International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:4, No:5, 2010

Visual Hull with Imprecise Input

Peng He

Abstract—Imprecision is a long-standing problem in CAD design and high accuracy image-based reconstruction applications. The visual hull which is the closed silhouette equivalent shape of the objects of interest is an important concept in image-based reconstruction. We extend the domain-theoretic framework, which is a robust and imprecision capturing geometric model, to analyze the imprecision in the output shape when the input vertices are given with imprecision. Under this framework, we show an efficient algorithm to generate the 2D partial visual hull which represents the exact information of the visual hull with only basic imprecision assumptions. We also show how the visual hull from polyhedra problem can be efficiently solved in the context of imprecise input.

Keywords—Geometric Domain, Computer Vision, Computational Geometry, Visual Hull, Image-Based reconstruction, Imprecise Input, CAD object

I. Introduction

IMAGE-BASED reconstruction investigates how to generate stereo objects from their projections. It has been broadly used in many inter-disciplinary applications, for example computer aided design (CAD), computed tomography (CT), robotic vision, face recognition, satellite imaging and so on. Among all the image-based reconstruction methods, whether it is a multi-camera system [25], [26], [10], [35] or camera-projector system [38], [33], [16], the precision of the results is not thoroughly discussed. In many research works, the reconstruction accuracy is shown either as visual comparison of pictures or as statistical data like standard deviation of multiple experimental results. However, in the application for industrial measurement, the accuracy of a single reconstruction is precisely required, i.e. we want to know which region is definitely inside or outside the objects of interest. Actually, under parallel projection, because of the limitation in the sensors, we can only know that the actual 3D point lies within a convex polyhedra which is given by a finite number of lower and upper bounds in a finite and fixed set of directions. The limited precision of computer data types cause similar inaccuracy in the result of reconstruction.

The real RAM machine model [31] which is the model used to prove the correctness of computational geometry algorithms, cannot handle the computer data structure which can only store a floating point number to a finite precision. Even though the basic arithmetic operators and analytic functions are Turing-computable [12], comparison between two real numbers is only semi-decidable [36]. This imprecision accumulates as a result of the combinatorial computations in most of classical computational geometric algorithms. Many research works have addressed this problem through different means, including the exact computation model [9], [6], [7], [17], [3], [18], [37], [1], ε-geometry [32], interval geometry [34], [22], [21] and so on.

The geometric (solid) domain [13], [14] is a robust framework which has been used to analyze many computational geometric problems with imprecise input, namely partial convex hull [15], partial Delaunay triangulation [11], [23], [24], [30] and partial Voronoi diagram [11]. All of the adapted algorithms prove to be computable operations in the context of recursion theory [8]. Therefore, it is a suitable framework for imprecision analysis in image-based reconstruction. Under this framework, each classical geometric object is obtained by a sequence of partial geometric objects [13] approximating it.

A visual hull is the closest approximation we can extract from the silhouettes (i.e. shape-from-silhouette) of a group of objects [27]. Many image-based algorithms for reconstruction are based on the visual hull of the objects. Classically, the visual hull is generated in an aspect graph manner [20], [19], [28], a key step of which is to build a partition of the space in which each region is numbered with its ways of visibility. Afterwards, the union of regions with visual number 0 is the visual hull. The 3D visual hull is more complicated than the 2D one since the boundary of regions with different visual numbers can be either planar or quadratic surface. There are external and internal visual hulls defined in [27] depending on whether the view point can enter the convex hull of the objects. However, when we mention the visual hull in this paper, we are referring to the former.

We aim at combining the advantages of geometric domain and image-based reconstruction to build a framework in which imprecision in the input can be exactly captured, preserved and reflected in the reconstructed results for demonstration or as the input for further analysis.

II. Partial Geometric Framework

Following the framework established in [13], each *partial geometric object* (A, B) consists of two disjoint open sets: A, the interior, and B, the exterior of the partial object, representing the sets of points which are definitely in the inner and outer regions of the object no matter how the input data is refined to a precise value.

A. 2D Elements

The basic element of our model is the partial point, which represents an imprecise point. As in [12], we fix say k approximating lines $l_1, l_2, \ldots, l_k$ through the origin, which give us $2k$ oriented directions $d_1, d_2, \ldots, d_{2k}$ labeled anti-clockwise. Two lines with outward normals d_i, d_{i+k} (i.e. perpendicular to the approximating line l_i) form an infinite strip S_i . The partial point is defined as

$$C = \bigcap_{i=1}^{k} S_i$$

P. He is with the Department of Computing, Imperial College London, 180 Queens Gate, SW7 2AZ, UK, e-mail: ph206@imperial.ac.uk.

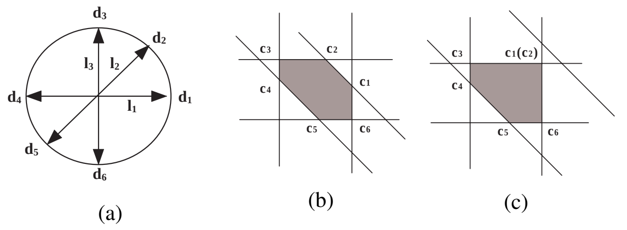


Fig. 1: A partial point determined by interval approximation on 3 fixed lines. (a) The 3 fixed lines (6 oriented directions) (b) A partial point (c) A partial point with coincident corner types

Figure 1 (a) and (b) give an example of a partial point with three approximating lines.

A nice property of this imprecise point model is that the intersection of two partial points is still a partial point. Note that a corner c_i of a partial point is the intersection point of two lines which give the boundaries of the partial point. These two lines have outward normals d_i and d_{i+1}. Thinking of a line moving along its normal d_a, which lies in between d_i and d_{i+1}, from infinity towards origin, c_i is the first point on the partial point it will touch. Therefore, we can say c_i is the furthest point along the direction d_a. However, there are cases that one or more directions are skipped over. As shown in Figure 1 (c), when the skipping-over happens, the closed corner is chosen as the corresponding corner type to maintain the equality of the corner type quantities and the furthest point property mentioned above. Therefore, a partial point is the best convex polygonal approximation of the imprecise point with the given approximation lines.

This model is also consistent with the imprecision caused by image sensors (cameras or projectors) and measuring devices. Under parallel projection, each approximating line actually corresponds to a projection plane. The infinite stripe is actually the best knowledge about the point from the corresponding projection. Figure 2 (a) gives an example of a partial point limited by projections on 3 planes. If the input data are coming from measuring devices, the imprecision is generally embedded in the coordinates. In this framework, such a point is modeled as a partial point with interval approximation along the coordinate axis, i.e. a rectangle (cuboid). For the simplicity of presentation, in the rest of the paper, we use the rectangular (cuboid) partial points. All the results can be easily generalized to objects formed by polygonal (polyhedral) partial points.

The *partial line segment (PLS)* of two partial points is the union of line segments with endpoints lying in the two partial points as in Figure 3 (a). A *selection* of a partial geometric object (or partial geometric object selection) is a classical geometric object with exactly one point in each partial points and connected in the same order as the partial geometric object. Figure 3 (a) shows a partial edge selection. To understand the shape of a partial line segment, we have the following two propositions.

Proposition 1. *Each PLS is the convex hull of its two partial points.*

In fact, each point in the partial point is a linear combination

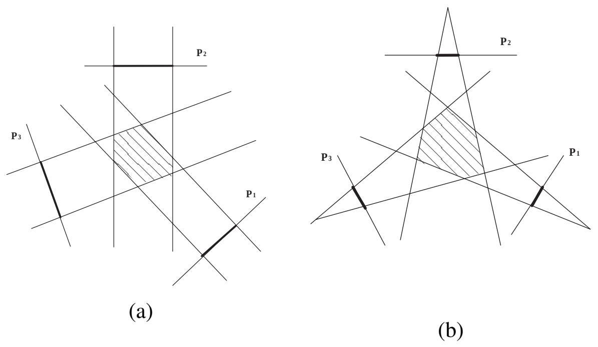


Fig. 2: The approximation of a point based on its projections on 3 planes P_1, P_2, P_3 under (a) parallel projection and (b) perspective projection.

of its corners. Each point on a line segment is also a linear combination of its two end points. Therefore, each point in the PLS is actually a linear combination of the corners of its two end partial points, which is the convex hull of the corners.

For each partial point, there are four corner types: top left, top right, bottom left, bottom right. A line or line segment passing through corners of the same type is defined as *same-type corner line* or *same-type corner line segment*. The *opposite-type corner line* or *opposite-type corner line segment* is a line or line segment passing through opposite corners of the two end partial points (e.g. top left from one and bottom right from the other). Given 2 partial point A and B, the *same-type half corner line* $\overrightarrow{AB}$ (respectively, *opposite-type half corner line* $\overrightarrow{AB}$) is a half-infinite line satisfying: (a) it is a part of a same-type corner line (respectively, opposite-type corner line) formed by A and B, (b) its original is a corner of B and (c) it does not coincide with any same-type corner line segment (respectively, opposite-type corner line segment) with respect to A and B.

Similarly, given two partial points, we can define the *partial line (PL)*, which is the union of lines passing through two points, one from each partial point. See Figure 3 (b). Also, a *partial half line* $\overrightarrow{AB}$ *(PHL[A, B])* (Figure 3 (c)) can be defined as the union of half lines with respect to partial points A and B satisfying: (a) it is a part of a line selection with respect to the PL formed by A and B, (b) its original lies in B and (c) it does not coincide with any partial line segment selection with respect to the PLS formed by A and B. Note that there are only two same-type corner lines or opposite-type corner lines that do not intersect with the given two partial points.

Proposition 2. *Given two partial points A and B, the PLS (respectively, PHL[A, B]) formed by them is bounded by two kinds of classical edges:*

1) *two same-type corner line segments (respectively, opposite-type half corner lines* $\overrightarrow{AB}$*) which are parts of two same-type corner lines (respectively, opposite-type corner lines) without any intersection with A or B;*
2) *the edges of partial points with their end points being either (i) the corner types whose opposite-type corner line segments do (respectively, don't) intersect with the two given partial points or (ii) end points of same-type*

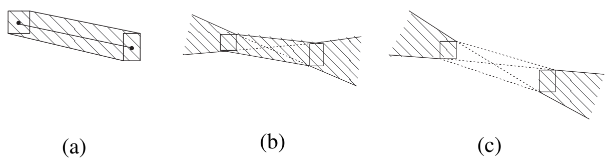


Fig. 3: (a) a partial line segment (PLS) (b) a partial line (PL) (c) two partial half lines (PHL)

corner line segments in 1.

We can easily see that for each point in a PLS there is a partial edge selection going through it. However, not every line segment within the PLS is a partial edge selection. The reason is that there are inner structure inside the PLS. If we only use its exterior information (PLS has empty interior), its inner structure is lost. Therefore, the PLS over simplifies the cases. When we want to use PLS as input, we need a data structure which stores its shape (interior and exterior) as well as its inner structure (end points separately lying in the two partial points). The same can be said for PL or PHL or any partial geometric object formed by partial points.

Proposition 3. *A PL is exactly bounded by:*

1) *two same-type corner line segments without any intersection with the two given partial points;*
2) *four opposite-type half corner lines, which are parts of two opposite-type corner lines without any intersection with the given two partial points;*

This proposition basically says that we do not need to consider the boundary lines of the partial points while generating a PL. If E, an edge of a partial point is part of a boundary line segment of the PL, then we can select a line passing through a point P in the interior of E (i.e. P is not an end point) and any point Q in the other partial point to generate a partial line selection $\overset{\frown}{PQ}$. It intersects E at P, which therefore contradicts with E being a boundary line of the PL.

B. Partial Polygon

Following the definitions above, a *partial polygon (PPG)* that is defined as the union of all possible polygons which may generate the same input data in the given reconstruction configuration environment.

We assume now that a PPG has the following properties: (i) Its partial vertices are pairwise non-overlapping. (ii) Its partial edges are given as a directed cycle. (iii) It is simple i.e., any polygon selection with respect to it is a simple polygon.

See Figure 4 (a) for an example of partial polygon and polygon selection. Note that the shaded area labeled with I is its interior and the area outside the dashed-line labeled with E is its exterior. In the practice, assumption (i) can be eliminated by a pre-process, in which if two partial points intersect, they will be replaced by their intersection.

A *corner selection* is a polygon bounded by any combination of same-type corner line segments, one from each partial edge, and boundary lines of partial points to generate an enclosed region. See Figure 4 (b) for an example of corner selection.

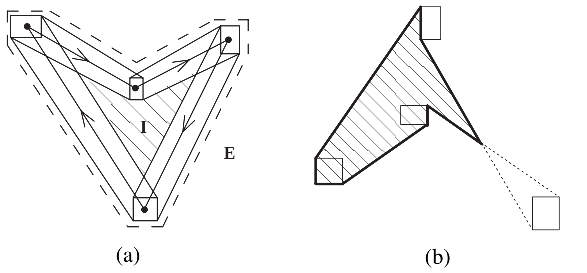


Fig. 4: (a) a partial polygon and a partial polygon selection (PPG) (b) a corner selection

Lemma 4. *The interior (exterior) of a PPG (PPG_I, PPG_E) is the intersection of inner (outer) areas of all corner selections. i.e.*

$$PPG_I = (\bigcap\{P|P \textit{ is a conner selection}\})^\circ$$

$$PPG_E = \bigcap\{P^c|P \textit{ is a conner selection}\}$$

where P^c and P° are respectively the complement and the interior of a geometric object P.

As it is shown in [13], if the polygon is guaranteed convex, i.e. each polygon selection is convex, then the PPG is equal to its partial convex hull.

$$PPG_I = \bigcap_{j=1}^{2n} \Gamma(\{c_{ij}|1 \le i \le N\}))^\circ$$

$$PPG_E = (\Gamma(\{c|c \textit{ is a corner of } C_i,\ 1 \le i \le N\}))^c$$

where $\Gamma(C)$ is the convex hull of a point list C and c_{ij} is the jth corner of the ith partial point C_i.

In the context of PPG, a partial edge is a PLS formed by two partial vertices. Therefore, a partial edge contains all the possible edge selections. Since any point on an edge is an indeterminant point (i.e. we cannot decide whether it is inside or outside the PPG). Therefore, the partial edges of a PPG contain all the points that are indeterminant. The remaining points are either in the interior or exterior of the PPG. Therefore, the interior and exterior are bounded by the boundary lines of partial edges which are either edges of the partial vertices or same-type corner line segments. Using the Lemma 4, we have the following theorem.

Theorem 5. *The interior of a PPG is the topological interior of the corner selection with the smallest area, and the exterior of the PPG is the complement of the corner selection with the biggest area.*

Theorem 5 tells us that in order to find the interior and exterior of a PPG, it suffices to identify the boundary lines of its partial edges. After generating partial edges and labeling their boundary same-type corner line segments. we have the PPG generation algorithm:

```
Algorithm 1 Interior and Exterior of a PPG
Require: Two boundary line lists of interior and exterior.
    Each list contains $n$ edges in the format of $\overline{p_{ik_i}p_{i+1k_i}}$, which
    means an edge connecting the $k_i$th corners of partial points
    $i$ and $i+1$.
Ensure: Two vertex lists for polygon approximations of interior and exterior of the PPG
  for $i$ = 1 to n do
    Pick $\overline{p_{ik_i}p_{i+1k_i}}$ from the interior list
    if $p_{i+1k_i}$ is the same as $p_{i+1k_{i+1}}$ then
      Add $p_{i+1k_i}$ into the vertex list
    else if $\overline{p_{ik_i}p_{i+1k_i}}$ intersects with $\overline{p_{i+1k_{i+1}}p_{i+2k_{i+1}}}$ then
      Add the intersection point into the vertex list
    else
      Add all vertices between $p_{i+1k_i}$ and $p_{i+1k_{i+1}}$ (the order
      of adding vertices reverses to the order of partial edges)
      on partial point $P_{i+1}$ into the vertex list
    end if
  end for
  The vertex list contains the vertices of the polygon approximation of the interior in the same order as the partial
  polygon
  {The exterior can be generated in a similar way}
```

Since the boundary lines of a partial edge can be identified in $O(1)$ time, for a simple polygon (no self-intersection), we have the PPG generation function with the same time complexity as the classical method.

C. 3D Extension

The 3D case is far more complex. In 2D, we only deal with points and lines in the plane, but in 3D we need to work on points and lines in space, planes and even quadratic surfaces. In this section, we present the 3D domain-theoretic framework of computational geometry which is an extension of the 2D framework. As in the 2D case, we assume the topology of the object is known.

The 3D partial points can be considered as polyhedra. In parallel projection, each pixel in the projection image is a small square. Therefore, a partial point is now the intersection of a finite number of infinite rectangular tubes. A more formal definition is obtained by fixing k approximating lines $l_1, l_2, \ldots, l_k$ through the origin, which give us $2k$ oriented directions $d_1, d_2, \ldots, d_{2k}$. Two planes with outward normals d_i, d_{i+k} (i.e. perpendicular to approximating line l_i) form an infinite thick plane T_i. The partial point C is define as

$$C = \bigcap_{i=1}^{k} T_i$$

Actually, each corner of a partial point is the intersection point of three planes. Their outward normals have three intersection points with a unit sphere (Figure 5 (a)). Think of a plane moving along its normal d_a, which lies within the spherical triangle [2] formed by those three intersection points, from infinity towards origin. c_i is the first point on the partial point it will touch, and therefore the furthest point along the direction d_a. Please refer to Figure 5 (b) for an illustration.

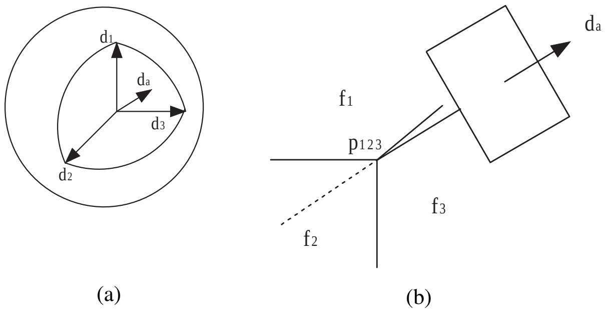


Fig. 5: (a) a unite sphere and 3 plane normals d_1, d_2 and d_3; (b) p_{123} being the furthest point along d_a;

Since each edge of a partial point is the intersection of two planes with outward normals d_i and d_j, the edges of the same type on different partial points are parallel, therefore coplanar. Similarly, the edges of the opposite types (intersection of planes with opposite normals) are also parallel and coplanar. Therefore, we can have the 3D same-type corner line (or same-type corner line segment or same-type half corner line) and 3D opposite-type corner line (or opposite-type corner line segment or opposite-type half corner line) with the same definition as their 2D counterparts but in the 3D context. We also have *partial same-type edge line* (or *partial same-type edge line segment*) or *partial opposite-type edge line* (or *partial opposite-type edge line segment*) which are respectively the union of lines (or line segments) have intersections with two edges of the same or opposite types of the two end partial points. Given 2 partial points A and B, the *partial same-type half edge line* $\overrightarrow{AB}$ (respectively, *partial opposite-type half edge line* $\overrightarrow{AB}$) is the union of half-infinite lines satisfying: (a) it is a part of a line which has intersection with two same (respectively, opposite) type edges of A and B, (b) its original is on B and (c) it does not coincide with any partial same-type (respectively, opposite-type) edge line segment with respect to A and B.

Actually, on the plane which passes through the two edges of the same type, if we think these edges as degenerated 2D partial points, the partial same-type edge line (or partial same-type edge line segment or partial same-type half edge line $\overrightarrow{AB}$) is actually the 2D PL (or PLS or PHL[A,B]) formed by them. The same can be said for partial opposite-type edge line (or partial opposite-type edge line segment or partial opposite-type half edge line $\overrightarrow{AB}$). See Figure 6 (a).

3D PSL (or PL or PHL[A,B]) has the same definition with the 2D one but in 3D context. Following the proposition 1, 2 and 9 we have the following corollaries.

Corollary 6. *Each PLS is the convex hull of its two end partial points.*

Corollary 7. *Given 2 partial points A and B, let N be the number of partial same-type edge lines without any intersection with the partial points. Then the PLS (respectively, PHL[A,B]) is bounded by two kinds of classical surfaces:*

(i) *N partial same-type edge line segments (respectively, partial opposite-type half edge line $\overrightarrow{AB}$) which are parts*

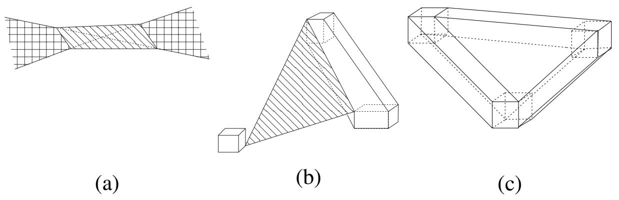


Fig. 6: (a) the planar view of an partial same-type edge line segment (forward-slash shading), two partial same-type half edge lines (grid shading) and a partial same-type edge line (both) (b) a reversed PV-PE face (c) a partial face (PF)

of N partial same-type edge lines (respectively, partial opposite-type edge lines) without any intersection with A or B;

(ii) *the faces of partial points with their edges being either (i) the edge types whose partial opposite-type edge line segments do (respectively, don't) intersect with the two given partial points or (ii) edges of partial same-type edge line segments (respectively, partial opposite-type half edge lines) in 7;*

Corollary 8. *Given 2 partial point A and B, let N be the number of partial same-type edge lines without any intersection with the partial points. Then the PL is exactly bounded by:*

(i) *N partial same-type edge line segments which are parts of N partial same-type edge lines without any intersection with A or B;*

(ii) 2*N partial opposite-type half edge lines which are parts of N partial opposite-type edge lines without any intersection with A or B;*

We only consider 3D objects given as triangular meshes since a polygonal mesh can always be further divided into a triangular mesh, although different triangulations may result in different partial triangular meshes. The *same-type corner face* is defined as a face connecting same-type corners of the partial vertices. Also, a *cross PV-PE face* is a face passing through: (1) the corner of a type on one partial vertex and (2) a same-type corner line segment connecting corners of the opposite type on the other two partial vertices which form a partial edge. See Figure 6 (b). A *partial face* (PF) is the union of all possible faces, each of which passes through exactly one point in each partial vertex. Also, a partial face is also the convex hull of its three partial vertices. Figure 6 (c) gives an example. Actually, as an extension to theorem 2, we have the following corollary

Corollary 9. *Each PF is bounded by three kinds of surfaces:*

- *two same-type corner faces that have not intersection with any of its three partial vertices;*
- *partial same-type edge line segments whose two boundary same-type corner lines are either (1) whose cross PV-PE face has intersections with the given three partial points or (2) boundary lines of same-type corner faces in 9;*
- *faces of a partial point which are boundary faces of all the partial edges it forms;*

Similar to PLS, for each point in a PF there is a PF selection passing through it. Also a *corner selection* is a combination of same-type corner faces (one from each PF), necessary partial same-type edge line segments and faces of partial points to make it an enclosed polyhedron. A *partial polyhedron* (PPH) is actually the union of all possible polyhedra with a given list of partial points and the topology of the polyhedron. We assume the PPH satisfies the same assumptions of PPG in the 3D context.

Corollary 10. *The interior of a PPH is the topological interior of its corner selection with the smallest volume, and the exterior of the PPG is the complement of its corner selection with the biggest volume.*

A PF with three partial points can be generated in constant time. Therefore, the generation of a simple PPH (no self-intersection), which is an analogy to the PPG generation algorithm, is of the similar time complexity with the classical partial polyhedron generation method.

III. VISUAL HULL WITH IMPRECISE INPUTS

The visual hull from polygons or polyhedra is a classical algorithm for generating the maximum silhouette equivalent structure from a set of objects. If the known shape is coming with imprecision, our framework can cover it while the classical algorithm becoming non-robust.

A. 2D Visual Hull with Imprecise Inputs

A *partial visual hull* of PPGs is a partial geometric object whose interior (exterior) is the set of points which are definitely inside (outside) the visual hull with respect to any polygon selection.

We will generate the partial visual hull of the objects of interest when they are given as PPGs. In the case that shapes are given as a set of imprecise vertices and their relationships (topology), the visual number for a view point may not be unique, i.e. regions of different visual numbers may overlap with each other. Like the classical visual hull algorithm [27], To have the partial visual hull, we need to have the *partial visual number partition* (partial partition for short) of the space allowing overlapping. Then we can extract the interior and exterior of the partial visual hull from it. In this partition, any region of the partial partition can have several possible visual numbers.

For simplicity, we assume no three partial points are collinear, i.e., no straight line intersects all three partial points. A large number of collinear cases indicate that the data are not refined enough with respect to the scale of the objects. This problem can be tackled by either refining the input data until there is no collinearity, or isolating the collinear cases and solving them by brute-force algorithms.

In order to construct the partial partition of $\mathbb{R}^2$, we need to consider the three cases of partial active segments as shown in Figure 7. They are actually PHLs and PLSs with arrows on their boundary lines. A *partial active segment (PAS)* is the union of all possible active segments [27]. The arrows show the direction from a region with lower possible visual numbers to a region with higher possible visual numbers. These arrows

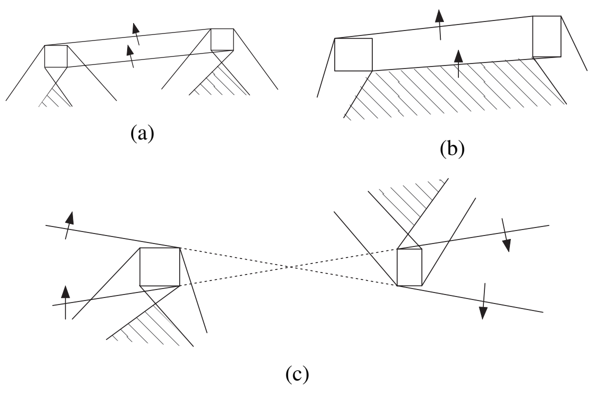


Fig. 7: Three situations of the partial active segments (PASs) the arrow shows the ascending direction of the visual numbers, i.e. the visual number increases when crossing over the boundary following the arrow direction

determine the *entering* and the *exiting* edges of a PAS as indicated in Figure 7.

Proposition 11. *Given a group of PPGs, the PASs contain all the points with multiple visual numbers, i.e. the PAS is the overlapping area of regions with different visual numbers.*

Therefore, after generating all PASs, the plane $\mathbb{R}^2$ is partitioned into two types of regions PASs and regions limited by boundary lines of PASs (these regions can be bounded or half-bounded). This is a generalized partition of the space in the sense that PASs may overlap with each other. We need to find the set of visual numbers of each region, i.e., its possible visual numbers for all possible polygon selections with respect to the PPGs.

Note that each region has at least one same-type corner line from some PAS as its boundary line.

Theorem 12. *The set of possible visual numbers of a region is a set of consecutive non-negative integers. If a PAS A borders a region B on its entering or exiting edge, A has one more visual number than B which respectively equals to the maximum or minimum visual number of A plus or minus one.*

Therefore, we have the rules of labeling as follows:

1) Start from an interior of a PPG and label it as $\{0\}$.
2) If the current region is labeled as $\{n, n+1, \ldots, m\}$, expand by labeling all regions which borders the current labeled region on a same-type corner line as follows:
   (i) If expanding in a PAS following the arrow direction, label the new region as $\{n, n+1, \ldots, m+1\}$.
   (ii) If expanding out a PAS following the arrow direction, label the new region as $\{n+1, n+2, \ldots, m\}$.
   (iii) If expanding in a PAS against the arrow direction, label the new region as $\{n-1, n, \ldots, m\}$.
   (iv) If expanding out a PAS against the arrow direction, label the new region as $\{n, n+1, \ldots, m-1\}$.

By repeating step 2, we will label all regions of the partition. See Figure 8 as an example of a partial partition of $\mathbb{R}^2$.

Finally, we merge all regions labeled with $\{0\}$ as the interior

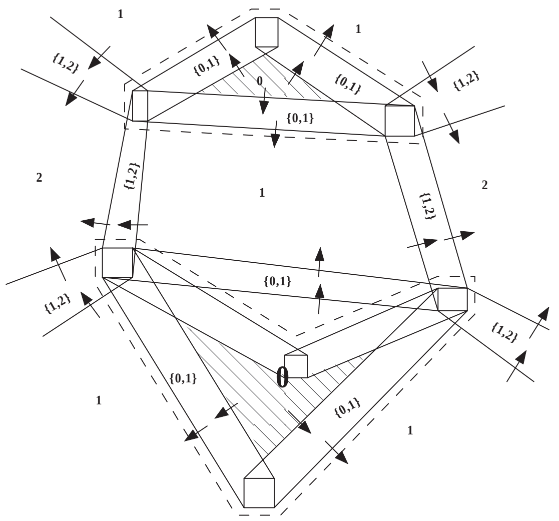


Fig. 8: A partial visual number partition of $\mathbb{R}^2$

and all regions with labels not containing 0 as the exterior of the partial visual hull.

B. 3D Visual Hull with Imprecise Inputs

In the classical algorithm, the 3D visual hull is generated with a brute force algorithm. The visual number of 3D case in [27] is not a natural extension of the 2D visual number. It is defined with respect to the topology of the shapes. However, in our framework, the topologies of the interior and exterior of a PPG may vary greatly which will cause inconsistency in the visual numbers. But the areas with visual number 0 are not affected so the interior and exterior of the visual hull can still be determined. Despite the pruning algorithms which can improve the efficiency of the computing, here are two main steps in the classical method:

(i) Identify the potential active surfaces, i.e. potential boundaries of regions with different visual numbers.
(ii) Pick any point in each cell and calculate its visual number. It is therefore the visual number of the whole cell.

And for item (i), there are two cases. VE surface which is a plane and EEE surface which is a quadratic surface.

We have 2 conjectures for each step respectively

Conjecture 13. *The partial VE and EEE surface are bounded by classical surfaces generated by the corners and boundary lines of the objects' partial point and partial edges.*

Conjecture 14. *For any view point, the zeroness of its partial visual number with respects to a given list of polyhedra, can be calculated by only considering the corner selections of the PPHs.*

If both conjectures are true, the complexity of the algorithm for the partial case is only multiples of the classical one which means that the time complexity remains the same.

IV. Future Works

In the future, following problems should be addressed to complete the framework and provide a thorough answer to the visual hull problem

- Currently we only consider cases under parallel projection. However, with perspective projection (Figure 2 (b)), properties like parallelism and coplanarity may not exist and problem can get more complicated. In the 3D case, the perspective partial point is the intersection of a number of infinite square pyramids.
- The partial visual hull of Solids of Revolution [29], smooth curved objects [4] or even piecewise smooth objects [5] are more complicated than the polygonal cases and will be studied in depth.
- The 3D partial visual hull problem should be fulled solved by either proving our conjectures.
- Study the cases where topology of the shape is unknown. Existing partial Delaunay triangulation algorithm [11] can be used to build the triangular mesh from a list of reconstructed partial points. This is quite useful if we want to analyze the precision of some 3D reconstruction configuration.

V. Conclusion

In this paper, we first present the geometric framework including the basic 2D partial geometric objects (Partial point, PL, PLS, PHL etc.) and their properties, some of which are established in [13], [14]. Then we discuss how to establish more complicated objects like partial polygons. Nevertheless, we extend the 2D framework to 3D and show how 3D basic geometric objects can be defined and modeled. The definition, properties and generating methods of the partial polyhedron are discussed.

To demonstrate the imprecision analysis based on this framework, we adapt the classical 2D visual hull generation methods to robustly process imprecise input data. The visual hull from partial polyhedra is discuss and 2 conjectures are provided. The proof of them will be studied in the near future.

Acknowledgment

The author would like to thank his supervisor professor Abbas Edalat for his invaluable support and inspiring comments, which improve the readability and quality of this paper.

References

[1] Leda. http://www.mpi-sb.mpg.de/LEDA/leda.html.

[2] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York, 9th printing edition, 1972.

[3] F. Avnaim, J.-D. Boissonnat, O. Devillers, F. P. Preparata, and M. Yvinec. Evaluation of a new method to compute signs of determinants. In *SCG '95: Proceedings of the eleventh annual symposium on Computational geometry*, pages 416–417, New York, NY, USA, 1995. ACM.

[4] Andrea Bottino and Aldo Laurentini. The visual hull of smooth curved objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(12):1622–1632, 2004.

[5] Andrea Bottino and Aldo Laurentini. The visual hull of piecewise smooth objects. *Comput. Vis. Image Underst.*, 110(1):7–18, 2008.

[6] Hervé Brönnimann, Ioannis Z. Emiris, Victor Y. Pan, and Sylvain Pion. Computing exact geometric predicates using modular arithmetic with single precision. In *SCG '97: Proceedings of the thirteenth annual symposium on Computational geometry*, pages 174–182, New York, NY, USA, 1997. ACM.

[7] Hervé Brönnimann and Mariette Yvinec. Efficient exact evaluation of signs of determinants. In *SCG '97: Proceedings of the thirteenth annual symposium on Computational geometry*, pages 166–173, New York, NY, USA, 1997. ACM.

[8] N. J. Cutland. *Computability: An Introduction to Recursive Function Theory*. Cambridge University Press, 1980.

[9] Olivier Devillers, Alexandra Fronville, Bernard Mourrain, and Monique Teillaud. Algebraic methods and arithmetic filtering for exact predicates on circle arcs. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, pages 139–147, New York, NY, USA, 2000. ACM.

[10] C. R. Dyer. Volumetric scene reconstruction from multiple views. In *Foundations of Image Understanding*, pages 469–489. Kluwer, 2001.

[11] A. Edalat, A. A. Khanban, and A. Lieutier. Delaunay triangulation and Voronoi diagram with imprecise input data. *Electronic Notes in Theoretical Computer Science*, 66(1), July 2002. Proceedings of the 5th CCA Workshop.

[12] A. Edalat, A. A. Khanban, and A. Lieutier. Computability in computational geometry. *New Computational Paradigms*, 3526:117–127, 2005.

[13] A. Edalat and A. Lieutier. Foundation of a computable solid modeling. In *Proceedings of the fifth symposium on Solid modeling and applications*, ACM Symposium on Solid Modeling and Applications, pages 278–284, 1999.

[14] A. Edalat and A. Lieutier. Foundation of a computable solid modelling. *Theoretical Computer Science*, 284(2):319–345, June 2002.

[15] A. Edalat, A. Lieutier, and E. Kashefi. The convex hull in a new model of computation. In *Proceedings of the 13th Canadian Conference on Computational Geometry*, pages 93–96, University of Waterloo, August 2001.

[16] Philipp Fechteler and Peter Eisert. Adaptive Colour Classification for Structured Light Systems. *IET Journal on Computer Vision*, 3(2):49–59, June 2009. Special Issue on 3D Face Processing.

[17] Steven Fortune. Polyhedral modelling with multiprecision integer arithmetic. *Computer-Aided Design*, 29(2):123–133, 1997.

[18] Steven Fortune and Christopher J. Van Wyk. Efficient exact arithmetic for computational geometry. In *SCG '93: Proceedings of the ninth annual symposium on Computational geometry*, pages 163–172, New York, NY, USA, 1993. ACM.

[19] Ziv Gigus, John Canny, and Raimund Seidel. Efficiently computing and representing aspect graphs of polyhedral objects. *IEEE Trans. on Pat. Matching & Mach. Intelligence*, 13(6), June 1991.

[20] Ziv Gigus, John F. Canny, and Raimund Seidel. Efficiently computing and representing aspect graphs of polyhedral objects. Technical Report UCB/CSD-88-432, EECS Department, University of California, Berkeley, Aug 1988.

[21] Chun-Yi Hu, Takashi Maekawa, Evan C. Sherbrooke, and Nicholas M. Patrikalakis. Robust interval algorithm for curve intersections. *Computer-Aided Design*, 28(6-7):495–506, 1996.

[22] Chun-Yi Hu, Nicholas M. Patrikalakis, and Xiuzi Ye. Robust interval solid modelling part ii: boundary evaluation. *Computer-Aided Design*, 28(10):819–830, 1996.

[23] A. A. Khanban and A. Edalat. Computing Delaunay triangulation with imprecise input data. In *Proceedings of the 15th Canadian Computational Geometry Conference*, pages 94–97, 2003.

[24] M. Kreveld, M. Loffler, and J. S. Mitchell. Preprocessing imprecise points and splitting triangulations. In *ISAAC '08: Proceedings of the 19th International Symposium on Algorithms and Computation*, pages 544–555, Berlin, Heidelberg, 2008. Springer-Verlag.

[25] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *Computer Vision, IEEE International Conference on*, 1:307–314, 1999.

[26] Kiriakos N. Kutulakos. Approximate N-view stereo. In *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part I*, pages 67–83, London, UK, 2000. Springer-Verlag.

[27] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, February 1994.

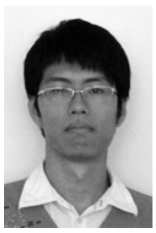
[28] Aldo Laurentini. Introducing the reduced aspect graph. *Pattern Recogn. Lett.*, 16(1):43–48, 1995.

[29] Aldo Laurentini. Computing the visual hull of solids of revolution. *Pattern Recognition*, 32(3):377–388, 1999.

[30] M. Loffler and J. Snoeyink. Delaunay triangulations of imprecise pointsin linear time after preprocessing. In *SCG '08: Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 298–304, New York, NY, USA, 2008. ACM.

[31] Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction (Monographs in Computer Science)*. Springer, August 1985.

[32] D. Salesin, J Stolfi, and L. Guibas. Epsilon geometry: building robust algorithms from imprecise computations. In *SCG '89: Proceedings of the fifth annual symposium on Computational geometry*, pages 208–217, New York, NY, USA, 1989. ACM.
[33] Daniel Scharstein. High-accuracy stereo depth maps using structured light. pages 195–202, 2003.
[34] Thomas W. Sederberg and Rida T. Farouki. Approximation by interval bezier curves. *IEEE Comput. Graph. Appl.*, 12(5):87–95, 1992.
[35] G. G. Slabaugh, T. Malzbender, W. B. Culbertson, and R. W. Schafer. Improved voxel coloring via volumetric optimization. Technical report, 2003. Center for Signal and Image Processing, Georgia Institute of Technology.
[36] Klaus Weihrauch. *Computability*. Springer-Verlag New York, Inc., New York, NY, USA, 1987.
[37] Chee Yap and Thomas Dub. The exact computation paradigm. 1994.
[38] Li Zhang, Brian Cudess, and Steven M. Seitz. Rapid shape acquisition using color structured lightand multi-pass dynamic programming. In *In The 1st IEEE International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 24–36, 2002.



Peng He was born in Beijing, China on 29th July, 1984. He received his bachelor degree in Hong Kong University in 2006 and his master degree in Oxford University in 2007. In 2008, he start a doctoral study in Imperial College London.

His research interests lie in geometric domain, computer vision and computational geometry.