Reverse Subdivision Multiresolution for Polygonal Silhouette Error Correction

Kevin Foster, Mario Costa Sousa, Faramarz F. Samavati, and Brian Wyvill

Department of Computer Science
University of Calgary,
Calgary, Canada
{fosterk,mario,samavati,blob}@cpsc.ucalgary.ca

Abstract. This paper presents a method for automatic removal of artifacts that appear in silhouettes extracted from polygonal meshes due to the discrete nature of meshes and numerical instabilities. The approach works in object space on curves made by chaining silhouette edges and uses multiresolution techniques based on a reverse subdivision method. These artifact-free curves are then rendered in object-space as weighted 3D triangle-ribbon strips.

1 Introduction

There has been significant research in non-photorealistic rendering focusing on quality silhouette extraction and rendering, in particular for 3D mesh-based silhouette line stylization algorithms [8,11,12,15]. Such algorithms are usually organized in four main steps: (1) extraction of individual silhouette edges from the mesh; (2) linkage of silhouette edges together to form long, connected paths, or chains; (3) removal of silhouette artifacts from the chains; (4) stylization of the strokes which involves two main sub-processes: smoothing the chain by fitting splines or using an interpolation/approximation scheme and controlling line quality attributes along the chain such as width and brightness.

A problem with extracting silhouette curves from polygon meshes is that the resulting curves may contain jagged artifacts because of numerical instability and unsuitable edges from the mesh (the mesh is a discrete approximation of a surface). These artifacts compromise the quality of the stroke stylization process and subsequent rendering results. Although there is a great deal of work which extracts silhouettes from polygonal meshes, there are few examples that attempt to correct errors and artifacts that can be created when this extraction takes place (*step 3*).

In this paper, we introduce a new approach to remove artifacts from chains of silhouette edges based on multiresolution. Because silhouettes created from polygonal meshes have a discrete nature, use of multiresolution systems that directly operate on discrete data are fitted effectively. Samavati and Bartels[1,13] provide this kind of multiresolution based on reversing subdivision. In their system, resolution can be increased and decreased efficiently without use of wavelets.

We employ this kind of multiresolution to remove silhouette artifacts automatically and efficiently. Furthermore, we can also use subdivision consistently with the multiresolution filters to contribute to the stroke stylization step.

2 Related Work

(1) Object-space silhouette extraction: There are many methods that extract silhouettes from polygonal meshes, including systems based on probabilistic testing [11], "Gauss Maps" [6], duality [7], cone maps[14] and adjacency information[2]. Any of these methods can be used with our error-removal system, provided they create linked silhouette chains. In this work, we extend the "edge-buffer" method[2] to create these chains.

(2) Removing silhouette artifacts: Works in this area either (1) correct errors from silhouette chains created from the actual mesh edges [8,12] or (2) create new, more suitable, silhouettes without use of the edges in the mesh [3, 7]. Correa et al.[3] avoid errors by creating 2D *u,v-images* which are basically projected images of the 3D scene with special colors on different u,v coordinates on the mesh. Their system analyzes pixel-neighborhoods and creates curves from the areas that contain silhouettes. Mesh edges are not used in this process; thus errors are avoided. Northrup and Markosian[12] remove errors by rendering raw silhouettes to image-space and case-checking. This includes elimination of undesirable silhouettes, redefinition of uneven endpoints so that they correspond and joining of edges to create smooth chains. Isenberg et al.[8] also correct silhouette errors directly from the edges using case-checks and solutions. However, their corrections are preformed in object-space. Hertzmann and Zorin[7] present an object-space approach that avoids errors by creating more suitable silhouette edges. These new edges are created by approximating points on the mesh edges where the silhouette would cross if the mesh was a smooth surface.

Our method, like Hertzmann and Zorin's[7], is general—we remove all errors without requiring classification of errors and evaluation of fixes. However, like Isenberg et al.[8] and Northrup and Markosian[12], our system removes errors from silhouette chains created from edges in the mesh instead of procedurally generating new edges. This approach desirable due to the speed and simplicity of extracting silhouette edges from a mesh.

(3) Multiresolution methods: Finkelstein and Salesin[5] demonstrate the first use of multiresolution in NPR with a curve-editing system based on wavelets. Furthermore, Kirsanov et al. [10] use coarsening methods to simplify silhouettes from detailed polygonal meshes. More information on this type of multiresolution is found in Stollnitz et al.[16]. We use a different type of multiresolution, "*local*" [1] and "*global*" [13] multiresolution, based on reversing subdivision to remove errors and provide a better system to simulate smooth pen strokes.

We now describe the main steps of our algorithm: (1) Create silhouette chains (Sec. 3), (2) Apply our multiresolution system to remove errors (Fig. 3, Sec. 4); and (3) Stylize the chains (Sec. 5). We then present and discuss results (Sec. 6) and provide conclusions and directions for future work (Sec. 7).

3 Silhouette Extraction

The definition of a silhouette edge for object-space methods is any edge shared by one front-facing polygon and one back-facing polygon. Upon loading a mesh, our system constructs an edge-buffer [2]. The edge-buffer, which can be viewed as an indexed graph of edges, is a fairly compact data structure that provides a fast lookup method requiring, for each frame, two binary operations per edge to extract silhouettes in object-space. Further details are supplied in [2].

3.1 Chains

The algorithm proceeds to the point where the edge-buffer has been traversed and all silhouette edges properly extracted. To better reproduce the artistic style described in Sec. 1, we use a two-pass algorithm to create a small number of long silhouette chains. As a first pass, our system links the silhouette edges on the model by finding the connected components of the edge-buffer. As a second pass, our system finds the matching vertex numbers on the bounds of each chain and joins these chains. If more than two chains can be linked, we join any chains that will create a loop first. Looping chains take precedence because our multiresolution system (Sec. 4) handles looping and non-looping chains slightly differently (non-looping chains are interpolated at the ends) and if a chain that should be looping is instead identified as two separate strokes, small artifacts might be created due to the interpolation at the ends of the chain.

This chaining method cannot guarantee the longest connected chains. However, it does generate satisfactory long chains for use with the multiresolution filters.

3.2 Artifacts

The chains extracted in the processes described above may contain artifacts such as zig-zags, overlaps and loops (Fig. 1). Such artifacts exist for two main reasons: (1) numerical instabilities in silhouette edge detection where many of the faces are viewed nearly edge-on; (2) meshes are just approximations of the underlying continuous surfaces and the edges that make them up are almost always unsuitable to be used as silhouette edges.

The set of four images in Fig. 1(b) illustrate different combinations of these artifacts. Silhouettes for these images have been calculated for an angle other than that displayed. Observe the black line, which is the actual silhouette, the unshaded front-facing polygons and the shaded back-facing polygons. As the silhouette crosses the surface, it moves back and forth across some invisible threshold, sometimes by many edges at a time. Clearly, edges taken directly from the mesh are not ideal to construct the silhouette. The invisible threshold that the extracted silhouette crosses is approximately where it should actually appear.

We interpret silhouette artifacts from the point of view of low and high-frequency portions of the silhouette curve. The extracted silhouette can be viewed as high-frequency noise components along the real silhouette curve. The

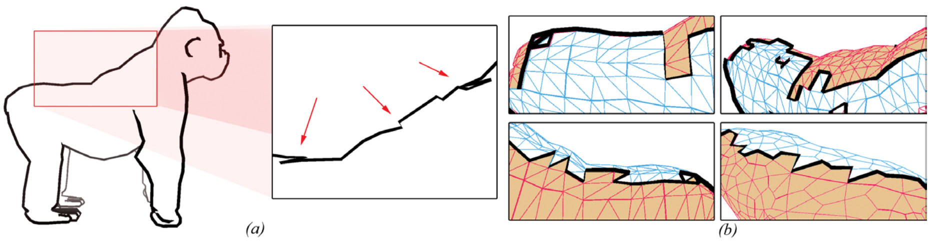


Fig. 1. **(a)** The silhouette of an ape mesh with highlighted errors. **(b)** Four images showing various silhouettes and underlying mesh that generated them. The silhouettes are presented at a perturbed view to provide a better understanding of the cause of the errors. Shaded polygons are back-facing.

challenge is to remove the high-frequency noise which occurs sporadically along the chain. We meet this challenge by using multiresolution filters, as described in the next section.

4 The Multiresolution Approach

The algorithm proceeds to where complete chains have been constructed from the silhouette edges. We denote these ordered sets of points as C^{k+1}. Using multiresolution, C^{k+1} can be decomposed to a low-resolution approximation C^k and a set of high frequency details D^k. Thus, C^k shows overall sweep of the silhouette and D^k shows waves and zigzags of the silhouette. In functional view, C^{k+1} is coefficient vector of high resolution scaling functions, C^k is coefficient vector of low resolution scaling functions and D^k is coefficient vector of Wavelet functions. The original data C^{k+1} can at any time be reconstructed from C^k and D^k. The process of transforming C^{k+1} to C^k and D^k is called *decomposition* and generating the original data C^{k+1} from C^k and D^k is called *reconstruction*. These can be applied to C^{k+1} more than one time. We can specify the multiresolution operations in term of the banded matrices A^k, B^k, P^k and Q^k. The matrix A^k transforms C^{k+1} to C^k:

$$C^k = AC^{k+1} \tag{1}$$

and B^k extracts details:

$$D^k = BC^{k+1} \tag{2}$$

P and Q act on C^k and D^k to reconstruct C^{k+1}

$$C^{k+1} = PC^k + QD^k \tag{3}$$

These matrices have a regular structure for every resolution. The only difference between A^k and A^{k-1} is their size. Consequently, the superscript of matrices can be removed. Because of the regularity of these matrices, they can viewed as filters that operate on C^{k+1}, C^k and D^k.

In order to find these four matrices, most multiresolution research works in the area of wavelets. In the case of smooth curves, the resulting wavelets are not very interesting (see appendix of Finkelstein[5] or page 94 of Stollnitz et al.[16]). With our method, C^{k+1} is a discrete approximation of a smooth curve and we just need to use appropriate A, B, P and Q and we do not need wavelets explicitly. Therefore, a discrete approach of multiresolution systems that directly operates on discrete data is fitted here more effectively. Bartels and Samavati[1] and Samavati and Bartels[13] provide this kind of multiresolution system based on reversing subdivision. In this kind of multiresolution, decomposition and reconstruction can be done efficiently without use of wavelets. They have also shown their results are more effective for data sets than conventional wavelets. In this work, we use their multiresolution filters that are constructed based on reversing Chaikin subdivision, Cubic B-Spline subdivision and Dyn-Levin subdivision. We present the masks of their Cubic B-Spline subdivision in Fig. 2a. These filters are much easier than their counterparts in Finkelstein and Salesin[5] and Stollnitz et al.[16].

For implementation, we just need to apply A and B on C^{k+1} to obtain C^k and D^k. Again by applying P and Q filters on C^k and D^k, or a modified version of D^k, we can reconstruct C^{k+1}. Note that these processes are simple linear time operations which do not use any extra storage. The resulting filters of Bartels and Samavati[1] are obtained based on solving the best C^k via a local least squares problem while the resulting filters in Samavati and Bartels[13] are obtained based on a global least squares problem. We call these two approaches *local* and *global* multiresolution. Note that these filters produce the optimum solution intrinsically without any extra-work in implementation. In the case of local multiresolution (Fig. 2), the implementation is very simple and straightforward. However, C^k is just a good approximation of C^{k+1} in a local sense. In contrast, C^k found from C^{k+1} with a global manner is the best solution possible (although it is more complicated than the local one). In fact, in the global case, the matrices A and B are full matrices. Nevertheless, they still have the regular structure. In order to achieve linear time operations, we solve the following banded system for decomposition [13]:

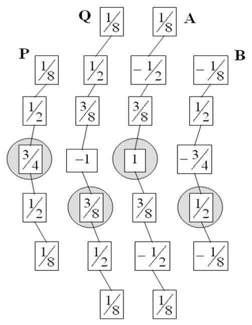
$$(P^tP)C^k = P^tC^{k+1} \tag{4}$$

$$(Q^tQ)D^k = Q^tC^{k+1} \tag{5}$$

In our experiments comparing local and the global multiresolution for silhouette error removal, we have found that the global MR generally creates better results (Sec. 6). However, the drawback of this approach is the need of solving the systems in equations 4 and 5.

4.1 Error Removal Pipeline

In this section, we provide details on how these filters can be used to eliminate silhouette artifacts. Our multiresolution pipeline consists of decomposing silhou-

(a)


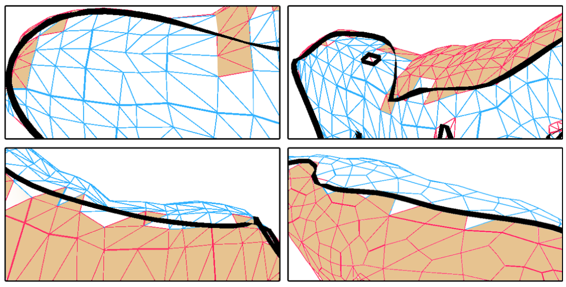
(b)


Fig. 2. (a) The bands of the matrices for Cubic B-Spline multiresolution (the A, B, P and Q diagrams represent all non-zero entities of a row for the A and B matrices and of a column for the P and Q matrices). The gray circles show the center entity. **(b)** Results of running our system on the silhouettes in Fig. 1b.

ettes to some level of detail, then reconstructing with only a small percent of the high-frequency details to remove errors (Fig. 3).

We modify equation 3 so that it can lessen the amount of details included in reconstruction:

$$\bar{C}^{k+1} = PC^k + eQD^k \tag{6}$$

where e is a scalar between 0.0 and 1.0 that varies the percentage of the details data added to the coarse data. The higher the value of e included, the greater the percent of the details data is included and the closer the stroke gets to the original data extracted.

Recall that the the low frequency path of the raw silhouette chain is generally correct (Fig. 1). The errors are all high-frequency divergences from this path. Since the high frequency portion of the silhouette chain is extracted and stored in details, a lower value for e eliminates more errors as a lower percent of the high-frequency details are included in the reconstructed strokes. We were able to generate accurate strokes suitable for scientific illustration with values from 0.0 to 0.4 for e, depending on the detail in the original mesh. A discussion of this is provided in Sec. 6.

Note that reconstruction can continue to a higher level of detail than the original chain. This is done by eliminating QD^k in equation 6 and results in an increase in smoothness. This is illustrated in the rightmost image in Fig. 3 (note quality improvement on the ape's head).

In our implementation, the user has control over the amount of times to decompose and reconstruct, the method to do this decomposition and reconstruction (Chaikin, Cubic B-Spline or Dyn-Levin), the scope of the method (local or global) and the amount of details to include in the reconstruction (the e value). Note that low-pass filters do not give this level of control.

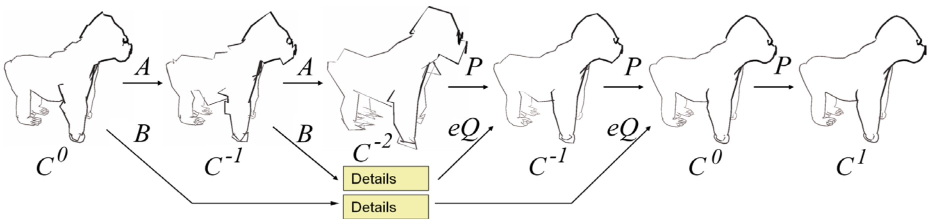


Fig. 3. We use Multiresolution filters to decompose and reconstruct silhouette chains without errors. Here is an example for an ape mesh with 7434 faces. We decompose twice from level C^0 to C^{-2} with **global cubic B-Spline** filters. Then, we reconstruct to level C^0 using minimized details (here, $e = 0.3$). The effect of this process is the removal of errors. Note that we can further process the mesh (to level C^1 or higher) without any details to smooth the strokes. This is equivalent to a subdivision step.

5 Rendering

For the results in this paper, we use the angled-bisector strip method as presented by Northrup and Markosian [12] and vary the weight and intensity of the stroke based on its depth into the scene. To preform Hidden Line Removal (HLR), we rely on the depth buffer. The original mesh is drawn in white and the strokes are drawn, slightly displaced towards the viewer. Thus, any strokes on the back of the surface will be occluded by the white mesh with the z-buffer. This approach does not work well for small meshes because the processed strokes do not follow the exact mesh; however it works well for medium to large size meshes. We leave an exact fast object-space solution to this problem for future work.

6 Results and Discussion

Our system achieves fast computation rates including preprocessing (building the edge-buffer) and rendering (chaining, multiresolution filtering, and stroke stylization). Furthermore, we have found that our method removes most errors with two levels of decomposition and reconstruction and a small value for e. Our method gains speed over other silhouette error correction methods because we do not need to identify errors to remove them. Thus, we do not need a large set of error condition/correction cases that must be evaluated locally for individual portions in the silhouette chain. However, this means that our method can inadvertently remove important features. Our system presents a tradeoff between feature-preservation and quality of filtering (directly related to the value e). Although this is not an issue for detailed meshes (features are preserved even with low levels of e), it can sometimes be impossible to remove errors from silhouettes of simple meshes without losing stroke accuracy (Fig.6).

We now present running times for different mesh sizes and the speed difference between local and global filters. Then we discuss quality of the results with notes on mesh size, user input and the different filter types.

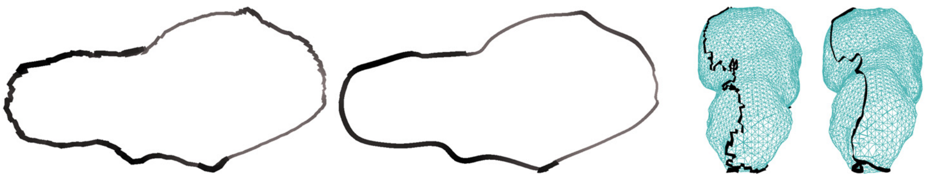


Fig. 4. From left to right: Original silhouettes from an asteroid, the results of processing and alternate views of the strokes with the mesh.

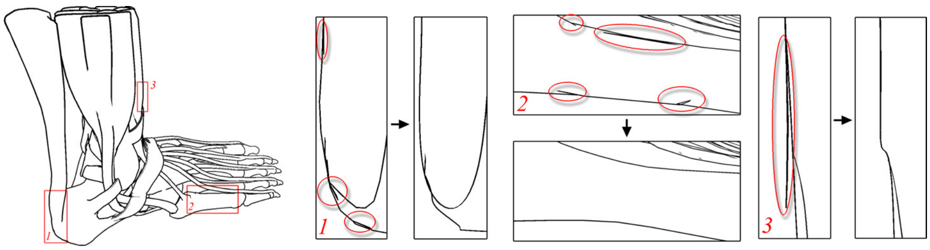


Fig. 5. Removing silhouette errors on large meshes is more important when zooming in on the mesh. Here, we circle the errors on three enlarged areas on the foot and provide our corrected strokes. Note that the errors are removed and the strokes are still very accurate to the mesh.

(1. Timing:) We have found that the local multiresolution filters generate realtime results for medium sized meshes (around 30,000 faces) and interactive rates for larger meshes. With two levels of decomposition and reconstruction and local Cubic B-Spline filters, the ox takes 0.414 milliseconds to filter (Fig.6), the ape 0.825 ms (Figs. 2, 3), the asteroid 1.065 ms (Fig. 4) and the foot 63.887 ms (Fig. 5). These results are ordered in increasing mesh size and are averaged from 256 tests with chains taken from the mesh at different angles. Clearly, our filters are efficient and even large meshes such as the foot run interactively.

As expected, the global multiresolution method is slower. For the ape and asteroid models, two levels of decomposition with global Cubic B-Spline filters take 7.779 and 19.75 ms respectively. This is a large increase over local times, but the method still preforms quickly for less detailed meshes where accuracy is most important. The added accuracy of global methods over local methods is not required for high resolution meshes. Running times and result images were gathered from a 2.65 GHz Pentium 4 with OpenGL/ATI Quadro graphics.

(2. User Input:) We found that medium to large meshes require little or no user-input (Figs. 4, 5). Error free strokes with no accuracy loss can almost always be generated with local multiresolution using two levels of decomposition and reconstruction and some small e value for details. The more detailed the mesh, the smaller e can be while still maintaining accurate strokes. We generally employed $e <= 0.1$ for meshes larger than 10,000 triangles. For smaller meshes (Figs. 3, 6) or for features on larger meshes only defined by several triangles, the user must use a global method (see next point on multiresolution type) and

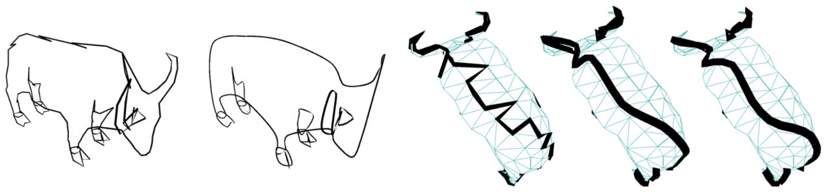


Fig. 6. Left to Right: A silhouette from a low resolution ox mesh, processed strokes with global Cubic B-Spline filters, an alternate view of the original and processed strokes, and processed strokes with global Chaikin filters. Note that the corrected strokes do not adhere well to the original mesh.

carefully adjust the amount of details and the decomposition and reconstruction steps to generate accurate strokes. It is in these situations that varying e results in a noticeable tradeoff between error-removal and feature preservation.

(3. Multiresolution Type:) We have tested the Cubic B-Spline, Dyn-Levin and Chaikin systems [1,13] with local and global multiresolution methods. The global method has produced more accurate results. This increase in accuracy can be seen in the right column of Fig. 2b where global methods are used compared to the left column where local methods are used). However, as presented in the timings section, the expense of global methods rises with mesh size. Fortunately, the extra accuracy given is usually only useful for small to medium sized meshes where the global method preforms in in realtime. For the figures in this paper, we have employed global methods for the smaller meshes in Figs. 2b(right column), 3 and 6 and have employed local methods for the larger meshes in Figs. 2b(left column), 4 and 5.

7 Conclusions and Future Work

We have presented a method to eliminate errors in polygonal silhouettes using multiresolution filters. Our method represents an improvement over previous works because it does not require specialized error/solution cases to remove errors—our solution is general. This improves efficiency over other methods because time is not spent identifying errors and looking up solutions. Furthermore, our system contributes to the stroke stylization step by automatically smoothing coarse chains. Finally, our system complements systems which create coarse approximations of silhouettes from very detailed meshes[10].

The drawback to our method is that it is not exact. We do not guarantee that errors will be removed and that accurate strokes can be generated for all meshes. Our approach can only be used to automatically generate accurate strokes for medium to large size meshes while coarser meshes can require a great deal of user input to create good output without losing detail. Finally, our method does not present a good way to eliminate unessential silhouette chains and an accurate hidden-line removal method must be developed for this system.

Despite these limitations, our method provides a new avenue to remove silhouette errors from polygonal silhouettes that is accelerated and more general. A future extension could be to improve the numerical stability during silhouette extraction with techniques that compute better normal vectors [17]. Furthermore, our system could be used to process non-silhouette strokes as an improvement to traditional B-spline or low-pass filtering methods [4,15]. Finally, our method could be combined with the approach presented by Kalnins et al.[9] for coherent silhouettes.

References

1. Bartels RH, Samavati FF (2000) Multiresolution curve and surface representation by reversing subdivision rules. Computer Graphics Forum, Vol. 18, No. 2: 97–120
2. Buchanan JW, Sousa MC (2000) The edge buffer: A data structure for easy silhouette rendering. Proc. of NPAR'00: 39–42
3. Correa WT, Jensen RJ, Thayer CE, Finkelstein A (1998) Texture mapping for cel animation. Proc. of SIGGRAPH'98:435–446
4. DeCarlo D, Finkelstein A, Rusinkiewicz S, Santella A (2003) Suggestive contours for conveying shape. Proc. of SIGGRAPH'03: 848–855
5. Finkelstein A, Salesin DH (1994) Multiresolution curves. Proc. of SIGGRAPH'94: 261–268
6. Gooch B, Sloan PJ, Gooch A, Shirley P, Riesenfeld R (1999) Interactive technical illustration. 1999 ACM Symposium on Interactive 3D Graphics: 31–38
7. Hertzmann A, Zorin D (2000) Illustrating smooth surfaces. Proc. of SIGGRAPH'00: 517–526
8. Isenberg T, Halper N, Strothotte T (2002) Stylizing silhouettes at interactive rates: From silhouette edges to silhouette strokes. Proc. of Eurographics'02
9. Kalnins RD, Davidson PL, Markosian L, Finkelstein A (2003) Coherent stylized silhouettes. Proc. of SIGGRAPH'03: 856–861
10. Kirsanov D, Sander PV, Gortler SJ (2003) Simple silhouettes over complex surfaces. Proc. of Symposium on Geometry Processing
11. Markosian L, Kowalski MA, Trychin SJ, Bourdev LD, Goldstein D, Hughes JF (1997) Real-time nonphotorealistic rendering. Proc. of SIGGRAPH'97: 415–420
12. Northrup JD, Markosian L (2000) Artistic silhouettes: A hybrid approach. Proc. of NPAR 2000
13. Samavati FF, Bartels RH (1999) Multiresolution curve and surface representation by reversing subdivision rules. Computer Graphics Forum, Vol. 18, No. 2: 97–120
14. Sandler PV, Gu X, Gortler SJ, Hoppe H, Snyder J (2000) Silhouette clipping. Proc. of SIGGRAPH'00: 327–334
15. Sousa MC, Prusinkiewicz P (2003) A few good lines: Suggestive drawing of 3D models. Proc. of Eurographics'03: 327–340
16. Stollnitz EJ, Derose TD, Salesin DH (1996) Wavelets for computer graphics: Theory and applications. Morgan Kaufmann, San Francisco
17. van Overveld C, Wyvill B (1997) Phong normal interpolation revisted. ACM Transactions on Graphics 16(4):397–419