping from Part Surface to Offset surface

Cutter Locations Determination for a Part Surface

To calculate a part facet's feasible tool orientations, its cutter locations must be found. Cutter location for a point on the part surface is its normal projection on the offset surface. We define such projection as an image of the part vertex/surface. Because the offset surface has a different triangulation from that of the part surface, the image of a part facet (on the offset surface) could intersect multiple offset surface facets. We define the offset facets that intersect with the part facet's image as the *mapped facets*. In this way, the accessibility of a part facet is the intersection of the visibilities of its *mapped facets*.

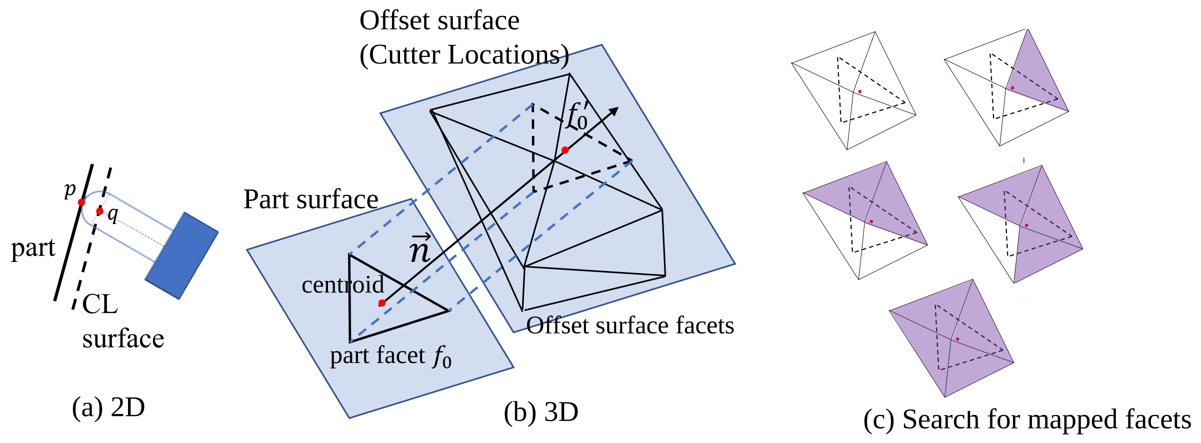


Fig. 4-9 Locating cutter locations; (a) Locating the cutter location of a point on part surface (b) Locating the cutter locations for a part facet (c) Searching all mapped facets by a graph traversal (Breadth First Search is chosen).

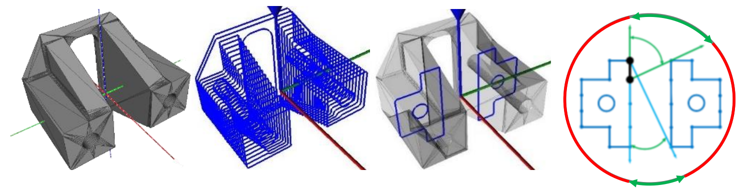
To find the mapped facets for a part facet f_0, a segment starting from the centroid of f_0, extended in the normal direction of f_0, of a length R (R is ball-end tool's radius) is created. The facet on the offset surface intersected by this segment must be one of the mapped facets (denoted as f_0'). Because the mapped facets must be connected (thus in a connected component), we can conduct a graph traversal starting from f_0' to enumerate all mapped facets. In each step of the traversal, we test if the current offset facet intersects with the image facet, if yes, we proceed to explore its neighbor facets, else we stop exploring this facet.

To speed up the collision detection which finds f_0', we create an Axis Aligned Bounding Box (AABB) tree for quick intersection query on the offset surface. This AABB tree is reused for locating cutter locations of other part facets.

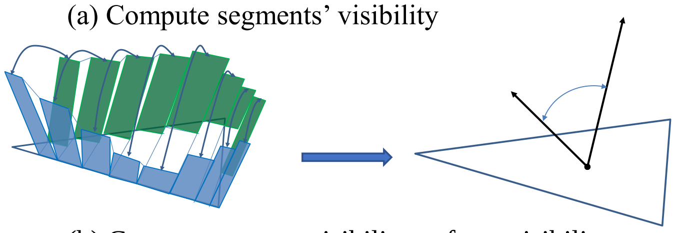
4.3.4 Computing the Visibility for Facets on the Offset Surface

In this paper, we use a slice geometry-based method from our previous work to compute visibility (Fig. 4-10) [10]. A brief review of the four main steps is as follows:

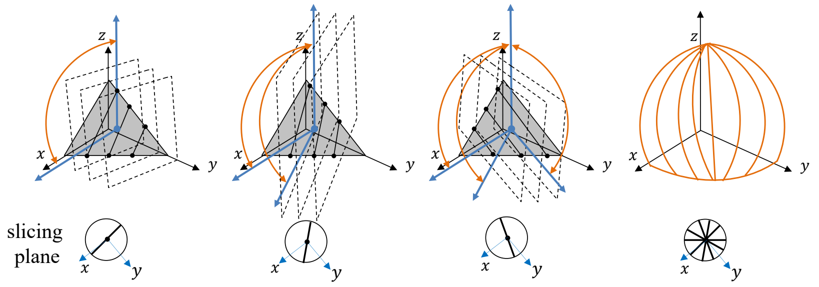
First, we select a set of parallel planes cutting the part into a set of slices. Each slice consists of a set of segments. A 2D visibility algorithm is conducted on each slice to compute visibility for each segment. The direction perpendicular to these parallel planes is called a slicing direction. Second, for each facet sliced by a set of segments, intersect the visibility from those segments to generate visibility for the facet. This process is repeated for all facets. Third, we repeat step one and two for other slicing directions. These directions should be comprehensive (normal planes of slicing direction span the entire space). Fourth, we gather the visibility results from different slicing directions (a set of arcs) to form the final visibility results.



(a) Compute segments' visibility



(b) Convert segment visibility to facet visibility



(c) Compute visibility on differently oriented planes

Fig. 4-10 The slice-geometry based visibility method

4.4 Implementation

4.4.1 Surface Offsetting Results

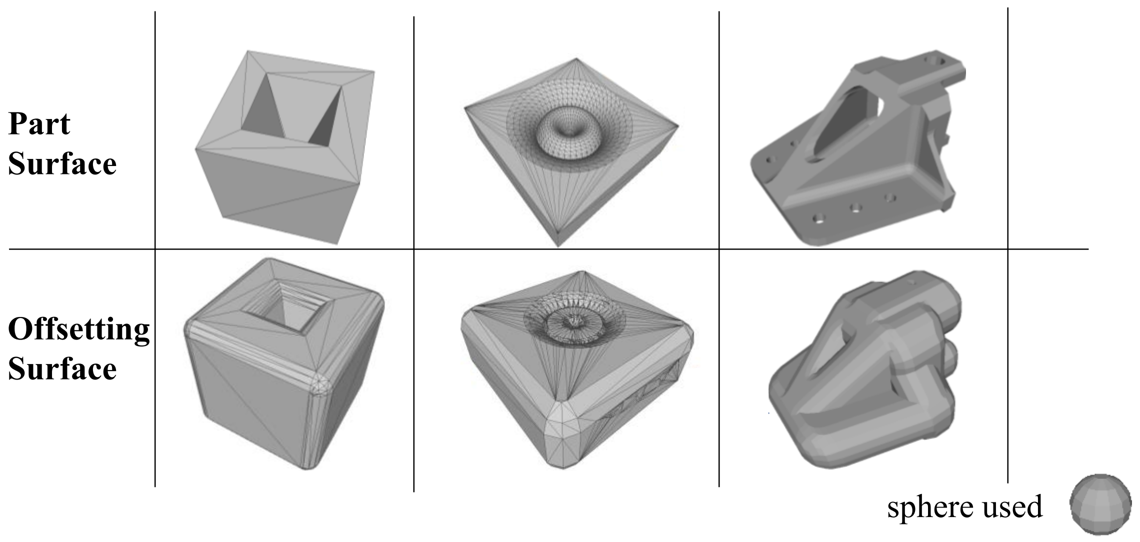


Fig. 4-11 Surface offset results for three models (via OpenSCAD)

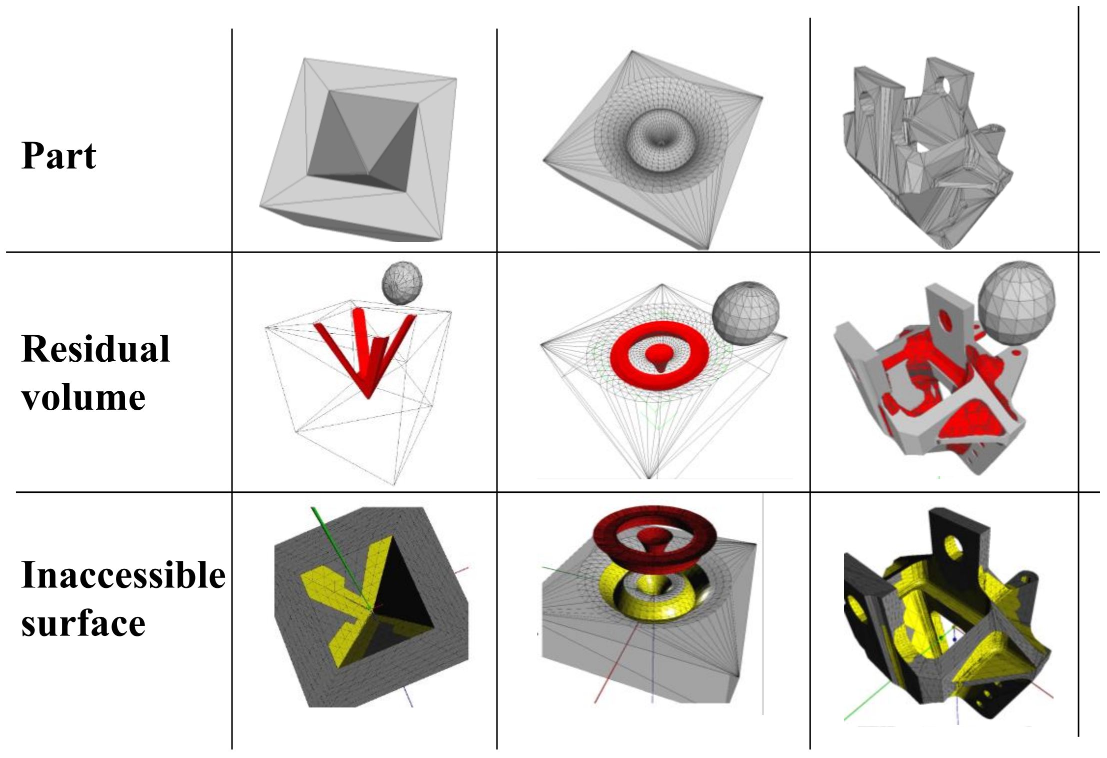
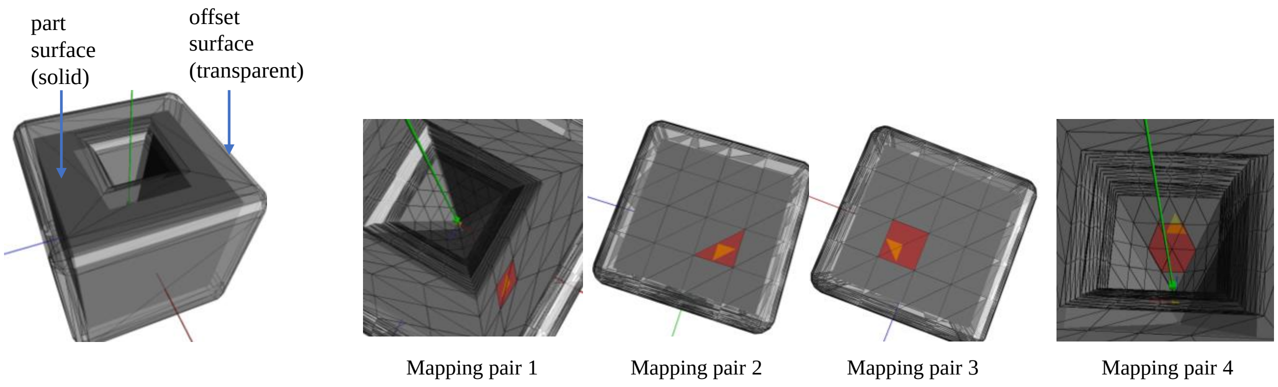


Fig. 4-12 Inaccessible surface results (yellow)

4.4.2 Cutter Location Determination (for a Part Facet) Results



Part and offset surfaces

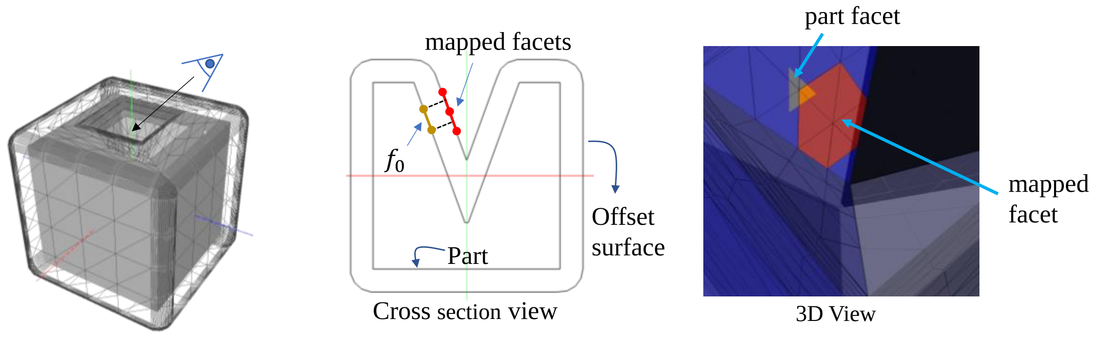
Mapping: Part facet (yellow) → Offset surface facets (red)

Fig. 4-13 Mapping from part faces to offset surface faces (four cases of mapping)

The example part shown in Fig. 4-13 is a cube with a tapered pocket on the top. In all 5 subfigures, the offset surface (using 3D Minkowski Sum, offset distance 0.5 inch) is shown in transparent mesh. The four subfigures on the right show four pairs of mapping. In each, the orange facet indicates the facet on the part surface while red facets indicate mapped facets on the offset surface, respectively. Depending on the size and position of the part facet, its mapped facets' count varies.

As we will conduct the intersection of visibility on the mapped facets, the more the mapped facets resemble the part facet, the more accurate the result is for the part facet. To better approximate the part facet, the polyhedron of the offset surface can be re-meshed to generate smaller facets.

4.4.3 Accessibility Results



(a) Part and offset surface (b) Facet of interest and its mapped facets on offset surface

Fig. 4-14 A part facet f_0 and its image

To show accessibility results, we use the part shown in Fig. 4-14a as an example. The part is a cube with a tapered pocket on the top. The offset surface of the part is shown as transparent which is offset outward by a tool radius of 0.5 inch (offset using 3D Minkowski Sum). We choose a facet f_0 on the part surface as an example (Fig. 4-14b). In the cross-section view, f_0 is shown as a yellow segment while in the 3D view, f_0 is shown as yellow triangle. In both views, we can see f_0's mapped facets (on the offset surface) are shown in red. Notice, the image of f_0 has intersected with 6 facets on the offset surface.

We then use the slice geometry-based visibility method to compute visibility for each of the mapped facets (Fig. 4-15). The visibility results for the six mapped facets are shown in six different colors. The leftmost subfigure represents visibility results on the unit sphere. For clarity, we separate these visibility results and show them one by one in the top view. The results show that the six different mapped facets have different visibility regions.

To obtain accessibility of f_0, we conduct an intersection of these six visibility regions (rightmost subfigure).

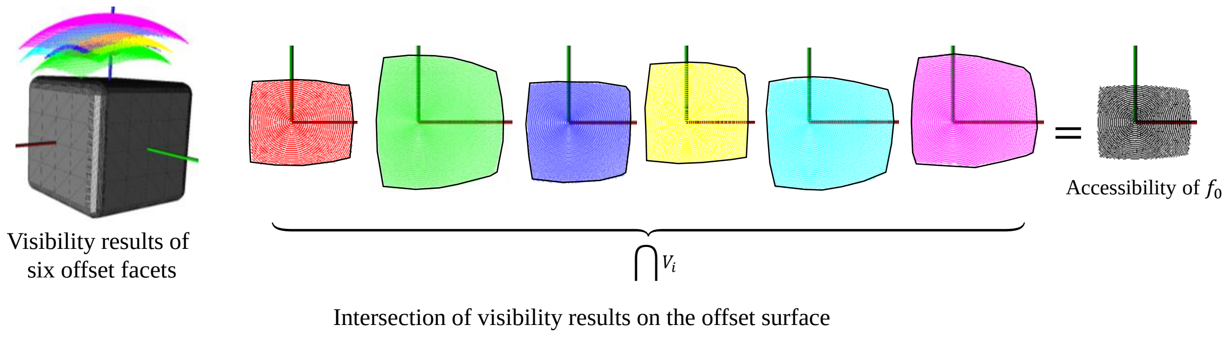
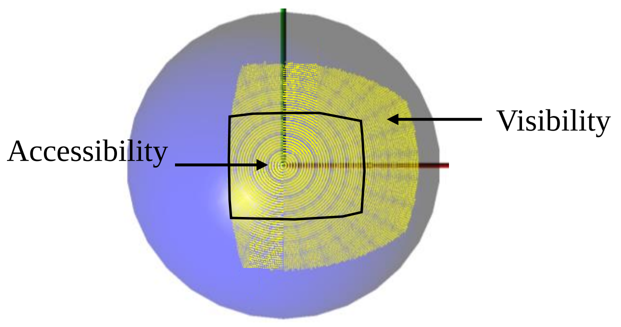

Visibility results of six offset facets
=
Accessibility of f_0
$\bigcap V_i$
Intersection of visibility results on the offset surface

Fig. 4-15 Visibility results of six mapped facets and their intersection

Intuitively, accessibility, which considers tool body for collision, should be more restricted than visibility. To verify this, we compute the visibility of f_0 and compare it with f_0's accessibility (Fig. 4-16). It can be seen that the accessibility region is considerably smaller than the visibility region which agrees with our expectation. The difference is partially attributed to the extra collision induced by the tool body and partially attributed to the intersection of visibility (namely approximating the image of f_0 as the union of mapped facets).



Accessibility versus Visibility for f_0

Fig. 4-16 Comparison between accessibility and visibility

4.5 Conclusion

This paper presented a new method to calculate the accessibility of a machine tool to a part surface. Instead of directly evaluating accessibility, we propose that the equivalent can be found through considering the visibility of an offset surface. This offset surface, found using a Minkowski Sum, allows us to consider an exact solution to visibility, and then map the visibility of the offset surface facets to the original part surface facets. It was shown that the use of an offset surface effectively accounts for the body of the tool in collision for accessibility computation. The method provides more than a Boolean solution to accessibility, rather, it calculates all feasible orientations of a ball-end tool. The results can be used for setup and toolpath planning in multi-axis CNC machining.

The method essentially reuses the visibility results, making it possible to adopt other visibility methods in the future. However, the 3D Minkowski Sum is computationally expensive. Computing the offset surface of a complex model with a fine resolution sphere mesh may take too long to process, making it less practical for rapid manufacturing efforts. Since the sphere mesh is always an approximation of the true sphere, the offset surface is always an approximation which leads to some inaccuracy of the results. Also, due to the triangulation difference between the offset surface and part surface, the accessibility results are always computed from the intersection of the *mapped facets* which is always conservatively approximated, making it less accurate. Moreover, this method only applies to ball-end cutter. In the future, work can be done on developing a more efficient surface offset algorithm considering the geometry of the specific type of tool used. In addition, a re-triangulation on the offset surface that guarantees each accessible part facet has an identical cutter location image should be developed to make the accessibility results more accurate.

4.6 References

[1] Miller, G., 1994, "Efficient algorithms for local and global accessibility shading," Proceedings of the 21st annual conference on Computer graphics and interactive techniques, ACM, pp. 319-326.

[2] Elber, G., 1994, "Accessibility in 5-axis milling environment," Comput Aided Design, 26(11), pp. 796-802.

[3] Tang, K., Cheng, C. C., and Dayan, Y., 1995, "Offsetting surface boundaries and 3-axis gouge-free surface machining," Comput Aided Design, 27(12), pp. 915-927.

[4] Kim, S.-J., and Yang, M.-Y., 2005, "Triangular mesh offset for generalized cutter," Comput Aided Design, 37(10), pp. 999-1014.

[5] Xu, X. J., Bradley, C., Zhang, Y. F., Loh, H. T., and Wong, Y. S., 2002, "Tool-path generation for five-axis machining of free-form surfaces based on accessibility analysis," Int J Prod Res, 40(14), pp. 3253-3274.

[6] TheComputationalGeometryAlgorithmsLibrary, 2017, "3D Minkowski Sum of Polyhedra," https://doc.cgal.org/latest/Minkowski_sum_3/index.html.

[7] Li, W., and McMains, S., 2014, "A sweep and translate algorithm for computing voxelized 3D Minkowski sums on the GPU," Comput Aided Design, 46, pp. 90-100.

[8] Lien, J.-M., 2010, "A Simple Method for Computing Minkowski Sum Boundary in 3D Using Collision Detection," Algorithmic Foundation of Robotics VIII: Selected Contributions of the Eight International Workshop on the Algorithmic Foundations of Robotics, G. S. Chirikjian, H. Choset, M. Morales, and T. Murphey, eds., Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 401-415.

[9] OpenSCAD, 2017, "The Programmers Solid 3D CAD Modeller," http://www.openscad.org/downloads.html.

[10] Hou, G., and Frank, M. C., 2017, "Computing the Global Visibility Map Using Slice Geometry for Setup Planning," Journal of Manufacturing Science and Engineering.

CHAPTER 5. A HYBRID APPROACH TO COMPUTE THE VISIBILITY MAP OF A POLYHEDRON

Abstract

This paper proposes a new visibility determination method that combines two independent visibility methods to balance computational cost and accuracy. One of them is an approximate method that computes a facet's visibility on a set of selected planes. The other is an exact method that uses a boundary tracing technique to determine the exact boundary of non-visibility of one facet due to the other. The complete non-visibility of a facet will consider all other facets as obstacles. To speed up the exact computation which has a higher time complexity than the approximate, we create an obstacle filter that generates considerably fewer candidate obstacles for the exact method. This filter is created using the approximate method's Quick Visible Region Identification functionality. In this way, we incorporate both methods in visibility determination that generates near-exact results. This hybrid method also provides easy control over the approximate-exact computation ratio. By adjusting this ratio, we realize a balance between computational cost and accuracy.

5.1 Introduction

Toolpath planning for Computer Numerical Control (CNC) machining has been a long-term research focus in the field of high precision manufacturing. Its intricacy is driven by the complex geometry of the CAD model, the mechanical characteristics of cutting, the optimization considering multiple parameters and the choice among a variety of toolpath strategies. Generating a feasible toolpath for an arbitrary model is no easy work. Some researchers still rely on the recognition of features to generate specific types of toolpaths [1].

Others consider a freeform surface and develop their toolpath strategy based on the concept of visibility or reachability [2]. In which, visibility determination becomes a prerequisite step in a toolpath's generation.

The methods to compute visibility divide into two categories in general: approximate and exact methods. Although an approximate method is usually faster because of adjustable resolution and approximation ratio, the inexactness may not be preferred in certain cases. The exact methods generate accurate visibility boundaries but are usually high in time complexity. This dilemma leads to choices that either sacrifice accuracy for processing time or vice versa. Often, engineers and researches choose one type that fits better into their immediate needs. This work attempts to merge the two approaches in a hybrid method that accomplishes both goals.

5.2 Related Work

There has been considerable research that addresses visibility computation, including the seminal method by Chen and Woo that used a Gaussian map [3], the Z-buffer method [4], discrete avoidance method [5], Minkowski Sum method [6], exact visibility methods using pairwise occlusion [7], and sliced geometry-based methods [8]. As modern machine's computational performance rises, researchers nowadays tend to use the GPU for scientific computation. Some researchers make use of the "occupation query" of GPU for visibility computation [2]. Some use the GPU for accessibility testing, which considers both the tool body and the tool holder [4]. In general, the time complexity of exact visibility computation that use pairwise occlusion is $O(n^2)$. The time complexities of approximate methods vary but they are more efficient than exact method in most cases.

One slice geometry-based approximate method first creates cross sections (slices) of the model using multiple parallel planes, then computes visibility on these 2D slices [8, 9]. It obtains 3D visibility by merging visibility from 2D slices which essentially reduces a 3D problem to a 2D one. The time complexity of this method is $O(k \cdot n)$, where n is the part mesh size, k is the total number of slices (among all directions). It has better time complexity than an approximate method that discretizes the visibility space (whose time complexity is $O(mn^2)$, where m is the discretization size of visibility space) and the exact methods mentioned above. Although efficient, it is still computationally costly if accuracy and resolution requirements are high. Besides, it generates inexact visibility results compared to exact methods.

In summary, a method that can generate exact or near-exact visibility results while also balancing processing time and accuracy is highly desirable. In this paper, we will introduce one such method by using an approximate method to preprocess the visibility and filter out unlikely obstacles for the exact method; which eventually reduces the processing time for the latter.

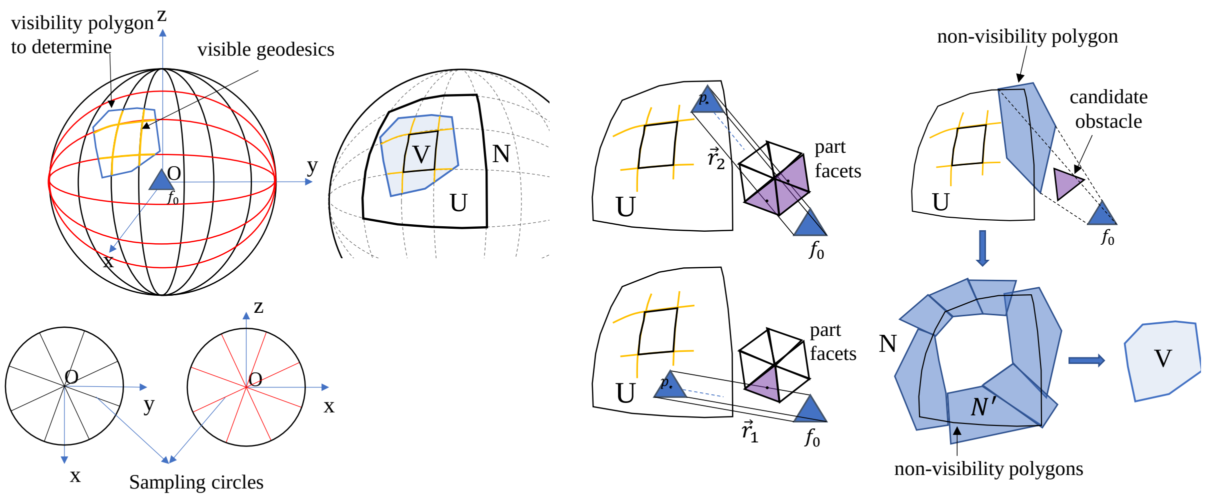
5.3 Methodology

The following analysis uses an example facet f_0 to illustrate the visibility determination process. We denote the approximate visibility method V1 as in [8]; denote the exact visibility method V2 as in [7].

5.3.1 Method Overview

The method has four steps (as illustrated in Fig. 5-1). And they are described in detail in the following paragraphs.

1. Visibility Evaluation on Selected Circles. This computes the visibility of facet f_0 on two sets of sampling great circles using V1. The first set are coaxial about the Z-axis. The second set are coaxial about the Y-axis. Both sets of great circles span the unit sphere homogeneously.
2. Visible Region Classification. The visibility results are represented as a set of geodesics on the unit sphere. We divide the unit sphere into three regions based on whether an area is enclosed or partially enclosed by geodesics. These three regions are marked as Visible (V), Nonvisible (N) and Undetermined (U), respectively.
3. Candidate Obstacles Determination. We extrude a 3D beam from base f_0 following a direction $\vec{r}$, where $\vec{r}$ is the line connecting the centroid of f_0 and a point p in the Undetermined region (U). Then, we move p in region U while finding all part facets that intersect the beam. These part facets S_{obs} are the obstacles of f_0 that will contribute to the visibility boundary in region U.
4. Non-visibility Computation and Final Non-visibility Merging. Finally, we compute the non-visibility for each facets pair $(f_0, f_i), \forall f_i \in S_{obs}$ using V2 and then union the results, denoted as N'. The final non-visibility is the union of N and N'. The complement of $(N \cup N')$ gives the final visibility of f_0.



(a) Step1: Visibility computation on sampling circles

(b) Step 2: Region classification by visible geodesics

(c) Step 3: Candidate obstacles determination by intersection

(d) Step 4: Non-visibility computation for all obstacles against f_0

Fig. 5-1 Overview of the hybrid visibility method

5.3.2 Undetermined Visible Region Determination

5.3.2.1 The Slice Geometry-based Visibility Method

In this paper, we use a slice geometry-based visibility method from our previous work to compute visibility (Fig. 5-2) [8]. Following is a review of the four main steps:

First, we select a set of parallel planes cutting the part into a set of slices. Each slice consists of a set of segments. A 2D visibility algorithm is conducted on each slice to compute visibility for each segment. The direction perpendicular to these parallel planes is called a slicing direction. Second, for each facet sliced by a set of segments, we intersect the visibility from those segments to generate visibility for the facet. This process is repeated for all facets. Third, we repeat step one and two for other slicing directions. These directions should be comprehensive (normal planes of slicing direction span the entire space). Fourth, we gather the visibility results from different slicing directions (a set of arcs) to form the final visibility results.

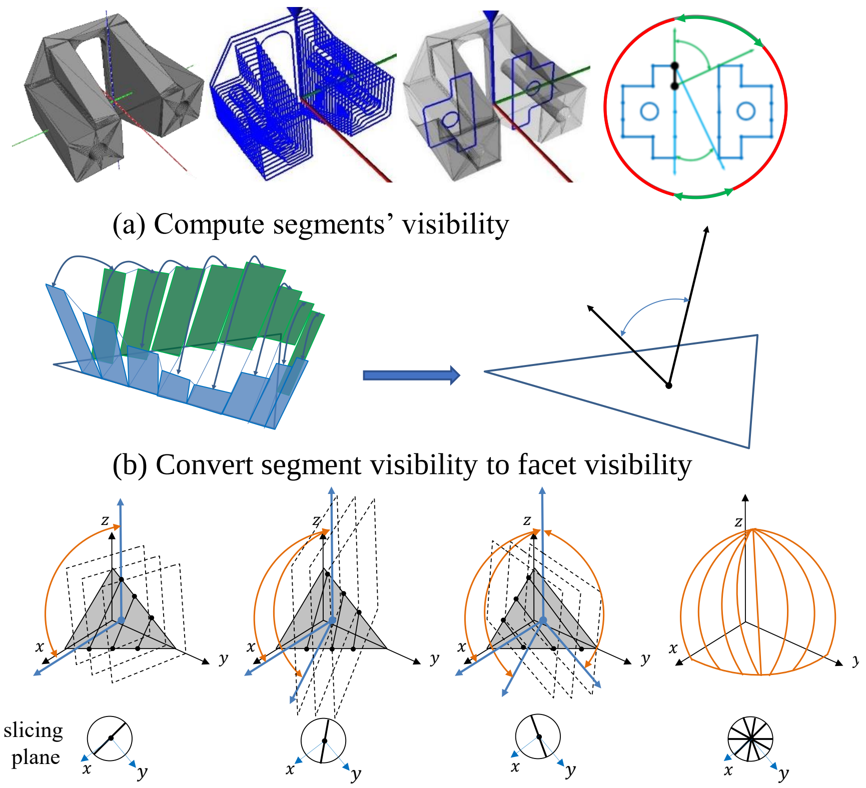

(a) Compute segments' visibility
(b) Convert segment visibility to facet visibility
z
x
y
slicing plane
(c) Compute visibility on differently oriented planes

Fig. 5-2 The slice geometry-based visibility method

5.3.2.2 Visibility Sampling Grid on Unit Sphere and Visible Region Classification

Assuming a part is single-bodied, we introduce a special property of its visibility polygons: A visibility polygon must be simple polygon without holes (a hole is a nonvisible region).

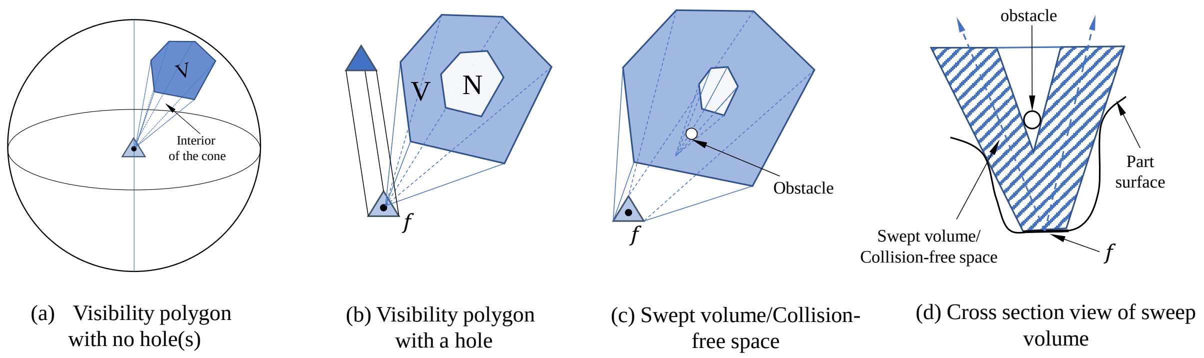

V
Interior of the cone
V
N
f
Obstacle
f
obstacle
Part surface
Swept volume/ Collision-free space
f
(a) Visibility polygon with no hole(s)
(b) Visibility polygon with a hole
(c) Swept volume/Collision-free space
(d) Cross section view of sweep volume

Fig. 5-3 A special property about visibility polygons

This property can be proven by contradiction. Suppose the visibility polygon contains a hole, then the visibility is in a ring shape (Fig. 5-3b) where the hole is a non-visibility area. By the definition of visibility, the 3D beam extruded from face f, directed at points in the visibility ring, will sweep out a collision-free volume in space (Fig. 5-3c, d). Therefore, this swept volume contains no solid. On the other hand, there must exists a solid above the swept volume, acting as an obstacle that generates the non-visibility area. The swept volume has divided the 3D space into two subspaces where the base f and obstacle solid are on different subspaces. Because we assume our part is single-bodied, this separation of base f and obstacle solid introduces a contradiction. Therefore, the visibility polygon for a single-bodied part contains no (non-visibility) holes.

This special property leads to a corollary: A spherical region whose boundary is visible is also visible in the interior (Fig. 5-4).

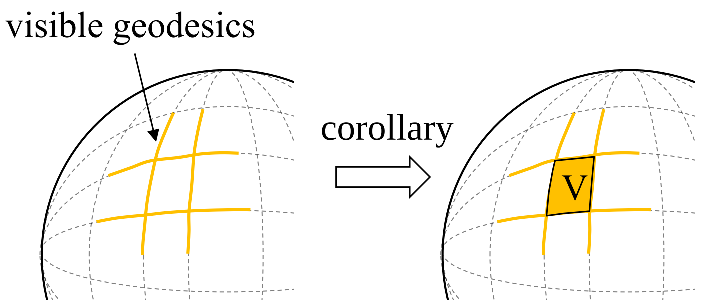
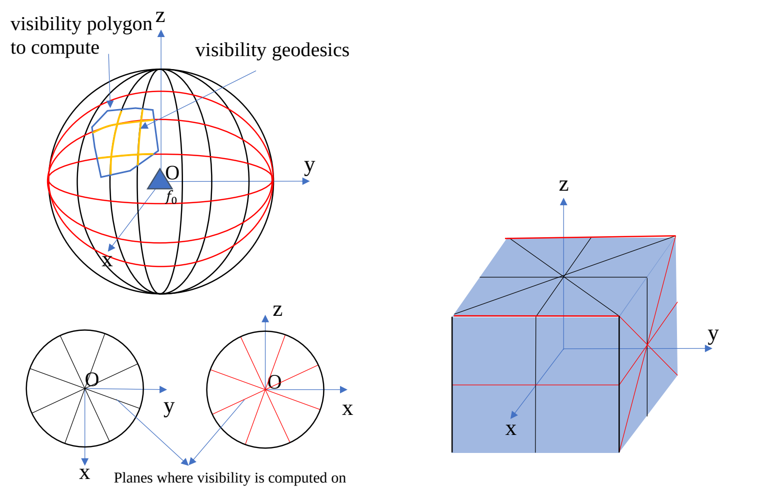


Fig. 5-4 A corollary for spherical region closed by visible geodesics

Using this corollary, we could quickly identify the visible regions given visible geodesics on the unit sphere that form closed regions. We call this the Quick Visible Region Identification (QVRI) technique.

One unique feature of the slice geometry-based visibility method (V1) is that visibility can be evaluated on selected great circles on the unit sphere. Doing so will generate a set of visible geodesics on the unit sphere. We can use this feature for QVRI. If a true visibility polygon is sampled by great circles and contains regions closed by visible geodesics, we can mark those regions visible and leave the remaining area to be determined later.

To create regions closed by visible geodesics, we establish a grid made by two sets of great circles on unit sphere where one set intersects the other. This grid subdivides the unit sphere into multiple small regions which we call *cells*. In this paper, we choose the first set of great circles to align with meridians (suppose north pole is to the positive Z direction); and the second set of great circles to be coaxial about Y-axis. This layout is shown in Fig. 5-5. The angle between adjacent great circles in a set is uniformly set to 5 degrees (adjustable). We then use V1 to evaluate visibility on these great circles. We define these two sets of great circles as *sampling circles* or simply a *sampling grid* together.


visibility polygon to compute
z
visibility geodesics
y
O
f_0
x
O
y
x
z
O
x
Planes where visibility is computed on
z
y
x

(a) Two sets of great circles for visibility computation

(b) The cubic projection of the two sets of great circles

Fig. 5-5 Two sets of great circles to compute visibility using V1

The intersections of these sampling circles create two types of spherical polygons: spherical triangles and spherical rectangles. In the following analysis, we use spherical rectangles for illustration.

After visible geodesics are determined on the sampling grid, we classify the unit sphere into three regions:

- **Visible Region** where a cell's boundary is visible.
- **Undetermined Region** where a cell's boundary contains at least one visible geodesic but is not completely visible.
- **Nonvisible Region** which covers all the remaining cells.

An example of the Visible Region Classification process is shown in Fig. 5-6.

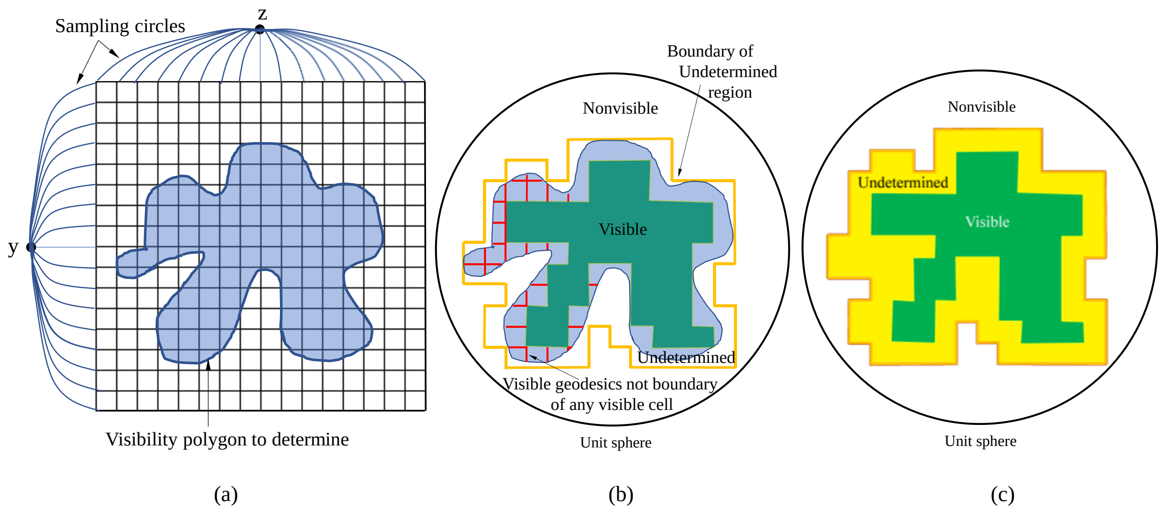

Sampling circles
z
y
Visibility polygon to determine
Boundary of Undetermined region
Nonvisible
Visible
Undetermined
Visible geodesics not boundary of any visible cell
Unit sphere
Nonvisible
Undetermined
Visible
Unit sphere
(a)
(b)
(c)

Fig. 5-6 Visible Region Classification. (a) Sampling circles and the visibility polygon to determine; (b) Locating Visible and Undetermined regions. Not all visible geodesics are shown; (c) The final Visible, Undetermined and Nonvisible regions.

After Visible Region Classification, the visibility results in Visible and Non-visible region are fully determined. The visibility in the Undetermined region is unclear. The cells in the Undetermined region could be either partially visible or completely invisible. In fact, the Undetermined region contains the boundary of the true visibility polygon of f_0. To determine this boundary, we use the exact visibility method V2. The boundary separate visibility and non-visibility. Thus, there must exist some obstacle facets whose occlusions establishes the exact shape of this boundary. In the following section, we will find such obstacle facets.

5.3.3 Exact Visibility Boundary Determination in the Undetermined Region

5.3.3.1 Candidate Obstacles Determination for the Undetermined Region

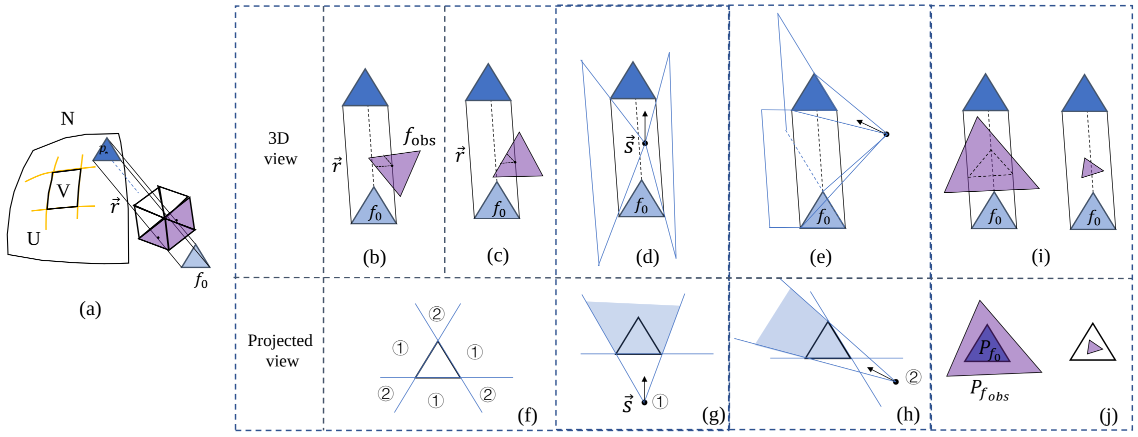


Fig. 5-7 Intersection test between the 3D extruded beam and part to discover obstacles that contribute to the forming of visibility boundary in Undetermined region

To find the obstacles (part facets) that contribute to the forming of visibility boundary of f_0 in the Undetermined region, an intersection test between the part and a 3D beam is conducted. Any part facet that intersects with the 3D beam (extruded from f_0 following direction $\vec{r}$ where $\vec{r}$ starts from the centroid of f_0 and point at any point p in the Undetermined region U) will contribute to the forming of a visibility boundary in the Undetermined area U.

The intersection between a triangle and a 3D extrusion beam has three scenarios. First, at least one vertex of the triangle is in the beam (Fig. 5-7b). This can be checked by testing if any triangle vertex is in all negative spaces of the side planes of the 3D beam. Second, no triangle vertex is in the 3D beam, but at least a triangle edge intersects with the 3D beam (Fig. 5-7c). In the projected view (projecting everything in the $\vec{r}$ direction onto a plane that is parallel to f_0), the edge of f_0 has divided the plane into 6 subspaces. These subspaces can be categorized into two types; Type one: any point in this subspace that is in the positive space of side plane and negative space of the other two side plane, and Type two: any point in this subspace that is in the positive space of two side planes and negative space of the other side plane (Fig. 5-7f). For any 3D segment $\vec{s}$, one vertex of $\vec{s}$ must fit into the two categories above. For $\vec{s}$ to interest with the 3D beam, the other vertex of $\vec{s}$ must lie in the space bounded by the planes shown in (Fig. 5-7(d-e)). The subfigure in Fig. 5-7(g-h) illustrates the respective projected view. Essentially, it is testing whether the other vertex of $\vec{s}$ is in the bounded space of corresponding planes. We can conduct this test for each edge of the obstacle triangle. If any edge intersects with the beam, the triangle intersects with the beam.

The third case is neither a vertex nor an edge of the triangle intersects with the 3D beam. In this case, we denote the projected image of f_0 as P_{f_0} and projected image of obstacle triangle f_{obs} as $P_{f_{obs}}$. Then, the obstacle triangle f_{obs} intersects the 3D beam if and only if P_{f_0} is contained by $P_{f_{obs}}$ or P_{f_0} contains $P_{f_{obs}}$ (Fig. 5-7j). To determine if a triangle is contained by another triangle, we can test whether all the vertices of one triangle is in the other triangle.

In conclusion, the time complexity for testing whether an obstacle facet intersects with a 3D beam is linear to the number of edges of f_0 and number of edges of f_{obs}. Because both facets are triangles in this paper, the edges count are 3 for both. Therefore, the time complexity of the intersection test is constant.

The above intersection test describes how to find the intersected part facets for one 3D beam direction $\vec{r}$. To account for all directions of $\vec{r}$, we need to traverse the endpoint of $\vec{r}$ in the undetermined region U. Since the undetermined region consists of many cells, we can traverse $\vec{r}$ for each cell and repeat the traversal for all cells. Although it is ideal to enumerate

$\vec{r}$ for all points in U, this is not practical. In this paper, we only consider the boundary vertices of each cell, assuming the intersecting facets introduced by these 3D beams captures all potential obstacle facets for that cell (Fig. 5-8). This is reasonable because our sampling great circles have small angle interval, the 3D beams introduced by the boundary vertices are heavily overlapped and occupy most of the swept volume if $\vec{r}$ is traversed for the entire cell. Therefore, the chance that an actual obstacle facet does not intersect with any of the boundary 3D beams is negligible.

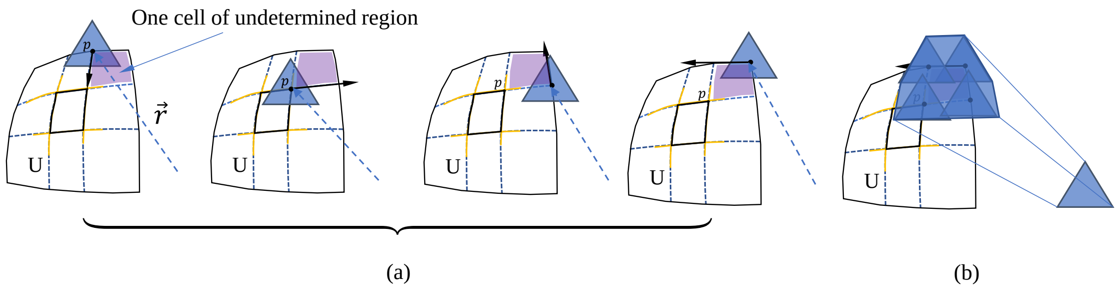


Fig. 5-8 Enumerate directions $\vec{r}$ at cell boundary vertices in the Undetermined region

In summary, the 3D beam intersection tests will be conducted for all cell's boundary vertices in the Undetermined region. The resulting part facets (candidate obstacle facets) will be used to compute non-visibility which will form the boundary of f_0's visibility polygon(s).

5.3.3.2 The Boundary Tracing Method for Exact Visibility Boundary

From the previous section, we have a base facet f_0 and a set of other facets that are obstacles of f_0. The goal is to compute the non-visibility polygon(s) of f_0 due to this set of obstacles. It can be realized in two steps. First, we compute the non-visibility of f_0 due to each obstacle facet; Second, we union the non-visibility regions due to each obstacle.

Given a pair of facets (one being the base facet whose visibility is to be determined the other being an obstacle), how can we determine the space that is obstructed by the obstacle for the base facet? The problem can be reformulated as follows: First, translate the base facet in space and form a 3D light beam by connecting the translated facet and the original base facet vertex-by-vertex. Second, extend the 3D light beam so it is infinite in length. Third, move the translated facet while keeping the infinite long 3D beam moving at the same time. Forth, if at any time, the 3D beam intersects with the obstacle facet, the direction determined by the segment connecting the centroid of the base facet and translated facet, is an invisible direction. The goal is to find the boundary between visible and invisible directions.

To solve this problem, we make use of a boundary tracing visibility method that determines the non-visibility region of one facet due to the other by computing sliding planes [7]. This method gives the exact non-visibility results in the form of spherical polygons on the unit sphere for a pair of facets. However, it does not address how to obtain the complete non-visibility for a facet, which essentially requires us to union the non-visibility polygons, either on the unit sphere or on a plane. To this end, a modified method is implemented from scratch using the CGAL and s2Geometry library [10, 11]. We use s2Geometry library for its capability of conducting Boolean operations of spherical polygons on the unit sphere.

An example of an exact visibility boundary represented as spherical polygons on the unit sphere is shown in Fig. 5-9. Following is the detail of this method.

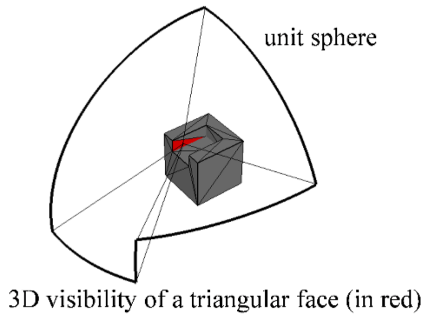


Fig. 5-9 The exact visibility boundary shown as spherical polygon on unit sphere

In order to compute the non-visibility of a base facet due to an obstacle facet, a boundary tracing approach is used (Fig. 5-10). This approach extrudes a 3D light beam from the base facet where all the side edges are parallel. Then the top cap of 3D light beam is pushed against the obstacle facet until touching (vertex/edge, vertex/vertex or edge/edge). Once touching, we trace the 3D beam along the boundary of the obstacle facet. The trace (of the beam's center line) results in a 3D cone which defines the boundary of directions where the 3D beam will collide with the obstacle. This 3D cone can be projected to a plane parallel to the base facet, creating a non-visibility polygon. It can also be projected to a unit sphere, creating a non-visibility spherical polygon.

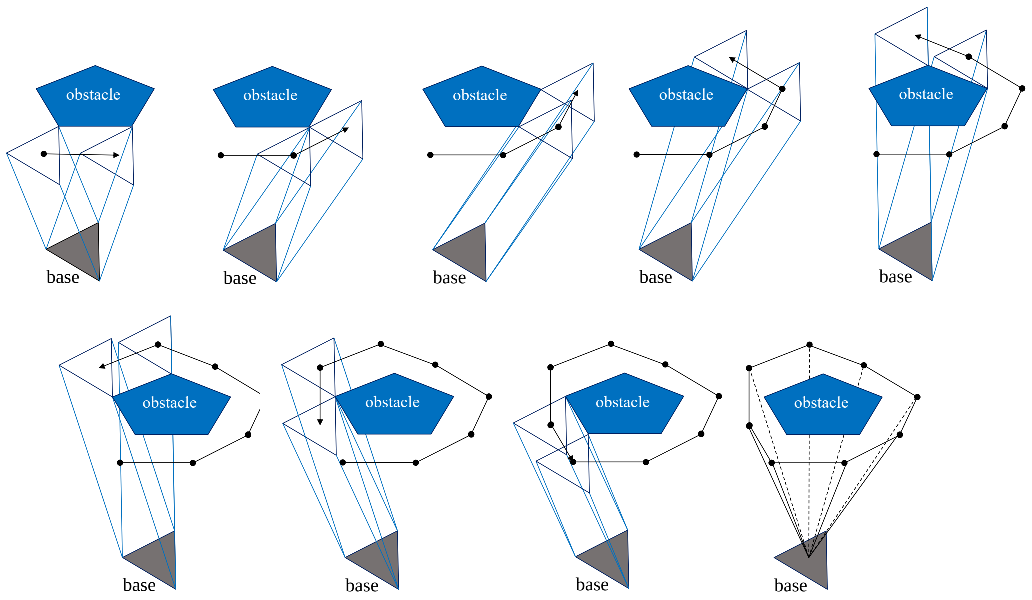


Fig. 5-10 Boundary tracing algorithm for a pair of faces

To quantify the 3D cone, we exam the tracing process. Notice, the tracing is based on the contact of the cap and obstacle facet. The contact pair, namely a vertex or edge from the cap and a vertex or edge from obstacle could change over the tracing process. The 3D beam can move if the contact pair contains an edge. In fact, the 3D beam can slide along the plane determined by the contact pair that contains an edge. These planes are called *sliding planes*.

A sliding plane, due to the boundary tracing nature, subdivides the 3D space into two subspaces where the obstacle facet and base facet are in different subspaces. Therefore, sliding planes can be found by examining planes defined by contact pairs (formed by the cap and obstacle's vertices and edges). The planes that separate the cap and obstacle facet are the sliding planes (Fig. 5-11).

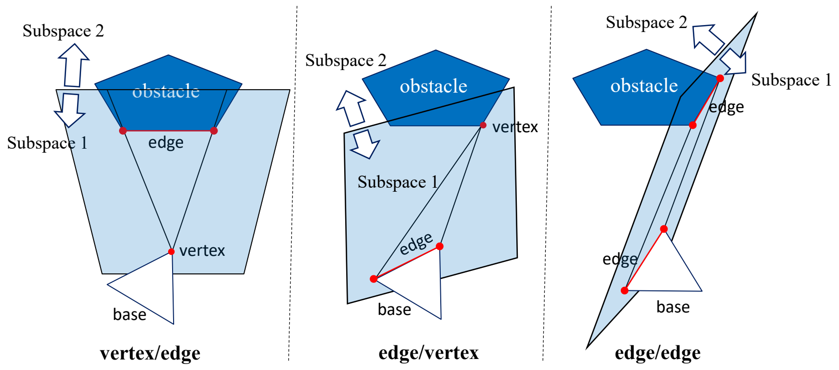


Fig. 5-11 Contact pair and sliding planes

Once sliding planes are determined, the 3D cone for non-visibility is determined (the 3D cone can be created by intersecting the negative subspace of each plane, assuming negative space is where the obstacle facet is, denoted as II subspace). To represent non-visibility results on a plane, we could intersect the 3D cone with a plane P parallel to base facet. Or equivalently, we could intersect sliding planes with plane P first (creating a set of oriented 2D lines) and intersect the negative space of these 2D lines to generate non-visibility polygon in 2D (Fig. 5-12).

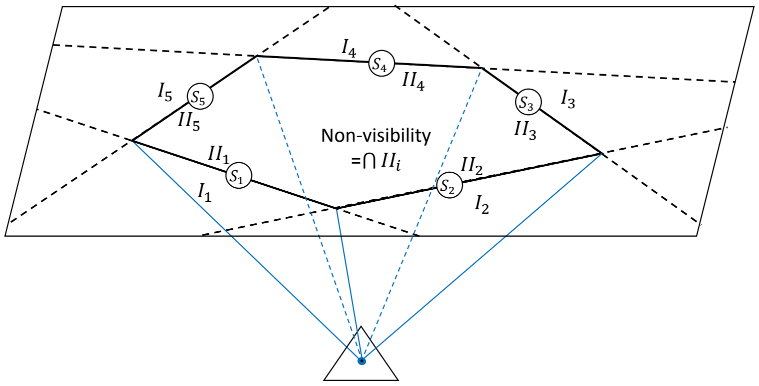


Fig. 5-12 Half spaces intersection for non-visibility. S_i denotes sliding plane i; I_i denotes visible half-space i; II_i denotes the complementary half-space of I_i.

As an example, Fig. 5-13 shows the non-visibility cone of a base facet (on x-y plane) due to an obstacle facet above. The results are represented on the $z = 1$ plane as well as on the unit sphere.

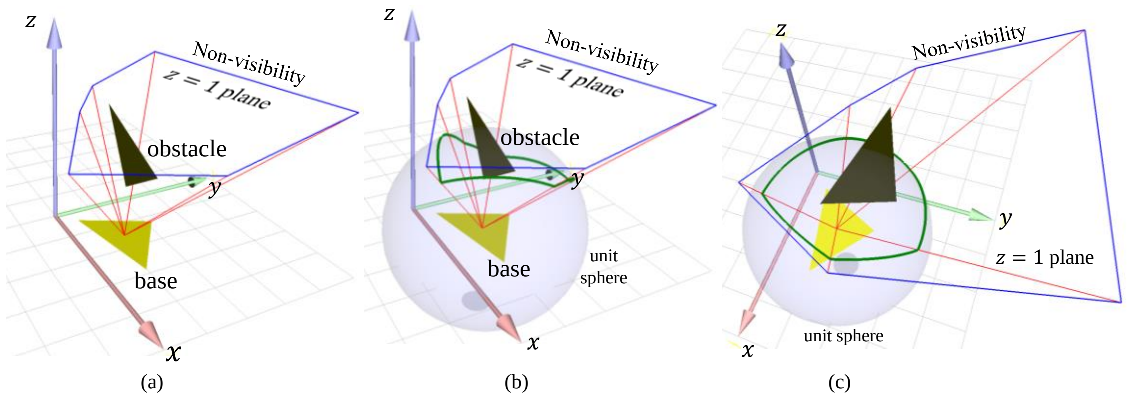


Fig. 5-13 Non-visibility polygon of a base facet due to an obstacle. (a) Represented on the plane $z = 1$, intersection is in blue, (b) Represented on unit sphere, intersection is in green. (c) Same as (b), but different view.

Now, the algorithm to compute non-visibility for one obstacle is complete. The next is to compute non-visibility for a set of obstacles and union them. Because the visibility space for a facet is at most a hemisphere and the images on a hemisphere can always be projected to the same plane, we could then represent the non-visibility results of a facet due to different obstacles to a plane that is parallel to the base facet. Fig. 5-14 shows an example of representing non-visibility results of a base facets due to five different obstacles.

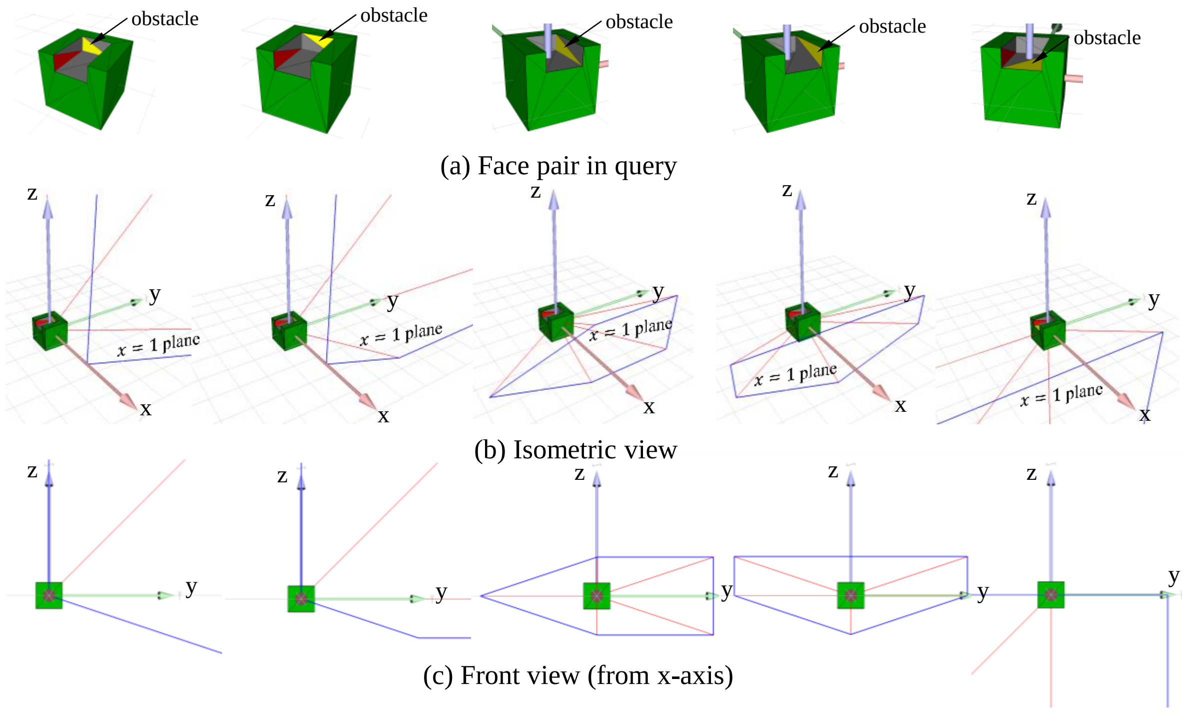


Fig. 5-14 Non-visibility polygons for a base face (in red) against five obstacle faces (in yellow). Non-visibility Polygons are in blue. (a) Five obstacle facets. (b) The non-visibility polygons for the corresponding face pair in isometric view. (c) Same as (b), but in front view.

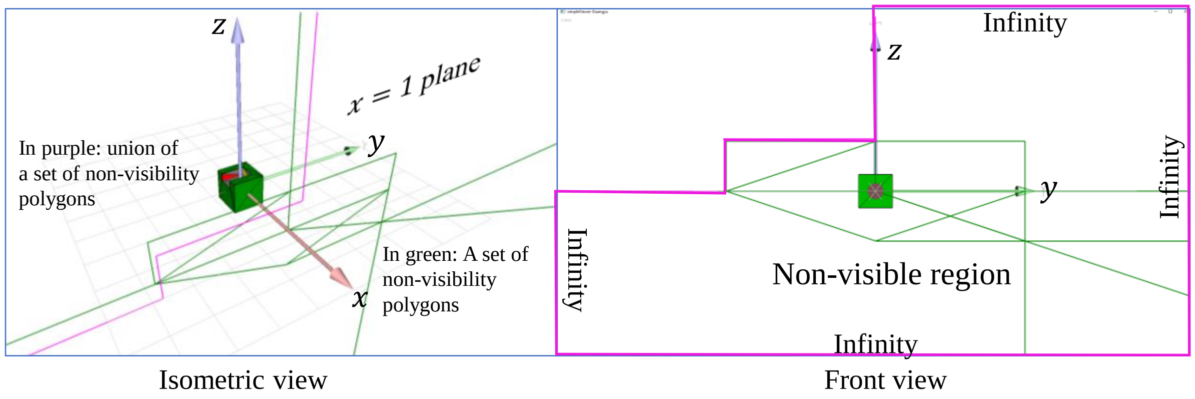

z
$x = 1$ plane
y
x
In purple: union of a set of non-visibility polygons
In green: A set of non-visibility polygons
Isometric view
Infinity
Infinity
Infinity
Infinity
Non-visible region
Front view

Fig. 5-15 A collection of non-visibility polygons (in green) for a facet (in red); The union of non-visibility polygons (in purple).

After we have the non-visibility polygons generated from different obstacles, we conduct a union of these 2D polygons. The union is the final non-visibility result of the base facet due to a set of obstacles (example given in Fig. 5-15). Notice, once we get the non-visibility results on a plane, we can project them onto the unit sphere for a better view.

5.3.3.3 Visibility Boundary Computation in Undetermined Region

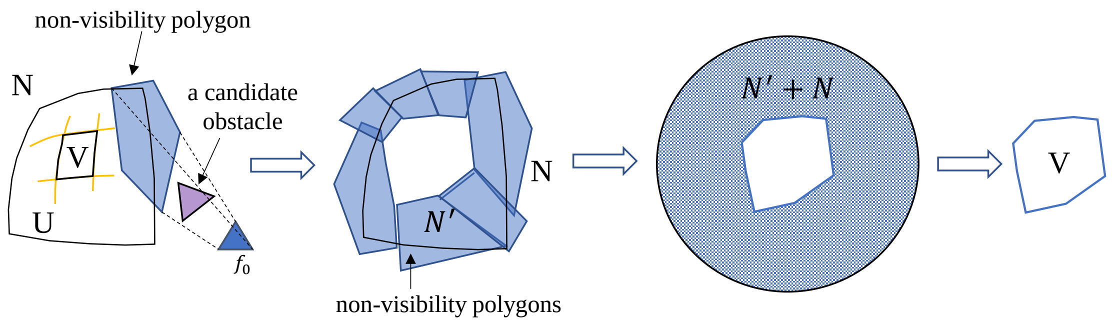
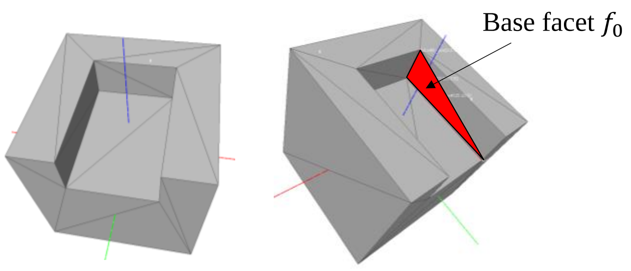


Fig. 5-16 Compute non-visibility in the Undetermined region; Union and complement for visibility polygon

Applying the boundary tracing visibility algorithm to the candidate obstacles for facet f_0, we get a set of non-visibility polygons on the unit sphere for f_0, defined as N'. Now, we union the Nonvisible region N with N'which results in the complete nonvisible region. By conducting a complement to $N \cup N'$, we get the final visibility polygon (Fig. 5-16).

5.4 Implementation

The implementation will be shown using the example model in Fig. 5-17 for clarity. It is a cube with a slot on the top face. The base facet to evaluate visibility is highlighted and referred to as f_0 in the following discussion. All other part facets are potential obstacles to this base facet.



(a) View angle 1 (b) View angle 2

Fig. 5-17 Test model

First, the visibility is computed for the entire model on sampling circles using visibility method V1. Recall that the sampling circles have an angle interval of 5 degrees. The visibility results for f_0 is shown in Fig. 5-18a. The results from V1 is in fact a set of continuous geodesics; therefore, they are represented as geodesics shown in Fig. 5-18b.

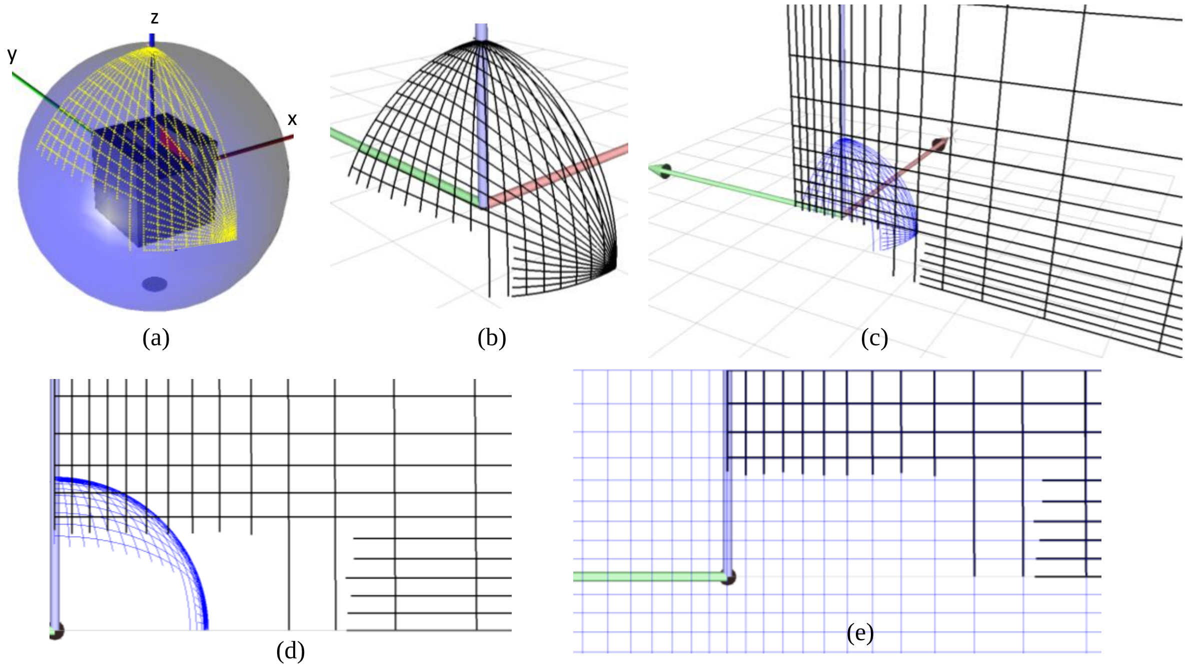


Fig. 5-18 The visibility results on sampling circles

Recall the visibility space for any facet is a hemisphere above it and any image in the hemisphere can be one-to-one mapped to the plane parallel to the facet f_0. We can therefore represent visibility on this parallel plane (referred to as plane P, Fig. 5-18c). Such plane projection makes it easier to find visible cells and geodesics that are not boundary to any visible cell (referred to as dangling geodesics). Now, the counterpart of geodesics are line segments on plane P (Fig. 5-18d). To distinguish the three visibility regions, namely Undetermined, Visible and Nonvisible, we also project the sampling circles to plane P (blue lines in Fig. 5-18e). Next, we will locate the dangling geodesics and the Undetermined region.

The line segments on the plane P form an arrangement which we can compute and represent using a Doubly-Connected Edge List (DCEL) data structure. The intersection of these line segments creates a set of vertices, segments and faces. It also creates an outermost unbounded face. We claim that all faces other than the unbounded face are visible cells. We also claim that if a line segment is surrounded by only the unbounded face, it is mapped from a dangling geodesic, referred to as a *dangling segment* in the following discussion. Traversing the segments on the inner hole of the unbounded face and conducting the check mentioned above will locate all dangling segments (red line segments in Fig. 5-19a). As the sampling interval is fixed and a dangling segment always connects to a vertex of some visible cell (referred to as v_0), we can find the two neighbor vertices of v_0, referred to as v_1 and v_2. Following the direction of the dangling segment, we can find the other three vertices v_3, v_4 and v_5. These six vertices define the left and right undetermined cell of the dangling segments (Fig. 5-19b). Repeating this for all dangling segments, we can find the Undetermined region. We claim that any cell that is neither a Visible cell nor an Undetermined cell, is a Nonvisible cell. The Visible Region Classification results are shown in Fig. 5-19c.

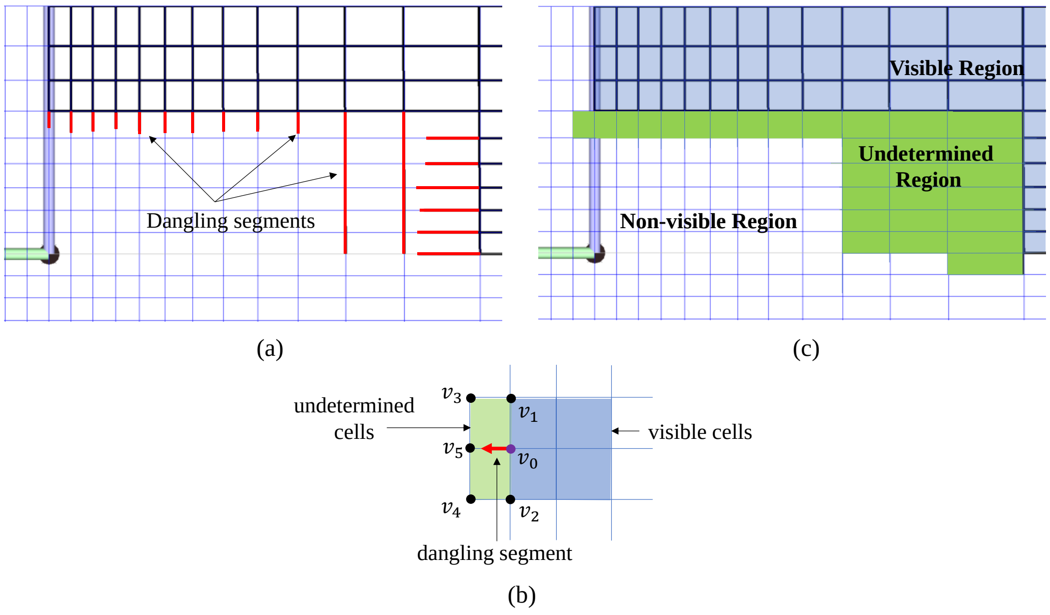


Fig. 5-19 Locating dangling segments and Undetermined region

To find the candidate obstacles contributing to the formation of the visibility boundary in the Undetermined region, we extrude a 3D beam from the base facet f_0, pointing to the vertices we obtained in the Undetermined region. The part facets that intersect with these 3D beams become the candidate obstacles for exact boundary computation using visibility method V2. Fig. 5-20 shows the results of the intersection between 3D beams and part facets. The final candidate facets are shown in Fig. 5-20f. Notice not all intersecting facets are counted, as useful obstacles must be in the same concave region with f_0.

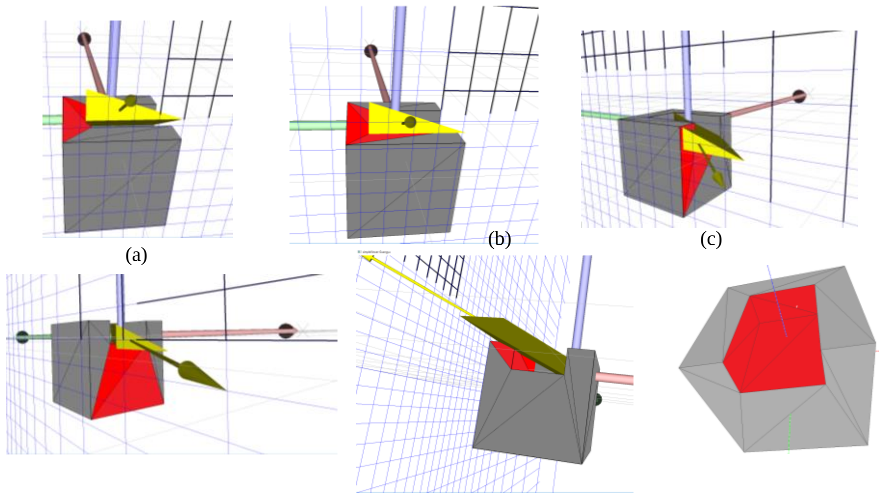


Fig. 5-20 Intersection between the part and 3D beams along different orientations; (a-e) Results in 5 different orientations; Intersecting facets are shown in red; 3D beam are shown in yellow. (f) The candidate obstacles considering useful obstacles must be in the same concave region.

Now, we have the candidate obstacle facets and we conduct the exact non-visibility computation on them against f_0 using V2. The subfigures in Fig. 5-21a shows the non-visibility results for f_0 against four candidate obstacles. The non-visibility results are then merged to create the non-visibility boundary in the Undetermined region (Fig. 5-21b), denoted as N'.

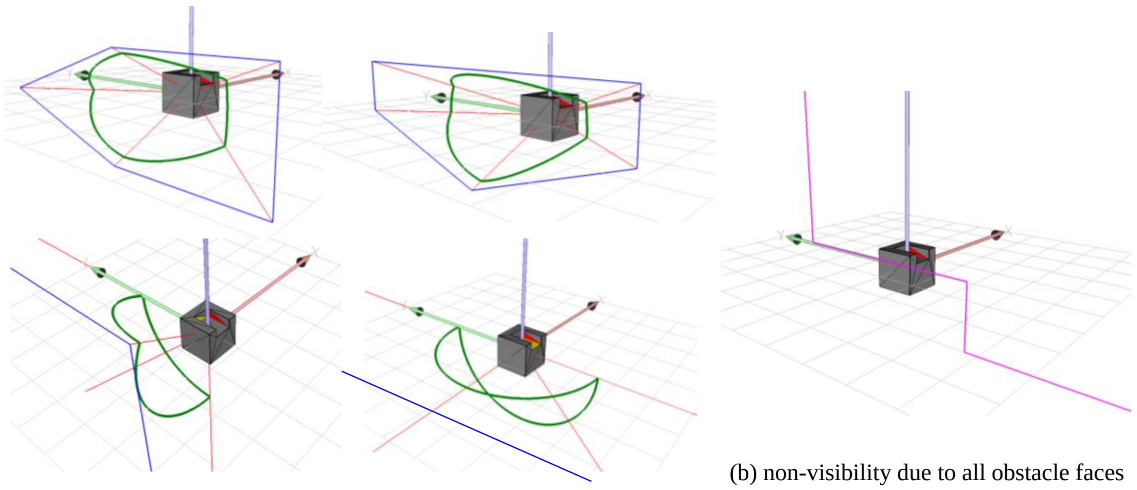

(a) non-visibility due to each obstacle face
(b) non-visibility due to all obstacle faces
(planar and spherical view)

Fig. 5-21 Non-visibility results. (a) Non-visibility results for each obstacle; Blue lines denote the results on a plane, green arcs denote the result on unit sphere (b) The union of non-visibility results for all obstacles.

Lastly, we combine this non-visibility result N' with the Non-visible region N obtained earlier to generate the final non-visibility polygon (Fig. 5-22a). We can easily obtain the visibility polygon by conducting a complement to the non-visibility polygon (Fig. 5-22b).

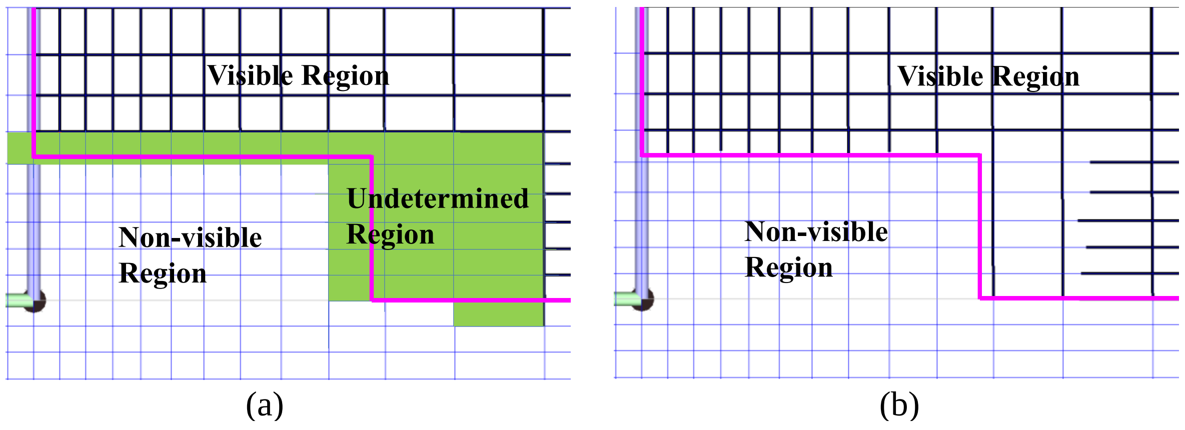


Fig. 5-22 Final visibility results; (a) Merging of Non-visibility results N' and Nonvisible region N. Purple boundary: N'. (b) Final visible and non-visible regions.

5.5 Conclusion

This paper has proposed and implemented a new hybrid visibility computation method that takes advantage of two independent visibility methods: an approximate method (V1) of time complexity $O(k \cdot n)$ and an exact method (V2) of time complexity $O(n^2)$ where k is the number of slices and n is the part mesh size. It reduces the processing time for exact computation at the cost of introducing approximate computation overhead. It also introduces an easy control over the approximate-exact computation ratio, making it possible to balance processing time and accuracy. The results have shown V1's output (computed once and available for all facets) has served as a good filter of obstacles for exact computation in V2.

However, the visibility computation in V1 and candidate obstacle determination has introduced time cost. If the time benefit gained from reducing obstacles for V2 does not offset the time cost on generating obstacle filter using V1, this method has no advantage in terms of computational efficiency. Also, we can only gain efficiency advantage on parts with large facets count, as can be seen from the time complexities of the two methods. Therefore, future work should be focused on investigating the impact of exact-approximate ratio and mesh granularity on the performance of the hybrid visibility method.

5.6 References

[1] Sheen, B.-T., and You, C.-F., 2006, "Machining feature recognition and tool-path generation for 3-axis CNC milling," Comput Aided Design, 38(6), pp. 553-562.

[2] Balasubramaniam, M., Sarma, S. E., and Marciniak, K., 2003, "Collision-free finishing toolpaths from visibility data," Comput Aided Design, 35(4), pp. 359-374.

[3] Woo, T. C., 1994, "Visibility Maps and Spherical Algorithms," Comput Aided Design, 26(1), pp. 6-16.

[4] Bi, Q.-Z., Wang, Y.-H., and Ding, H., 2010, "A GPU-based algorithm for generating collision-free and orientation-smooth five-axis finishing tool paths of a ball-end cutter," Int J Prod Res, 48(4), pp. 1105-1124.

[5] Suh, S. H., and Kang, J. K., 1995, "Process Planning for Multiaxis Nc Machining of Free Surfaces," Int J Prod Res, 33(10), pp. 2723-2738.

[6] Liu, M., Liu, Y. S., and Ramani, K., 2009, "Computing global visibility maps for regions on the boundaries of polyhedra using Minkowski sums," Comput Aided Design, 41(9), pp. 668-680.

[7] Li, Y., and Frank, M. C., 2007, "Computing non-visibility of convex polygonal facets on the surface of a polyhedral CAD model," Comput Aided Design, 39(9), pp. 732-744.

[8] Hou, G., and Frank, M. C., 2017, "Computing the Global Visibility Map Using Slice Geometry for Setup Planning," Journal of Manufacturing Science and Engineering.

[9] Frank, M. C., Wysk, R. A., and Joshi, S. B., 2006, "Determining setup orientations from the visibility of slice geometry for rapid computer numerically controlled machining," J Manuf Sci E-T Asme, 128(1), pp. 228-238.

[10] CGAL, 2019, "The Computational Geometry Algorithms Library,"https://www.cgal.org/, p. software library of computational geometry algorithms.

[11] Google, 2019, "S2Geometry,"http://s2geometry.io/, p. A Geographic information systems.

CHAPTER 6. CONCLUSION

This dissertation has introduced **three new computation methods** for visibility and accessibility:

1. An approximate slice geometry-based visibility method with controllable accuracy and resolution, and time complexity $O(n{\cdot}k)$; n: facets count; k: slices count.
2. A tool accessibility method for ball-end cutters based on visibility results and surface offsetting.
3. A hybrid visibility method with controllable exact-approximate computation ratio and innovative processing time reducing strategy.

This collection of contributions introduce new approximation strategies for visibility computation; explain relations between visibility and accessibility; and take advantage of both approximate and exact visibility computation. In the future, work could be done on solving the limitations of each methods, investigating the notion of partial visibility, accessibility of general type cutters, and the applications of accessibility, like the setup and toolpath optimization for multi-axis CNC machines.