An Object Space Approach for Outermost Silhouettes Extraction

Workshop on Multimedia Technologies

Ren-Long Lee and Wen-Kai Tai*

Computer Science and Information Engineering

National Dong Hwa University, Taiwan

1, Sec. 2, Da Hsueh Rd., Shou-Feng, Hualien, Taiwan, Republic of China

E-mail:wkdai@mail.ndhu.edu.tw

Abstract

Conspicuous outermost silhouette edges reveal the peculiar portrayal of model, and it is useful for applications, such as motion blur, morphing, simplification, skeletonization, etc. We propose a practical and effective approach to approximately extract the outermost silhouette edges from polygonal models under arbitrarily perspective views. All facet-normals and edges are encapsulated into a hierarchical tree in the preprocessing phase to facilitate the extraction of outermost silhouette edges. A two-stage extraction procedure is used at run-time. The first stage prunes edges, hardly to be recognized as the outermost silhouette edges using the inclusion relationship. The remaining edges are then used to approximate the outermost silhouette edges with respect to the intersection tests in the second stage. Experimental results show that our approach is effective and sufficient to most applications.

Keywords: silhouettes, outermost silhouettes, motion blur

1. Introduction

Silhouette is salient perceptible cues that humans use to estimate and recognize objects. For years, a number of graphics and visualization applications have been utilized them to assist further processing, such as model simplification [4, 5, 16], image-based rendering [14], non-photorealistic rendering [2, 8, 10], calibration [13], model registration [11], shadow computation [11], skeletonization [17], motion blur [2], etc.

Much attention has been devoted to silhouette edges extraction [1, 3, 7, 12, 15]. Benichou and Elber [1] deal with this problem under orthogonal projection. The authors mapped the normals of all polygons onto the Gaussian sphere as associated arcs with polygon edges and furthermore projected the sphere onto a unit cube. Then, the silhouette edges extraction problem has been reduced into finding the intersection of line segment on the plane. This approach only works on orthographic view; however, most applications require the use of perspective view.

Buchanan and Sousa [2] propose a fairly compact data structure, edge buffer, for silhouettes rendering. The edge buffer stores flags of vertex, front-facing, and back-facing. Initially, the flags of vertex are established according to the dependence of vertex with edges and the flags of front-facing and back-facing, FB flags, are set to zero. At each frame, the FB flags for the edges are updated, XOR a 1 value, depending on whether the polygon is front-facing or back-facing. The rendering of silhouette edges can be done by checking the FB flags. Although the edge buffer is easy to implement, explicitly examining all edges can not be avoided.

Barequet et al. [12] solve this problem based on a point-plane duality in three dimensions. Their approach applied dual transform to a viewpoint and edges. In this case the silhouette edge occurs when the point dual of edge intersects the plane dual of viewpoint. Finding the silhouettes edges is then equivalent to finding which segments are intersected by a query plane. For reducing silhouette edges computations, they stress the silhouettes update between consecutive frames. That is, segments cross the wedge plane defined by the two positions of the viewplane, and silhouette edges occur when exactly one of whose endpoints is in the wedge. Thus their algorithm obtains silhouettes from a sequence of view points under perspective projection. Obviously, answering the query for the first time requires a brute-force computation of all the edges on the silhouette.

Johnson and Cohen [7] develop a data structure, the spatialized normal cone hierarchy and apply it for model silhouette edges extraction, local minimum distance computations, and area light source shadow boundary determination. An edge is said to be a silhouette edge if the span of the two neighboring facet-normals over an edge contains a vector orthogonal to the view vector. Hereby, the silhouette edge can be determined by computing the angle between the normal cone and view cone axis. This data structure emphasize that it interactively solve seemingly disparate problem, so the silhouettes extraction is not quite fast.

Sander et al. [15] extract the silhouette edges from an anchored cone of normals in a search tree. Anchored cones provide conservative bounds on the front-facing and back-facing regions of a set of faces. By locating a viewpoint, they can cull the faces involved in the node which are completely front-facing or back-facing. The extracted silhouette edges are further formed exterior silhouette in the stencil buffer and then used for clipping coarse mesh which is larger than the original one. The exterior silhouette is similar to outermost silhouettes, but in the image space. Hence, they have no information about which silhouette edges are exterior.

Silhouette edges are attractive features, but the outermost silhouette edges are even more representative than it in sketching models. The appearance of the outermost silhouette edges of an object is one of the strongest visual cues regarding the shape and the main feature of the model. For a given light source, the shadow of a given model is formed with darkened area where polygons sheltered from streams of light. Between the obscure and bright area, that will form an outline of model corresponding to the light source. Turning the light source into a view point, the outermost silhouette edges consist of projected silhouette edges contributing to the outline of model. Although many studies have been published concerning all silhouette edges extraction, little information is available on delineating a given model using the least amount of silhouette edges while retaining the features of the model.

In this paper, we propose an approach to extract outermost silhouette edges for a given model. In the preprocessing phase, given a polygonal mode, we construct an OBBTree of the model and record necessary information for each node. In the run-time phase, for a given viewpoint and some constrained parameters, we determine candidate silhouette edges from the OBBTree according to the inclusion relationship, and then extract outermost silhouette edges from these candidate silhouette edges with respect to the inclusion test efficiently and effectively.

In this paper, we have a number of contributions as follows:

- Propose an idea of extracting outermost silhouette edges, a sharp feature of polygonal models under arbitrary perspective views.
- Propose an approach for outermost silhouette edges extraction based on the pruning strategy first and then incrementally approximating outermost silhouette edges.
- A demonstration that the outermost silhouette edges produce glamorous motion blur of a given model in an interactive environment.

The organization of this paper is as follows. In section 2, our approach is specified. In section 3, we demonstrate experimental results that show the effectiveness of the proposed approach. Furthermore, one practical application is exhibited. Finally, the conclusion of this work and some possible improvements in the future are stated in section 4.

2. Approach

Our approach consists of two phase: a preprocessing phase and run-time interaction phase. In the preprocessing phase, we construct an OBBTree and compute parameters stored in each node. In the run-time interaction phase, we set proper conditions and use OBBTree to facilitate the outermost silhouettes extraction. Notice that our work is based on the assumption that all facet-normals face outward from the model, every edge is bordered by exactly two faces, and breaks in the mesh are not allowed.

2.1 OBBTree

The OBBTree was introduced by S. Gottschalk et al. [6], and used for efficient and exact interference detection among complex models undergoing rigid motion. An oriented bounding box, OBB for short, is a box whose faces have normals or axes that are all mutually orthogonal. For a given polygon soup, the OBBTree is constructed from top to down by fitting a box to a group of triangles and partitioning them into two subgroups. Applying the process recursively creates a tree with leaf nodes containing a triangle. The boxes will tend to align with the geometry of a flat surface patch. Furthermore, the interference of two OBBs is determined by 15 potential separating axes test. Although the original purpose of the OBBTree was to assist collision detection, we will show that it was also useful for outermost silhouettes extraction.

2.2 Run-time

Intuitively, the outermost silhouettes extraction can be achieved by finding all silhouette edges and eliminating inner silhouette edges which are completely covered by other objects in a projected plane, but this method has some drawbacks. Too many silhouette edges need to be examined; moreover, the cost of determining whether a silhouette edge is an outermost silhouette is high. Instead, our approach is to reduce silhouette edges, and we decrease the cost of outermost silhouettes extraction. The run-time algorithm consists of two stages: a candidate silhouettes extraction stage and an outermost silhouettes extraction stage.

2.2.1 Polygon Inclusion

Before we discuss the actual approach, we first define an ***inclusion*** relationship between polygons of a sub-mesh. Consider a two-manifold polygonal object in which each polygon accompanies some adjacent polygons. For a given polygon and view point, V , the inclusion occurs when the polygon and the adjacencies are all front-facing or back-facing. More formally, let a geometric polygon P with normal $\bar{N}$ and the adjacent polygons,

$P_1,\ P_2,\ P_3,\ldots,\ P_n$, with facet-normals, $\overrightarrow{N_1},\ \overrightarrow{N_2},\ \overrightarrow{N_3},\ldots,\ \overrightarrow{N_n}$, respectively. A polygon P is inclusion if the following holds.

$$\forall(\vec{N}\bullet\vec{V})(\overrightarrow{N_i}\bullet\vec{V})<0 \quad for\ 1\le i\le n,\quad or$$
$$\forall(\vec{N}\bullet\vec{V})(\overrightarrow{N_i}\bullet\vec{V})>0 \quad for\ 1\le i\le n$$

where $\vec{V}$ the vector from any vertex lying on the polygon to the viewpoint.

The major strategy of candidate silhouettes extraction is to reduce silhouette edges, while containing outermost silhouettes. When we apply the inclusion test to all polygons in an exhausted manner, we can find all silhouette edges. This result contains the outermost silhouettes, at the expense of high computation cost, especially for models with too many silhouette edges. Our idea was from the polygon inclusion relationship which could preserve the outermost silhouettes, and pruning interior polygons. Furthermore, we want to reduce inner silhouette edges as much as possible; hence, we extend the inclusion relationship to a polygon cluster or bounding volume. When we approximate the collection of polygons with a bounding volume of similar dimensions and orientations, the candidate silhouettes would be more accurate. As S. Gottschalk et al. [6] mentioned that an OBB is a tight-fitting bounding volume and aligns itself with the geometry. Hence, we take advantage of OBBs for clustering, and entering the polygons into a hierarchical clustering tree, or more precisely, an OBBTree.

2.2.2 Neighboring OBBs

The neighboring OBBs play the same role of adjacent polygons in the inclusion relationship. To choose appropriate neighboring OBBs is important, since they dominate the result of inclusion test. Now we assume that an OBBTree is a complete binary tree, the sibling nodes would represent almost equal dimensions of polygons. The sibling nodes are candidates of neighboring OBBs, since they could be close to the circumstance of a sub-mesh. In other words, the result of inclusion test would not easily to be affected by a fraction of polygons. We use the interference method, as mentioned in OBBTree, to quickly determine neighboring OBBs of an OBB.

In practice, an OBBTree is not always a complete binary tree. When the places of sibling nodes in the complete binary tree corresponding to the OBBTree are empty, then their parents become candidates. This will ensure that an OBB is surrounded by neighboring OBBs, in other words, no breaks on the boundary of OBB. Figure 1 illustrates the spatial relationship between an OBB and its neighboring OBBs.

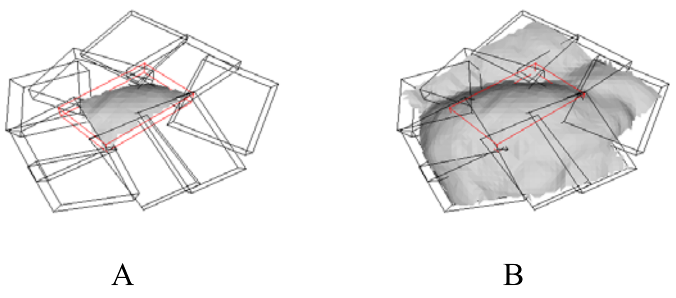

Figure 1: A: an OBB of sub-meshes placed at center with its neighboring OBBs. B: an OBB and its neighboring OBBs located on the portion of model.

2.2.3 Candidate Silhouettes Extraction

Now we describe our candidate silhouettes extraction method using an OBBTree. Since our approach is similar to silhouettes detection, we will determine a normal and a view direction for each OBB. The normal of an OBB should correspondingly represent the orientation of underlying geometric polygons. This is done by finding an average normal, the representative normal, from facet-normals contained in the node. Since the center of OBB is the centroid of polygons, the view direction is determined by the view point and center of OBB.

For a given view point and an OBBTree, we traverse the tree from top to bottom, while applying inclusion test to nodes. The polygons attached to edges contained in a node and in all of its descendants. If we determine that a node is entirely inclusion, then none of edges contained in the node's subtree could be the outermost silhouettes, and thus depth-first traversal skips subtree below the node.

In order to choose the suitable nodes, the orientation of normal vectors contained in an OBB should not be drastically varied; in other words, we must at least avoid opposite normal vectors. Let the ***maximal variation angle***, MVA, be the maximal angle between the representative normal, $\overrightarrow{RN}$, and $\overrightarrow{N_i}, i = 1,2,...,n$ be the normal vectors corresponding to polygons contained in a node, MVA is defined as

$$MVA = \min_{i=1...n}(\overrightarrow{RN} \bullet \overrightarrow{N_i}),$$

The maximal variation angle limits the variation of normal vectors, but it can not restrict the topology of polygons of sub-meshes. High curvature of polygons still causes inaccuracy, so we measure the flatness of each polygon cluster. The ***flatness*** is considered as.

$$flatness = \min E / \max E,$$

where the minE is the minimal extent and the maxE is the maximal extent of an OBB.

Here, we summarize the data structure of an OBB node and the procedure of inclusion test as follows:

```
Struct node {
    vector OrthogonalBasis // normalized eigenvectors of covariance matrix
    point OBBCenter // the centroid of polygons of OBB
    vector RepresentativeNormal //average normal of OBB
    float MVA
    float flatness
    NeighborhoodList Neighbors //neighboring OBBs
}
```

```
Function Inclusion(Node n, Viewpoint p)
    if ((n.flatness, n.neighbors.flatness) $>$ flatness_threshold) &&
    ((n.MVA, n.neighbors.MVA) $<$ MVA_threshold)
        return false  //not a suitable node
    if  (p $\in$ front-facing(n)  &&  p $\in$ front-facing(n.neighbors))  ||  (p $\in$ back-facing(n)  &&
    p $\in$ back-facing(n.neighbors)) //inclusion test
        return true    //skip this subtree
    else
        return false  //not inclusion
```

The candidate silhouettes extraction will produce a silhouette edge list, and that are less than the actual silhouettes, but containing the outermost silhouettes. The framework of the candidate silhouettes extraction is summarized below:

```
Procedure CSE(Node n, Viewpoint p)
  if !Leaf(n) //check if n is leaf node
      if(!Inclusion(n, p)) //inclusion test
          CSE(n.LeftChild, p)
          CSE(n.RightChild, p)
  else
      FindSilhouetteEdges(n, p) //Find silhouette edges of the triangle with
                                //respect to viewpoint p, that is,
                                //candidate silhouette edges
```

2.2.4 Outermost Silhouettes Extraction

To extract the outermost silhouette edges of a polygonal object in a scene from a given viewpoint, we must solve the partial visibility problem so that only those parts of silhouette edges are extracted.

Consider a polygonal object, O, composed of polygons, $T_i \in O$. Every edge, $E_j \in T_i$, is shared by exactly two polygons $T_1(E_j)$ and $T_2(E_j)$ with facet-normals, $\overrightarrow{T_1N}$ and $\overrightarrow{T_2N}$ .An edge is a silhouette edge if one of its adjacent polygons is front-facing and the other is back-facing with respect to viewpoint, V .

Definition 1 An edge, E_j , is a silhouette edge if $(\overrightarrow{T_1N} \bullet \overrightarrow{P_{T1}V})(\overrightarrow{T_2N} \bullet \overrightarrow{P_{T2}V}) < 0$, in which P_{T1} is any vertex of T_1 and P_{T2} is any vertex of T_2

Every vertex, $P_k \in E_j$, is shared by polygons, $T_{adj}(P_k)$, that denoted the adjacent polygons of P_k.

Definition 2 A vertex, P_k , is considered as an outermost vertex if the ray $\overrightarrow{VP_k}$ is non-intersecting with T_i except $T_{adj}(P_k)$.

The set of all silhouette edges of O will be denoted $S(O)$. Let ε be the set of outermost vertices of E_j. Every edge, E_j was the union of points, $P(E_j)$,

Definition 3 E_j is an outermost silhouette edge if $E_j \in S(O)$ and $\exists P(E_j) \in \varepsilon$.

In object space, to decide an outermost silhouette edge is difficult and time consuming, since an edge is a series of continuous vertices. For speed up, we just sample one vertex of edge to verify whether it is an outermost silhouette edge. The extreme vertices of edge may share by a number of polygons; hence, they are excluded from sampling vertices. The remaining vertices of edge are bordered by two polygons; hence, they are all suitable. In our current implementation, the center of edge is the sampling vertex. We utilize the ray-casting technique to validate whether an edge is outermost silhouette edge.

From the stage of candidate silhouettes extraction, we had obtained a set of edges and identified some nodes as inclusion case. For accelerating the state of an edge is quickly determined, the inclusion nodes, OBBs, are treated as polygons of model. Every candidate silhouette edge will associate a ray which is formed with viewpoint and the center of edge. For a given OBBTree and ray, we traverse the OBBTree in a level-order fashion. If the ray is non-intersecting with any inclusion node or polygon [10] except adjacent polygons, the edge is an outermost silhouette edge. In practice, the inclusion node or bounding volume is always larger than the underlying geometry polygons. That will cause missed outermost silhouette edges, since the bounding volume is intersected by the ray easily. Therefore, we adopt its descendants as inclusion instances which are practical intersected objects as shown below:

```
Procedure InclusionDownLevel(node n, downlevel L)
if L is zero
    mark the node n as inclusion instance
else
    L=L-1
    if !Leaf(n) //check if n is leaf node
        InclusionDownLevel(n.LeftChild, L)
        InclusionDownLevel(n.RightChild, L)
    else
        mark the node n as inclusion instance
```

The state of inclusion instance should be resolved before the stage of outermost silhouette extraction; hence, the pseudo code of candidate silhouette extraction is modified as follows:

```
int level //global variable
Procedure CSE(node n, viewpoint p)
    mark the node n as not inclusion instance
    if !Leaf(n) //check if n is leaf node
        if(!Inclusion(n, p))  //inclusion test
            CSE(n.LeftChild)
            CSE(n.RightChild)
        else
            InclusionDownLevel(n, level)
    else
        FindSilhouetteEdges(n, p)
```

The state of outermost silhouette edge is determined as shown in below:

```
Function IsOMSEdge(Ray OMSRay)
Queue<TreeNode *> q;
TreeNode* CurrentNode=root
while(CurrentNode)
    if OMSRay intersect CurrentNode
        //the default setting of polygons is inclusion instance
        if CurrentNode is Inclusion Instance and not adjacent polygon
```

```
                return false      //not outermost silhouette edge
            else
                q.Add(CurrentNode->LeftChild)
                q.Add(CurrentNode->RightChild)

        CurrentNode=*q.Delete(CurrentNode) //retrieve next node

    Return true      //outermost silhouette edge
```

3. Results and Discussion

We have applied the proposed approach implemented in C++ to five polygonal models. Figure 2 illustrates the models with which we experimented. Table 1 shows the model complexity. All experiments were preformed on a PC with one 667MHz PentiumIII, 512 Mb of memory, using an Elsa Geoforce2GTS graphic card, and running the Windows 2000 operating system.

Preprocessing a model consists of building an OBBTree, neighboring OBBs, flatness of OBBs, maximal variation angle of OBBs, and representative normals. This takes between one and fifteen minutes depending on the model complexity.

The exactly outermost silhouettes extraction algorithm was not presented to date. Hence the measurement of efficiency and accuracy was based on the fine primitive, triangle, with OBBTree. In other words, the intersection of OBBs are being substituted for triangles, and thus producing the optimal outermost silhouettes. We will call this a brute-force approach in the following discussions.

The run-time method was divided into two stages, the candidate and outermost silhouettes extraction. At the stage of candidate silhouette extraction, the parameters, maximal variation angle and flatness, will cause our approach to miss edges of outermost silhouettes. The choosing of descendants of inclusion node at the outermost silhouette extraction stage also causes missed edges. Hence, we compare acquired edges of each stage with outermost silhouettes, obtained from brute-force approach, to measure the error ratio.

For each model we ran the same experiments set up as follows. The viewpoint position was moved between 2.0 and 4.0 along z axis while x and y coordination were fixed to 0. The model was placed at the origin and rotated around three axes in equal momentum. In other words, almost all parts of a given model were examined in different viewpoints. We then recorded the number of edges and elapsed time at each frame. We first explain the meaning of abbreviation in tables here:

#SE: silhouette edges

#CE: candidate silhouette edges

#BOMS: outermost silhouette edges of brute-force approach

#OMS: outermost silhouette edges of our approach

#eOMS: missed outermost silhouette edges

error %: error ratio of missed edges to outermost silhouettes

culling %: culling ratio of silhouette edges

BOMST, OMST: outermost silhouettes extraction time of brute-force approach, and outermost silhouettes extraction time of our approach. The unit of measurement is in millisecond.

DL: the level of descendant of inclusion node

Table 2 summarizes the differences between the brute-force and our approach. The culling ratio depends on the flatness of sub-meshes and variation of geometric shape of model. The errors of the stage of candidate silhouettes extraction were caused by the center of OBBs, representative normls, and the location of viewpoint. Figure 3 illustrate the optimal outermost silhouettes of models. The speedup factors of all tested models at stage of outermost silhouettes extraction are faster than the brute-force approach. In addition to errors of candidate silhouettes extraction, the other errors were caused by the OBBs, since they are larger than the underlying geometry polygons. But the appearance of outermost silhouette edges is still obvious. Experimental results of the testing models are shown from Figure 4 to Figure 8 respectively.

In all cases, our outermost silhouettes extraction time is lower than the brute-force approach. Figure 10 show the comparison of outermost silhouettes complexity for several simplified bunny meshes. The graph indicates that the time for our approach increases sub-linearly with respect to the number of silhouette edges in the model, whereas the brute-force approach increases linearly.

The complexity of extracted outermost silhouette edges should be adapted to applications. That's unnecessary to extract exactly outermost silhouette edges such as motion blur as shown in Figure 9, since the approximating outermost silhouette edges also retain the principal feature of a given model.

4. Conclusions and Future Works

We have presented a simple approach to interactively extract the approximating outermost silhouettes of polygonal models under perspective projection. All polygons of the input model are put into an OBBTree, and several necessary parameters are maintained in each node. At run-time, the approximating outermost silhouette edges are quickly extracted according to constrained conditions. The results reported here should be beneficial to researchers attempting to portray more fantastic visual effect of polygonal models.

In this work, we have selected an OBB for occlusion simplicity. Yet, one could expect a somewhat more accurate if a *k*-DOPs [8] will be used. Besides, the number of neighboring OBBs of OBB is fixed and almost equal size. This will limit the culling ratio of silhouettes. One could adopt adaptive neighboring OBBs. When the testing OBB is more front-facing or back-facing, the probability of becoming outermost silhouettes is to decrease. We can choose small size neighboring OBBs, since larger one could cause the representative normal tends towards opposite orientation with respect to testing OBB. Thus that would lead inclusion test into failure, in other words, no culling occurs.

We present a practical application, motion blur, which uses our approach to further processing and make pretty visual effect. We are continuing further research in these application areas to make the best use of outermost silhouettes.

5. References

[1] F. Benichou and G. Elber, "Output sensitive extraction of silhouettes from polygonal geometry," Proc. 7th Pacific Graphics Conference, 60-69, 1999.

[2] G. J. Brostow and I. Essa, "Image-based motion blur for stop motion animation," Proceedings of the 28th annual conference on Computer graphics and interactive techniques, August 2001.

[3] J. W. Buchanan and M. C. Sousa, "The edge buffer: A data structure for easy silhouette rendering," Proc. 1st Int. Symp. on Non Photorealistic Animation and Rendering, 39-42, 2000.

[4] J. Cohen, M. Olano, and D. Manocha, "Appearance preserving simplification," Proc. ACM SIGGRAPH, 115-122, 1998.

[5] P. Cignoni, C. Montani, C. Rocchini, and R. Scopigno, "A general method for preserving attribute values on simplified meshes," Proc. IEEE Trans. Visual. Comput. Graphics, 59-66, 1998.

[6] S. Gottschalk, M. Lin, and D. Manocha, "OBBTree: A hierarchical structure for rapid interference diction," Proc. ACM SIGGRAPH, 171-180, 1996.

[7] D. E. Johnson and E. Cohen, "Spatialized normal cone hierarchies," Proc. Symp. on Interactive 3D Graphics, 129-134, 2001.

[8] James T. Klosowski, Martin Held, Joseph S.B. Mitchell, Henry Sowizral, and Karel Zikan, "Efficient collision detection using bounding volume hierarchies of *k*-DOPs," Proc. IEEE Trans. Visual. Comput. Graphics, 21-36, 1998.

[9] M. Kaplan, B. Gooch, and E. Choen, "Interactive artistic rendering," Proc. 1st Int.

Symp. on Non Photorealistic Animation and Rendering, 67-74, 2000.

[10] T. Möller and B. Trumbore, "Fast, minimum storage ray-triangle intersection," Journal of graphics tools, Vol. 2, No. 1, 21-28, 1997.

[11] J. D. Northrup and Lee Markosian, "Artistic silhouettes: A hybrid approach," Proc. 1st Int. Symp. on Non Photorealistic Animation and Rendering, 31-38, 2000.

[12] M. Pop, G. Barequet, C. A. Ducan, M. T. Goodrich, W. Hung, and S. Kumar, "Efficient Perspective-Accurate Silhouette Computation and Applications," Proc. 17th International Annual Symposium on Computational Geometry, 60-68, 2001.

[13] P. Ramanathan, E. Steinbach, and B. Girod, "Silhouette-based multiple-view camera calibration," Proc. Vision, Modeling and Visualization, 3-10, 2000.

[14] R. RASKAR and M. Cohen, "Image precision silhouette edges," Proc. Symp. on Interactive 3D Graphics, 135-140, 1999.

[15] P. V. Sander, X. Gu, S. J. Gortler, H. Hoope, and J. Snyder, "Silhouette clipping," Proc. ACM SIGGRAPH, 327-334, 2000.

[16] J. C. Xia, J. El-Sana, and A. Varshney, "Adaptive real-time level-of-detail-based rendering for polygonal models," Proc. IEEE Trans. Visual. Comput. Graphics, 171 – 183, 1997.

[17] Y. Zhou and A. Toga, "Efficient skeletonization of volumetric objects," IEEE Trans. on Visualization and Comput. Graphics, 5(3), 195–206, 1999.

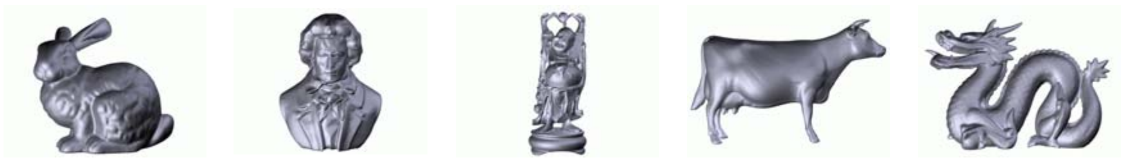

Figure 2: Five tested models.

Model	Vertices	Facets	Edges
Bunny	34837	69664	104499
Beethoven	25005	50000	75003
Buddha	25129	49999	75294
Cow	25029	50037	75070
Dragon	25362	50000	75616

Table 1: Complexities of tested models.

	Bunny	Beethoven	Buddha	Cow	Dragon
					Candidate silhouettes extraction stage
#SE	3445	2462	5270	2451	4790
#CE	2366	1860	4600	2221	4277
culling%	31.32	24.43	12.71	9.41	10.70
error%	7.41	3.20	1.37	2.01	0.89
					Outermost silhouettes extraction stage
#BOMS	969	489	576	930	654
#OMS	870	454	541	858	595
#eOMS	98	35	35	71	59
error%	10.16	7.18	6.07	7.67	9.03
BOMST	516.94	415.76	936.72	369.22	942.54
OMST	165.80	154.66	431.78	157.48	325.98
speedup	3.12	2.69	2.17	2.34	2.89
					Constrained parameters
Flatness	0.271731	0.255534	0.200803	0.154298	0.212615
MVA	3.58868e-006	5.00005e-006	5.00015e-006	4.99635e-006	5.00005e-006
DL	4	4	2	2	2

Table 2: Comparison of our outermost silhouettes extraction approach and the brute-force approach.

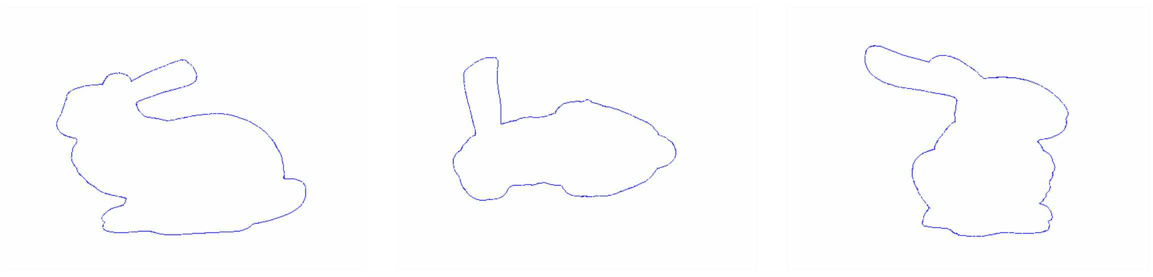
Figure 3: Bunny. Outermost silhouettes of the brute-force approach.

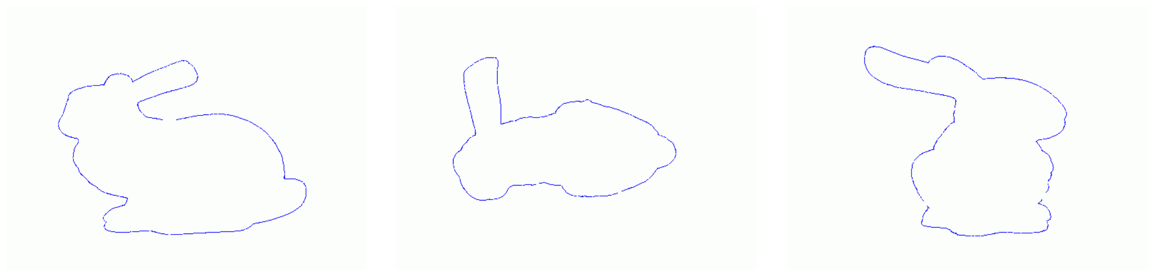
Figure 4: Bunny. Outermost silhouettes of our approach in which intersection objects are OBBs. The value of DL is four.

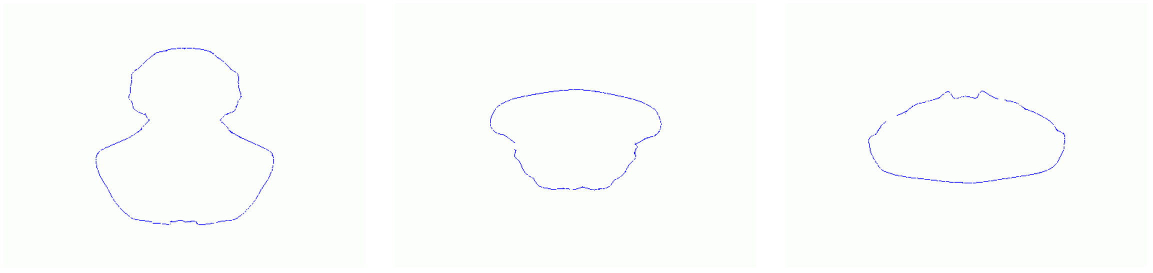
Figure 5: Beethoven. OMS of our approach in which intersection objects are OBBs. The value of DL is four.

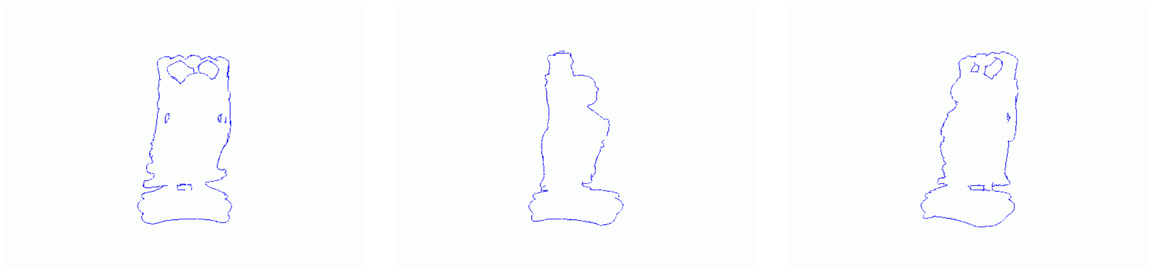
Figure 6: Buddha. OMS of our approach in which intersection objects are OBBs. The value of DL is two.

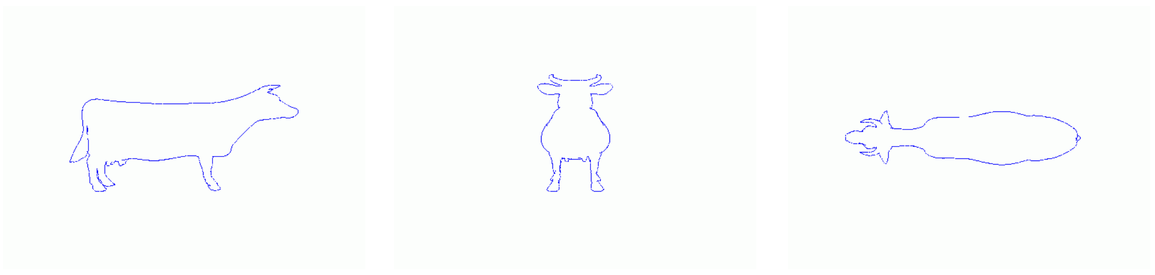

Figure 7: Cow. OMS of our approach in which intersection objects are OBBs. The value of DL is two.

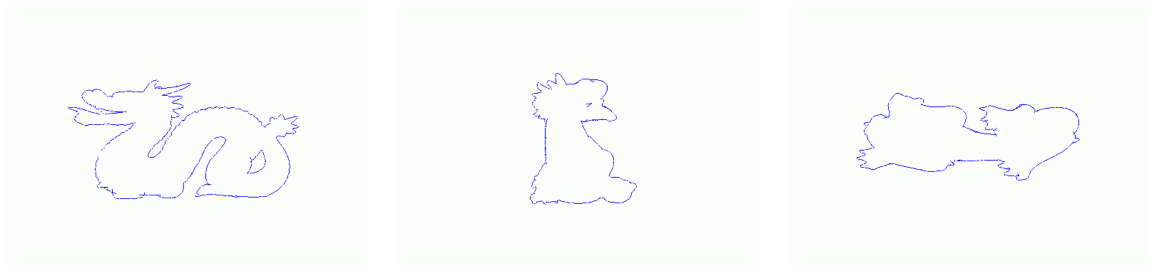

Figure 8: Dragon. OMS of our approach in which intersection objects are OBBs. The value of DL is two.

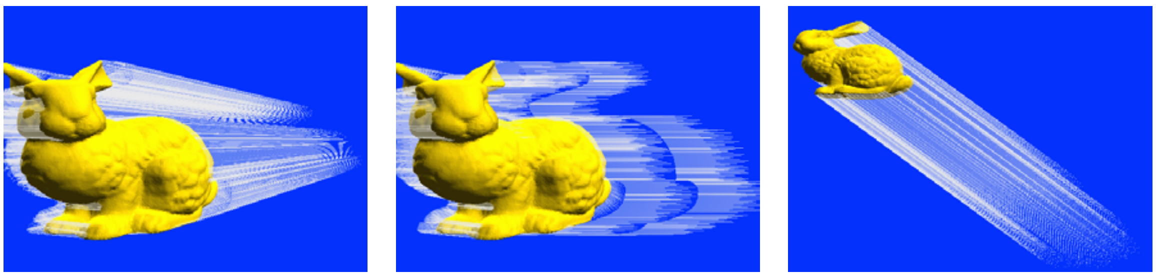

Figure 9: Motion blur of Bunny in which coarse outermost silhouettes create glamorous visual effect.

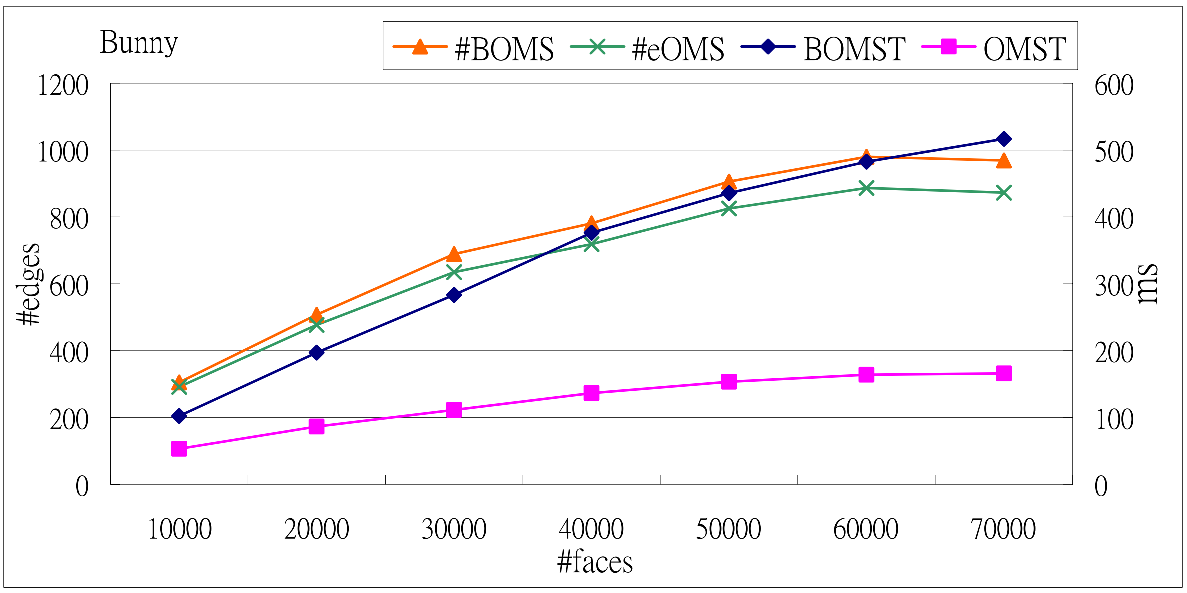

Figure 10: Comparison between our approach and brute force approach in different level of bunny.