Polygonal Silhouette Error Correction: A Reverse Subdivision Approach

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

***Abstract*— A method for automatic removal of artifacts and errors that can appear in silhouettes extracted from polygonal meshes is presented and evaluated. These errors typically appear in polygonal silhouettes due to the discrete nature of meshes and numerical instabilities. The approach presented works in object space on silhouette curves made by chaining together silhouette edges and uses multiresolution techniques based on reverse subdivision. Two hidden line removal methods along with a traditional method to render strokes as 3D triangle strips in object space are also presented.**

***Index Terms*— Non-photorealistic rendering, Reverse Subdivision Multiresolution, Polygonal Silhouette Artifact Removal**

I. Introduction

Artists and technical/scientific illustrators commonly use line drawings to effectively represent the form of 3D objects. Such drawings are usually termed *pure* line drawings, consisting entirely of lines that define the edges of shapes and use of no tones [1] (Fig. 1). The medium of choice is typically pen-and-ink due to its several appealing properties. Pen strokes can represent virtually any shape if used properly [2], [3]. This makes them ideal for printing and harmonizing with text due to their scale and use of the same ink [4]. Also, pen-and-ink images handle duplication and degradation much better than images created with traditional half-toning processes, This article examines a particular form of pure pen-and-ink illustration: contour (silhouette) drawings.

A contour (silhouette) drawing only shows the outline of the subject, and usually does not use interior strokes (Fig. 1). Artists place a great deal of attention on illustrating contours and use them for many applications such as in cartoons, technical illustrations, architectural design and medical textbooks. A general principle in drawing states that an accurate set of contours highlighting an outline provides good perception of mass [1]. This principle is supported by studies in perceptual rendering which reveal that a few simple lines defining the contour of an object often suffice to determine its 3D surface [5]. Contours also convey important cues to distinguish between different objects and for object-to-ground recognition [6].

Artists illustrate contours with two processes [1], [3], [7]:

- by emphasizing the placement of the subject's outline outside the silhouette boundary of its form rather than within it.

Kevin Foster, University of Calgary
Mario Costa Sousa, University of Calgary
Faramarz F. Samavati, University of Calgary
Brian Wyvill, University of Calgary

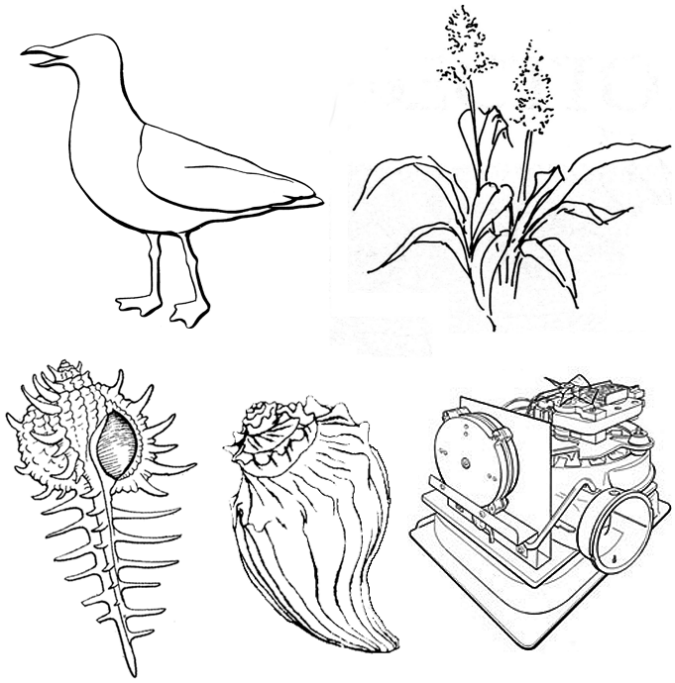

Fig. 1. Several illustrations consisting of contour (silhouette) strokes.

- by carefully controlling the various characteristics or qualities of the line, in particular the suggestion of movement which is achieved by drawing long line segments with various degrees of linear weight and emphasis.

Observe these processes in Fig. 1. The weight and emphasis variation depends on the subject matter and on the information that the illustrator wants to present. For example, observe how line width variation on the seagull image (Fig. 1, top-left) is used to imply shadow.

Non-photorealistic rendering (NPR) is a young research field in computer graphics that aims to provide techniques to help create images in expressive and interpretive styles such as pen-and-ink. There has been significant research in non-photorealistic rendering on contour extraction and stylization, in particular for 3D polygonal mesh-based line stylization algorithms [8], [9], [10], [11]. Note that in NPR, contour drawings are referred to as silhouette drawings, and this convention is used for the remainder of this article. 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 errors and artifacts from the chains.
4) Stylization of the strokes.

Step 4 involves two main sub-processes: (1) smoothing the stroke by fitting splines or using an interpolation/approximation scheme and (2) creating line quality attributes in the stroke such as width and brightness. Although there is a great deal of work which extracts and stylizes silhouettes from polygonal meshes (*steps 1,2 and 4*), there are few examples that attempt to correct errors and artifacts that can be created with object-space extraction approaches (*step 3*, Figs. 4 and 5).

Object-space extraction is desirable because it allows for control of stroke stylization, extracts silhouettes at a geometric level instead of a pixel level and can also extract hidden silhouettes from a surface. The artifacts that object-space extraction creates occur because of numerical instability and unsuitable edges from the polygonal mesh (the mesh is a discrete approximation of a surface). The quality of the stroke stylization process (step 4) and subsequent rendering results are compromised due to these errors.

Correcting these errors has been previously explored [8], [12], [10], [13]. These approaches remove errors with a set of error-cases and corresponding solutions or they extract sub-polygon silhouettes that do not contain errors. The results of these methods are bound to the resolution and dimensions of the polygonal mesh.

Our approach uses multiresolution based on reversing subdivision [14], [15] to remove errors from the discrete raw silhouette data (Sec. V). This approach results in good approximations to traditional hand-drawn pen-and-ink silhouettes (Fig. 1). This system is resolution-independent as the multiresolution error removal filters can create coarse approximations of chains and can smooth the strokes to a higher level of detail than that of the original data. The primary advantages of this method are (1) error-corrected sub-polygon strokes can be generated from the raw silhouette data without the need to inspect individual error-cases and (2) the strokes generated with this method include automatic stylization controllable by the user to create strokes with various levels of accuracy.

The remainder of this article is organized as follows. First, we provide background and related work (Sec. II). Then, we introduce our system pipeline (Sec. III), describe our technique to generate silhouette chains (Sec. IV), provide details for our method to remove errors with multiresolution filters (Sec. V) and review our stylization approach (Sec. VI). We then provide results (Sec. VII) and conclusions with direction for future work (Sec. VIII).

II. Background and Related Work

We now present an overview of the work related to this article. First, research focussing on silhouette extraction is detailed (Secs. II-A to II-D). Then, we describe silhouette errors and artifacts and provide methods that correct these errors (Sec. II-E). Finally, we provide a review of multiresolution techniques (Sec. II-F).

A. Silhouette extraction

There is a large body of work covering silhouette extraction and stylization. Efficient silhouette extraction is important because silhouettes are view-dependent and need to be reevaluated for each frame in an animation or after each viewing adjustment. These methods work in *object-space* (a 3D geometry-based approach), *image-space* (a 2D pixel-based approach) or use a combination of both. Before presenting a review of silhouette extraction methods, the definition of a silhouette as used by researchers in NPR is provided.

B. Definition of a Silhouette

The traditional, artistic definition of a silhouette is the outline of an object, or the boundary surrounding an object's shadow when the view direction is the same as the lighting direction (Fig. 2). The NPR definition of a silhouette for a 3D surface differs from this slightly. In NPR, the silhouette is defined as the curve on a surface where the normal direction[1] at every point on the curve is ninety degrees from the view direction (the direction from the eye to the point on the curve). This means that an object can have many silhouettes, that they can be inside the shadow boundary described above and that some of these can be occluded, depending on the dimensions and orientation of the object as displayed to the viewer.

Fig. 2. A silhouette drawing (using the artistic definition of a silhouette).

Mathematically, the silhouette for a smooth surface is defined as follows: a point X on a surface with normal $\mathbf{N}_X$ is on the silhouette for an eye position E if the angle between $\mathbf{N}_X$ and $(\overline{X-E})$ is 90 degrees (Fig. 3, left). This definition includes interior silhouettes as well as the object's outline. Unfortunately, this definition does not hold for polygonal meshes because surface normals for polygonal meshes are not defined at arbitrary points on the surface (they are usually only defined for each polygon or sometimes for each vertex). The silhouette set for polygonal meshes is defined as the set of edges which share a front-facing and a back-facing polygon (Fig. 3, middle). The orientation of polygons is found as follows. X is set to the polygon's midpoint. Assuming the polygon's normal $\mathbf{N}_X$ is pointing away from the surface, if $\mathbf{N}_X \cdot (\overline{X-E}) > 0$, it is back-facing and if $\mathbf{N}_X \cdot (\overline{X-E}) < 0$,

[1]The *normal* is the direction perpendicular to the surface at any position.

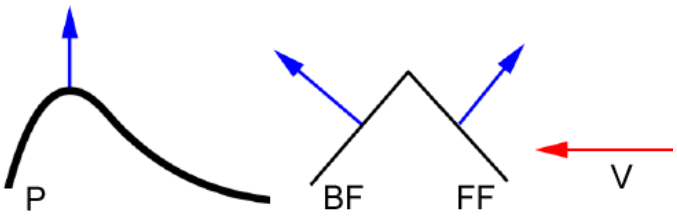

Fig. 3. A silhouette point and a silhouette edge. The rightmost arrow denotes the view direction V. *Left:* a silhouette point for a smooth continuous surface is defined as any point P whose normal is 90 degrees from the view vector. *Middle:* a silhouette edge for a polygonal mesh is defined as any edge which shares a front-facing (FF) and a back-facing (BF) polygon.

it is front-facing. The third case, $\mathbf{N}_X \cdot (\overline{X - E}) = 0$, means that the polygon is exactly edge-on to the view direction. In this case, all of the edges from the polygon are added to the silhouette set.

The simplest approach is to find the set of silhouette edges for a polygonal mesh is by brute-force. This method iterates through each edge in the polygonal model and finds silhouettes by determining the orientation for the polygons that the edge belongs to, as described in the previous paragraph. While this approach is easy to implement, there are many faster methods to find silhouette edges.

C. Silhouette Extraction Algorithms

There is a large body of work that explores silhouette extraction and stylization. These algorithms can be classified either as image-space or object-space (or both), whether they require visibility calculations or not, whether they extract silhouette edges or pixels and whether they allow animation or not [6] (Table I). Since our approach is primarily concerned with addressing issues of object-space extraction, this section focusses on object-space approaches. For completeness, image-space and hybrid approaches receive brief attention below.

Image-space algorithms analyze discrete 2D image buffers created with data projected from the 3D scene and extract discontinuities to create silhouette pixels [16], [17], [18]. Saito and Takahashi [16] present the foundation for image-space silhouette extraction with several filters that estimate the first-order and second-order differentials of the image. Image-space methods based on this approach are generally the fastest approach to extract silhouettes. This is because they solve silhouette detection and visibility in a single step and extract discontinuities from image buffers efficiently [16], [17]. Furthermore, these methods are often very simple to implement and entail little memory overhead. Unfortunately, image-space methods only extract visible silhouettes at the pixel-level; they suffer from aliasing artifacts and they do not lend themselves well to stylization. This is because individual pixels provide insufficient information to stylize a complete stroke and methods for applying stroke texture, realistically simulating width and stroke properties must manually be coded. Object-space approaches provide most of this functionality automatically.

Object-space algorithms are often used to extract silhouettes where stylization of the silhouette is required or when the actual 3D silhouette edges are required. Object-space algorithms extract geometric edges from the polygonal mesh (instead of pixels) and have access to surface information, such as the normal, for any point in the stroke. Furthermore, object-space approaches can extract all parts of the silhouette, not just those that are visible. They usually do this by comparing the view direction to surface normals as described in Sec. II-B and some methods use techniques to ignore edges that cannot be silhouettes. These properties make this approach much more suitable for stylization than image-space approaches. Unfortunately, object-space approaches are more computationally expensive than image-space approaches and often require a secondary process to determine silhouette visibility.

Hybrid algorithms attempt to maintain the fast extraction of image-space approaches and incorporate more stylization control using object-space. These approaches do not extract a geometrical representation of the silhouette. Instead, they use a special rendering pipeline which modifies the position of front-facing and back-facing polygons so that the silhouette is highlighted when rendered with the z-buffer [19], [20], [21], [22]. Most of these methods require several rendering passes to function. For example, Raskar and Cohen's approach [22] uses a 2-pass rendering. During the first pass, all polygons are rendered in the background colour with depth-testing enabled. During the second pass the polygons are rendered again, except this time they are draw in the silhouette colour with front-face culling enabled. Silhouettes appear by employing the equal-to depth function during this pass. The primary advantage of hybrid approaches is that they provide more stylization options than image-space methods at a comparable speed. Unfortunately, they do not provide the stylization control of object space approaches, can suffer from z-buffer inaccuracy and only provide a pixel-level representation of the silhouettes.

D. Object-Space Silhouette Extraction Methods

Specific techniques for object-space silhouette extraction will now be discussed.

The "Edge-Buffer" technique [23], used in this article (Sec. IV-A), extracts all silhouettes efficiently via a partial vertex adjacency graph. This graph contains a set of bits for each edge in the polygonal mesh. Every time the scene is rendered from a different angle or the model moves, these bits are modified on a per-polygon basis depending on if the polygon is front or back-facing. Then, individual silhouette edges are quickly extracted using bit-wise logical operators. The advantages of this method are that it works with animated surfaces, it extracts every silhouette edge (and other types of edges specified by the user) and it does not require the expensive preprocesses required by other guaranteed techniques detailed shortly [19], [12], [24].

Other research attempts to lower the number of silhouette tests stochastically by estimating which edges are most likely to be silhouettes. Markosian et al. [9] use probabilistic testing and chaining to find silhouettes. Their method tests edges that

Approach	Method		Additional Operations	Precision		Misc.	
Reference	Image Sp.	Object Sp.	Requires Visibility Test	Pixel	Edge	Allows Animation	Intelligent Extraction
[16]	√			√		√	
[17]	√			√		√	
[18]	√			√		√	
[19]	√	√		√		√	
[20]	√	√		√		√	
[21]	√	√		√		√	
[22]	√	√		√		√	
[23]		√	√		√	√	
[9]		√	√		√	√	
[19]		√	√		√		√
[12]		√	√		√		√
[24]		√	√		√		√

TABLE I
A CLASSIFICATION OF POLYGONAL SILHOUETTE EXTRACTION METHODS.

were silhouettes in previous rendering steps and random edges in the vicinity of these. When a silhouette edge is found, their method recursively follows the silhouette until it loops or degenerates. Unfortunately, this approach doesn't guarantee that untested edges are not silhouettes.

There are several other methods that lower the number of silhouette tests but guarantee that all silhouettes will be extracted [19], [12], [24]. These methods use various types of space partitioning to determine quickly which faces contain silhouettes. Unfortunately, spacial partitioning requires complicated implementation and intensive pre-processing. This makes such approaches not suitable for animation, memory intensive and challenging to implement.

Gooch et al. [19] present a system for interactive technical illustration of polygonal meshes. This system includes a module which colours the interior of the surface, a module which creates silhouettes and two hybrid silhouette revealing techniques. They also present an object-space software method which uses a preprocessing step to allow a fast runtime extraction. This preprocess projects the vertex normals for each edge onto a sphere called a "Gauss Map" and saves the arcs created. At runtime, silhouettes for a certain viewing direction can be found by determining the arcs which intersect a plane through the origin of the sphere. This method gains in efficiency by storing arcs in a hierarchy which allows for quick culling of regions that cannot contain a silhouette.

Hertzmann and Zorin [12] present a system which calculates hatch marks and silhouettes. Their system also employs a method that quickly culls faces which cannot contain a silhouette based on geometric duality. Each vertex is mapped onto a hypercube in 4D space using its position and tangent plane. The problem of finding faces which intersect the silhouette is reduced to intersecting the triangles in the 4D space with the viewpoint's dual plane. Any edge that intersects in this test intersects the silhouette. The system's speed gain comes from an octtree space subdivision which subdivides the vertices on each side of the hypercube and can quickly determine which groups of edges contain silhouettes.

Sander et al. [24] find groups of faces that might contain the silhouette using a hierarchical tree that stores mesh edges and their associated faces and "anchored cones". The system binds two cones to each node in the tree. One cone represents the viewpoint for the entire group of faces in a node to be front-facing, and the other cone represents the position of the viewpoint for all of the faces to be back-facing [6]. By determining if the viewpoint lies inside any of the cones, their system can quickly cull groups of faces which cannot contain a silhouette.

The processing time required to set up the data-structures used for some of these systems [24], [12] can be very expensive for large polygonal meshes, making animation and morphing difficult or impossible with current hardware. Despite this, these methods are useful to accelerate silhouette extraction for static surfaces where they provide a large speed increase over the brute-force approach.

E. Silhouette Error Correction

Silhouettes extracted directly from 3D meshes may contain artifacts such as "zig-zags", overlaps and loops (Figs. 4, 5). The causes of these artifacts are:

1) Numerical instability: Methods that extract edges from the polygonal mesh compare the result of a dot product operation with zero. This can return incorrect results where polygons are nearly edge-on to the view direction, due to floating point precision problems.
2) Unsuitable edges from the mesh: Since the mesh is a discrete approximation of a surface, edges that make up this mesh will rarely conform exactly to the "actual" silhouette. Depending on the connectivity and orientation of the edges, completely unsuitable edges might be used in the silhouette.

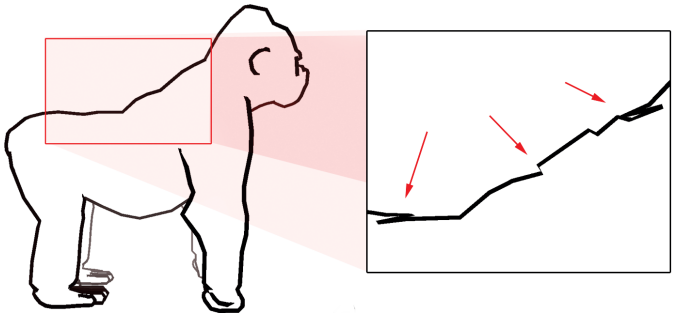

Fig. 4. The silhouette of an ape mesh with highlighted errors and artifacts.

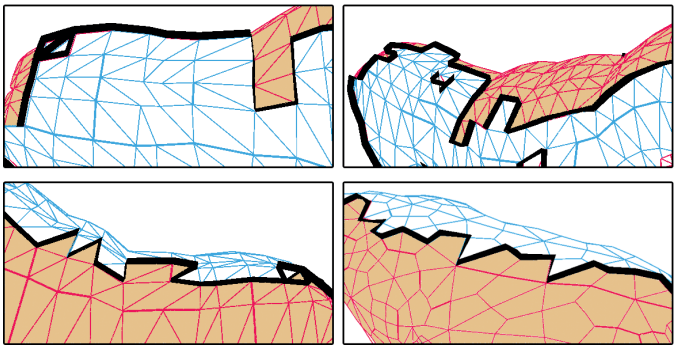

Fig. 5. Four images showing various silhouettes and underlying mesh that generated them. We present the silhouettes at a perturbed view to provide a better understanding of the cause of the errors. Shaded polygons were back-facing when the silhouettes were extracted.

These artifacts compromise the quality of the stroke stylization process and subsequent rendering results. In Fig. 5, four frames illustrate different combinations of artifacts. Silhouettes for these images have been calculated with view direction α, but are displayed with view direction β, such that $\alpha \neq \beta$. Note the black line, which is the extracted silhouette, the unshaded front-facing polygons and the shaded back-facing polygons. As the silhouette crosses the mesh surface, it moves back and forth across an invisible threshold which is the "actual" silhouette. Edges taken directly from the mesh only provide an approximation to where the actual silhouette should appear.

Table II lists methods that provide polygonal mesh silhouette error correction. These either (1) correct errors from silhouette chains created from the actual mesh edges [10], (2) create more suitable, sub-polygon silhouette lines without using the edges in the mesh [13], [12] or (3) do both [8]. These methods use object-space approaches [12], [8] or hybrid combinations of image-space and object-space [10], [13].

Côrrea et al. [13] create continuous, smooth silhouettes on a 3D model. This system uses an image-space solution to generate new sub-polygon silhouette edges. Their system creates 2D *u,v-images* which are coloured based on the u,v coordinates of the mesh. Discontinuities in this image correspond to visible silhouettes and boundaries on the 3D model and are found by analyzing pixel-neighborhoods. For each silhouette pixel, a silhouette edge is mapped into object-space using the depth buffer. Curves are created by joining these newly created silhouette edges. Unfortunately, this system requires a large amount of user input to function.

Northrup and Markosian [10] present a system which extracts silhouettes in object-space and performs corrections in image-space. Silhouettes are extracted using the process described in [9] and are projected to image-space. Then their system checks for error-cases and applies the corresponding solutions. These include elimination of undesirable silhouettes, joining uneven endpoints and other operations to create smooth chains. To stylize, the system renders corrected chains in image-space using an "artistic-stroke" method to create a wide range of expressive strokes and styles.

Isenberg et al. [8] also correct silhouette errors directly from the edges by checking for various error-cases and providing a solution for each case. In their system, however, stroke cleaning occurs in image-space and object-space. To accomplish this, their system uses a two-pass approach which first analyzes silhouettes based on the mesh and then checks their appearance. In the first pass, adjacent edges connecting at acute angles below a given threshold are replaced by a single edge, to prevent "zig-zag" patterns. Also, triangle clusters which create large artifacts on the silhouettes are identified and removed. At this stage, the silhouettes contain only edges from the mesh. The second pass removes silhouette "zig-zags" and short segments by analyzing vertices in image space merging them. Moreover, sharp angles in the silhouette are unlinked so that good output is created during stylization. At this stage, since vertices have been modified, the silhouette edges will not follow the exact geometry of the surface.

Hertzmann and Zorin [12] present an object-space approach that minimizes errors by creating new sub-polygon silhouette edges for smooth polygonal meshes converted from free-from surfaces. Their system relies on vertex normals instead of standard face normals. Silhouette edges are created by estimating the exact position where the silhouette crosses edges from the mesh by interpolating vertex-normals along the edge and joining adjacent pairs of these points to create edges. Their system also supports a hatching algorithm to create interior strokes along principal directions of curvature.

In contrast to these previous approaches, our error removal technique does not require classification of errors and evaluation of fixes. Furthermore, like Isenberg et al. [8] and Northrup and Markosian [10], our approach removes errors from silhouette chains created from mesh edges instead of procedurally generating new edges. Unlike these techniques, our approach modifies silhouette edges using sub-polygon resolution. Furthermore, the silhouettes thus generated are resolution-independent which is useful to simulate realistic pen strokes. This also proves to be useful for examining silhouettes from detailed meshes closely and when extracting silhouettes for simple meshes. Finally, the system can also generate artistic expressive strokes.

F. Multiresolution

Multiresolution (Sec. V) is a technique whereby a set of data can be *decomposed* into a set of coarse data and details, each of which is usually half the size of the original data. Then, the original data can be completely *reconstructed* using only the coarse data and details.

Finkelstein and Salesin [25] demonstrate the first use of multiresolution in NPR with a curve-editing system based on wavelets. More recently, Kirsanov et al. [26] use coarsening methods to simplify silhouettes from detailed polygonal meshes. More information on the wavelet multiresolution approach is found in Stollnitz et al. [27].

To remove errors from polygonal silhouette chains, our Multiresolution Error Removal (MAR) approach uses a different type of multiresolution that is based on reversing subdivision. This multiresolution, developed by Samavati and Bartels [14], [15], offers simple linear time operations [28] and several different filter sets to operate. Furthermore, Samavati and

Approach	Error-Removal Method		Correction Source		Precision		Silhouettes Corrected	
Reference	Image Sp.	Object Sp.	Polygon Edges	Sub-poly Edges	Pixel	Geometry	Visible	All
Côrrea et al. [13]	√			√	√		√	
Hertzmann, Zorin [12]		√		√		√		√
Northrup, Markosian [10]	√		√		√		√	
Isenberg et al. [8]	√	√	√			√	√	

TABLE II
A CLASSIFICATION OF POLYGONAL SILHOUETTE ERROR-CORRECTION METHODS.

Bartels present two versions of this approach, *local* [14] and *global* [15] filters. The local filters are obtained based on solving the best solution via a local least squares problem while the global filters are obtained based on a global least squares problem. These filters produce an optimal solution intrinsically without any extra work in implementation. In the case of the local filters, the implementation is very simple and the coarse data it generates is the best l^2 approximation for a set of data in a local neighborhood. In contrast, coarse data found with the global approach is the best l^2 approximation for the entire set of data. The global approach requires a more complicated implementation, however it still produces results in linear time.

III. Our System Pipeline

To remove errors, our system (1) links single edges from the Edge-Buffer [23] into long strokes, (2) removes artifacts and errors from the silhouette using the MAR approach and (3) stylizes the strokes. These are described in order in the following sections.

IV. Extracting Silhouette Chains

To create silhouette chains, we employ a two-step process. First, our approach uses the edge-buffer [23] (Sec. IV-A) to extract individual silhouette edges. Once these silhouette edges are extracted, the system creates silhouette chains using the partial directed-graph information found in the Edge-Buffer (Sec. IV-CC).

A. The Edge-Buffer

The Edge-Buffer, designed by Buchanan and Sousa [23], efficiently extracts feature edges from open or closed polygonal meshes. This approach considers feature edges as silhouettes, boundaries (edges which are only connected to one polygon) and artist edges (edges specified by the user to always be drawn). To this end, the system uses a specialized data structure and a simple traversal algorithm to find all such edges in a single pass.

B. Initialization

To operate, the Edge-Buffer approach requires two data-structures: the Edge-Buffer itself and a data-structure for the polygonal mesh that stores indexed faces and vertices. Moreover, each face stores the indices of the vertices they contain. Once the polygonal mesh has been loaded and initialized in such a structure, the Edge-Buffer is initialized as an array V containing all m vertices in the mesh. For any $V[1 \cdots m]$, $V[i]$ is the linked list of all the vertices v_j adjacent to vertex v_i (Fig. 6a,b). Note that this list is sorted in increasing order of vertex numbers j starting from i, such that each pair (v_i, v_j) forms an edge. Thus there is only one entry for each i, j combination, where $i < j$. Note that an effect of i, j ordering is that higher indexed vertices have fewer links. Each node in the linked list contains the following data (Fig. 6c):

1) j, the identifier of vertex v_j adjacent to v_i. If all vertices are stored in an array, then j is the index to v_j in the array.
2) five bit fields (F, B, F_a, B_a, A). Each of these fields contains a boolean value used during run-time to extract the silhouette, boundary and artist edges quickly. Their use is explained in Sec. IV-B.1 below.

1) Run Time Operation: After initialization, feature edges are ready for extraction and display using five bit fields (F, B, F_a, B_a, A). F and B are used for silhouette and boundary extraction and F_a, B_a, and A are used to extract artist edges.

First, the bit fields FBF_aB_a are initialized to 0. Then, the system classifies every polygon in the mesh as either front or back-facing using the dot-product operation (Sec. II-B). As each polygon is checked, the bit fields FB for each of the edges that make up the polygon are updated. This can be done efficiently because each polygon has its edges cached in the polygonal mesh's data-structure. The bits are updated in the following manner. The current value of F is inverted if the polygon is front-facing. Similarly the current value in B is inverted if the polygon is back-facing. Once all polygons are tested, non-boundary edges will have been visited twice and boundary edges will have been visited once. Silhouette edges are are extracted when $FB = 11$, and boundary edges are extracted when when $FB = 10$ or $FB = 01$.

Artist edges are handled as follows: the user first initializes all artist edges by setting their A bit. The F_a and B_a bit fields are used to determine which of these edges are front-facing or back-facing respectively. As each polygon is checked, the F_a bit for each edge in the polygon is set to 1 if it is front-facing. Likewise, the B_a bit is set to 1 when the polygon is back-facing. After all edges are processed, front-facing and back-facing artist edges are extracted when $F_aB_aA = 101$ or $F_aB_aA = 011$ respectively.

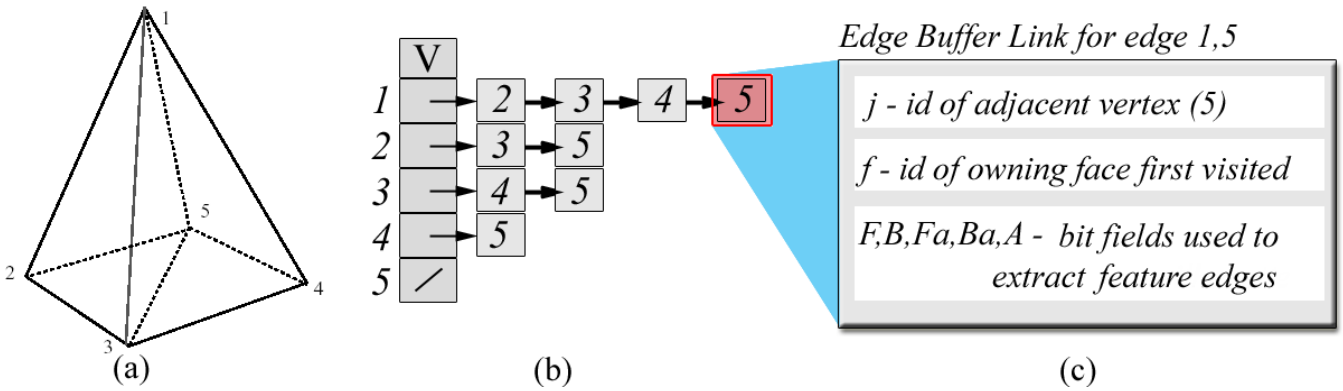

Fig. 6. A visual description of the Edge-Buffer: (*a*) a simple polygonal object, with vertex numbers listed; (*b*) an example of how edge adjacency information is stored in the Edge-Buffer [23] for this polygonal object; (*c*) the data stored in each node in the linked lists.

C. Edge-Buffer modifications

The original Edge-Buffer system [23] extracts individual edges as described in Sec. IV-A. The MAR approach requires complete silhouette chains (Sec. V), so the Edge-Buffer must be extended to create these chains from individual edges. Note that the MAR approach treats boundary, silhouette and artist edges (Sec. IV-A) equally; all types of edges extracted by the Edge-Buffer are chained together.

A two-pass algorithm is used to chain individual silhouette edges extracted by the Edge-Buffer. First, the system links all extracted edges on the model by finding the connected components of the Edge-Buffer (Fig. 7, *top-right*). The algorithm iterates through all edges (v_i, v_j) for each vertex v_i in the Edge-Buffer. Once this step finds a silhouette edge, it proceeds to vertex v_j and searches for another silhouette edge. The system continues in this fashion until it cannot find another silhouette edge. This step must know which edges are already part of a chain and which edges have not been used yet. This is accomplished by adding another bit-field, the used bit "U", to each Edge-Buffer node. Each time the view transformation changes (eg. when generating animated sequences, or when changing the view direction), U is initialized to 0. During the chaining process, any extracted edge that is used in a chain has its U bit set to 1. While creating future chains, any edge with its U bit set to 1 is excluded from chaining. Thus, each edge can only belong to one chain. Once all extracted edges have been used, this portion of the algorithm is complete. At this point the chains will only contain increasing vertex elements (Fig. 7). This is due to the directed nature of the Edge-Buffer. Thus, a second step is needed to join chains that do not have increasing vertex numbers (Fig. 7, *bottom-right*).

In this second pass, the system joins chains with matching vertex numbers on their bounds. If more than two chains can be linked, priority is given to chains that will create long loops when joined. Looping chains take precedence because the MAR approach (Sec. V) handles looping and non-looping chains slightly differently (non-looping chains are interpolated at the start and end of the chain). Thus, it is important to identify looping chains to avoid creating small new artifacts at boundaries of the chain. The MAR approach also requires a minimum chain length to function properly (the lowest is a length of 4), depending on the type of filter used (Sec. V).

This chaining method cannot guarantee the longest connected chains. Instead, it guarantees satisfactory long chains for use with the multiresolution filters.

V. Local and Global Filters

Once complete silhouette chains have been constructed from the Edge-Buffer, the MAR error removal approach is applied. Before details are provided for this, an effective review of the multiresolution techniques [15], [14] that form its base is in order.

Multiresolution methods decompose a dataset C^{k+1} into a low-resolution approximation C^k and a set of high frequency details D^k. Note that k is used in this document to specify the level of detail in the data. 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*, while 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 once.

In the functional view, C^{k+1} is the coefficient vector of high resolution scaling functions, C^k is the coefficient vector of low resolution scaling functions and D^k is coefficient vector of Wavelet functions. If C^{k+1} is a silhouette chain, C^k shows the overall sweep of the silhouette and D^k shows silhouette errors (Sec. II-E) because these are the high frequency portions of the chain.

The multiresolution operations can be specified in terms of the banded matrices A^k, B^k, P^k and Q^k. The matrix A^k transforms C^{k+1} to C^k:

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

and B^k extracts details:

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

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

$$C^{k+1} = P^k C^k + Q^k D^k \tag{3}$$

These matrices have a regular banded structure for every resolution for the looping case. In the non-looping case, the

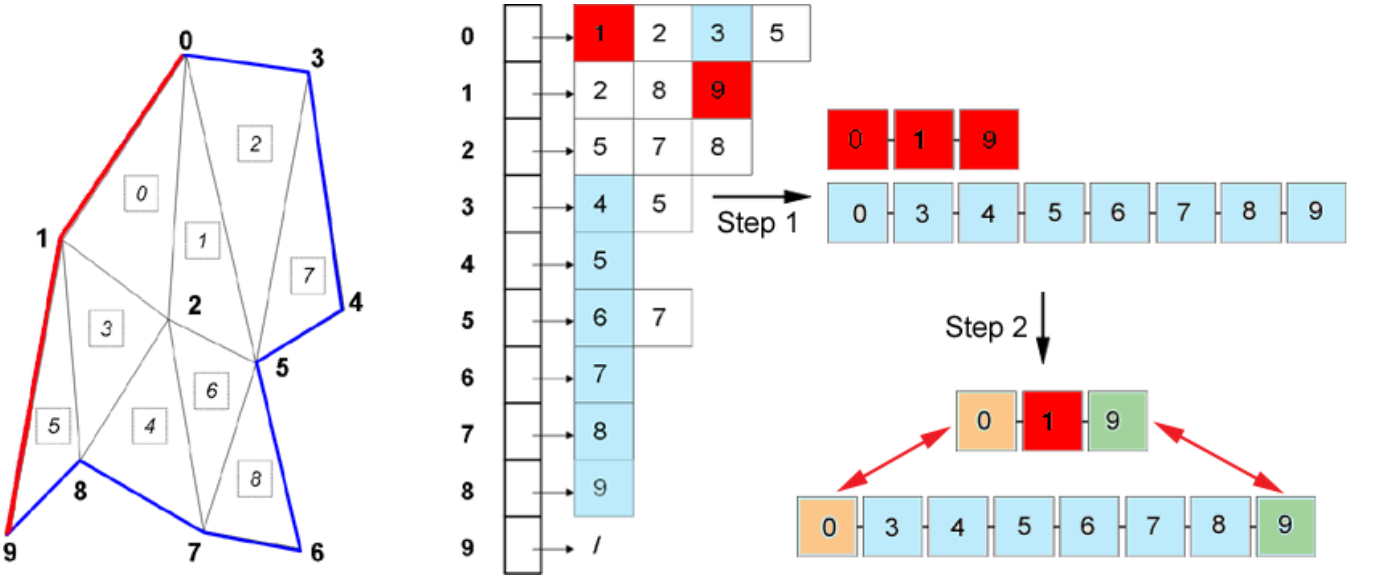

Fig. 7. The two pass process used to chain silhouette edges. *Left:* An example polygon mesh with vertex and face numbers and its associated Edge-Buffer structure. *Top-right:* The first step in silhouette chaining. In this example, two chains are created, shown with different colours on the polygonal mesh. These are extracted individually, following nodes of the Edge-Buffer until no further connections can be created. *Bottom-right:* The second step for chaining. The ends of each chain are examined for any matching vertex indices. Matches are joined with preference for creating looping chains.

matrices are regular except at the bounds of the matrix, where data is interpolated. Examples (refer to [14]) of the non-zero entries that can be used for the rows that make up the A^k and B^k matrices and the columns that make up the P^k and Q^k matrices are provided in Figs. 8-10. These matrices can be viewed as filters that operate on C^{k+1}, C^k and D^k due to their regularity. Furthermore, the only implementation difference between A^k and A^{k-1} is their size. Consequently, the superscript of matrices can be removed.

In order to find these four matrices, most multiresolution research uses wavelet-based techniques [25], [27], [26]. In the case of smooth curves, the resulting wavelets are often inadequate (see appendix of Finkelstein and Salesin [25] or page 94 of Stollnitz et al. [27]). For the MAR approach, C^{k+1} is a discrete approximation of a silhouette curve and the only requirement is use of the appropriate appropriate A, B, P and Q filters. Therefore, a discrete approach of multiresolution systems that directly operates on discrete data is fitted here more effectively. Samavati and Bartels [14], [15] provide this kind of multiresolution based on reversing subdivision. They also demonstrate that their results are more effective for discrete data sets than conventional wavelets. In the MAR approach, their multiresolution filters constructed based on reversing *cubic B-Spline* subdivision, *Chaikin* subdivision and *Dyn-Levin interpolation* subdivision are used. The bands that make up the A, B, P and Q filters are provided for these systems in Figs. 8, 9 and 10.

For implementation, A and B are applied to C^{k+1} to obtain C^k and D^k. Again by applying P and Q filters on C^k and D^k (or, as is done in the next section to remove errors, a modified version of D^k), C^{k+1} can be reconstructed. Recall that these filters produce an optimal solution intrinsically without any extra work in implementation, they usually require no extra space and they provide linear-time operations. For

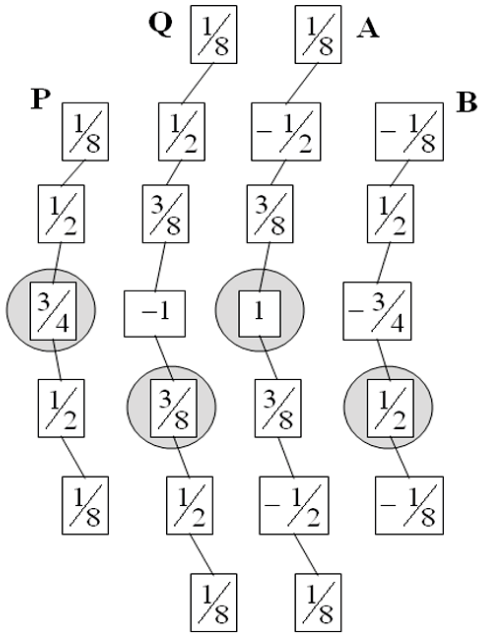

Fig. 8. The bands of the filters for the cubic B-Spline case (the A, B, P and Q diagrams represent all non-zero entities of a row for the *A* and *B* matrices and of a column for the *P* and *Q* matrices). The gray circles show the center entity.

the local approach, A, B, P and Q are all regular banded matrices. In the global approach, A and B are full matrices. Nevertheless, they still have the regular structure. In order to achieve linear time operations, the MAR approach solves the following banded system for global decomposition [15]:

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

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

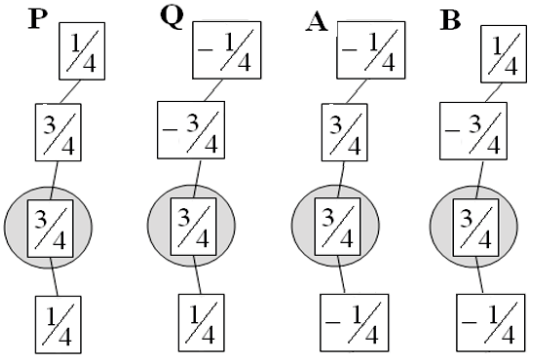

Fig. 9. The bands of the Chaikin filters.

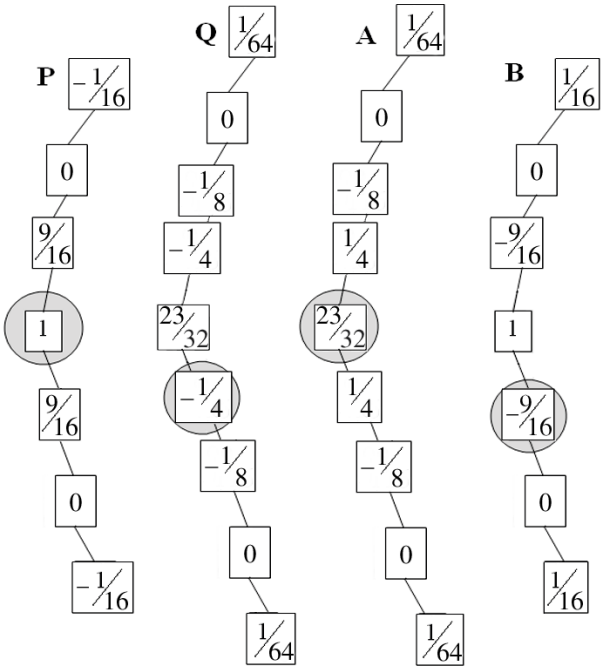

Fig. 10. The bands of the Dyn-Levin interpolation filters.

In a recent work, Bartels et al. [29] demonstrate that the local approximation is a good estimate of the global approximation. For silhouette error removal, experiments comparing local and global multiresolution show that for low resolution meshes, global multiresolution is required to create accurate strokes (Sec. VII). The drawback of using global multiresolution is the need of solving the systems in Eqs. 4 and 5.

A. Other Multiresolution Approaches

The reverse subdivision multiresolution filters of Samavati and Bartels [15], [14] were chosen over that of Finkelstein and Salesin [25] and Stollnitz et al. [27] for three primary reasons. First, reverse-subdivision multiresolution has been demonstrated to be more effective for discreet data than wavelet-based approaches [15], [14]. Second, Samavati and Bartels provide various types of filters with a "local" or a "global" scope while Finkelstein and Salesin and Stollnitz et al. only provide filters for a wavelet-based cubic B-Spline multiresolution. This provides variety, to properly remove errors and stylize silhouettes, and efficiency, as the approaches provided by Samavati and Bartels are much faster than the wavelet-based approach (see Brunn et al. [30]). The third reason that reverse subdivision multiresolution was chosen over the wavelet-based approach is because the masks it uses are simpler. The masks that make up wavelet-based filters have a large width which contain complicated rational numbers compared to a narrow width with simple fractions used by Samavati and Bartels. Simple fractions are preferred to minimize rounding errors. Also, the narrower mask width of the reverse-subdivision filters further increases computational efficiency.

B. Error removal pipeline

In this section, details are provided on how the multiresolution techniques presented by Samavati and Bartels [15], [14] are used in the Multiresolution Artifact Removal (MAR) approach. MAR's default silhouette error removal pipeline decomposes silhouettes in two steps and reconstructs to the original level of detail with a scaled-down version of the high-frequency details to remove errors (Fig. 11). Note that the user can change the number of times that the silhouette is decomposed as large errors may require three steps of decomposition and small errors may only require one step. A discussion of this is provided in Sec. VII.

As shown in the previous section, normal reconstruction with coarse information and high-frequency details returns the coarse data to its exact original form. Scaling down the amount of details means that the high-frequency data will be lessened in the silhouette, resulting in removal of errors. The MAR system modifies 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 detail data added to coarse data. The higher the value of e, the closer the stroke gets to the original data extracted.

Raw silhouette chains from the Edge-Buffer can be thought of as a low frequency "correct" path plus high frequency errors (Fig. 5). 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.

The error removal effect of the MAR approach is illustrated in Fig. 12. In this image, the original silhouettes have been decomposed and reconstructed once with global cubic B-Spline filters. In the leftmost image in this figure, 100% of the details are included in reconstruction ($e = 1.0$) resulting in the exact original silhouette being regenerated. Moving right in Fig. 12, fewer and fewer details are included, until the rightmost image, where 0% of the details are included ($e = 0.0$). Note in the leftmost image that the jagged movement of the silhouette on the beaver's back has created some minor artifacts. As details are removed, these high frequency "zig-zags" are scaled down while the low frequency, correct path of the silhouette is maintained.

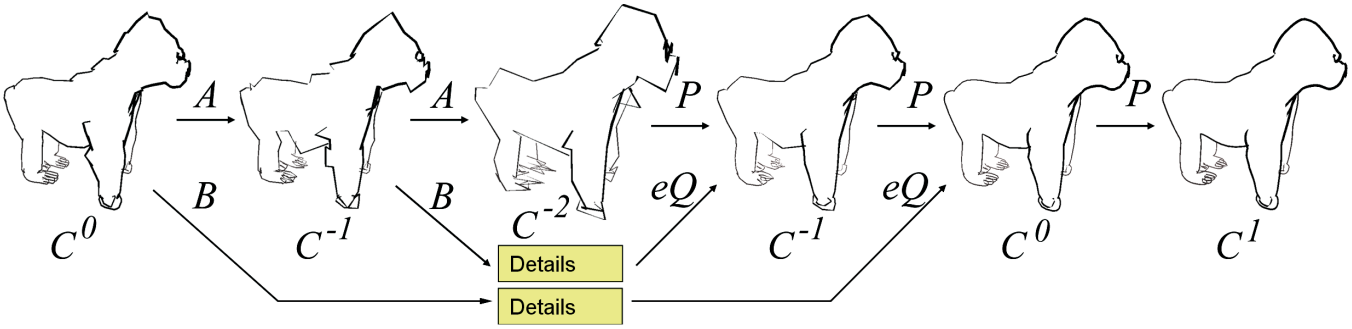

Fig. 11. The MAR approach uses multiresolution filters to decompose and reconstruct silhouette chains without errors. Here is an example session error removal session for an **ape** mesh with 7434 faces. Proceeding from left to right, the silhouette chains are first decomposed twice from level C^0 to C^{-2}. Then, the system reconstructs to level C^0 using scaled details (here, $e = 0.3$). The effect of this process is the removal of errors. Finally, the MAR approach reconstructs past the original level of detail to C^1 to provide a smoother more stylized silhouette.

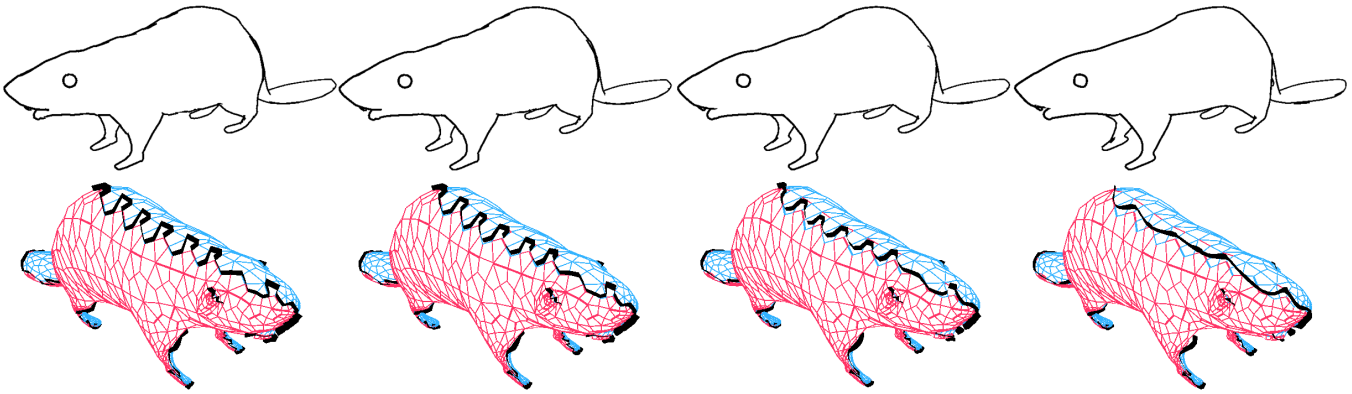

Fig. 12. The effect of removing details. In this image, silhouettes from a **beaver** model are shown for two angles, head-on in the top row and from behind with mesh information on the bottom row. One level of decomposition and reconstruction is used here with global cubic B-Spline filters. From left to right, $e = 1.0$, $e = 0.66$, $e = 0.33$ and $e = 0.0$.

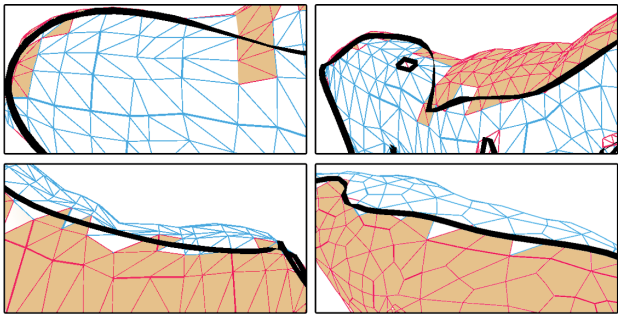

Fig. 13. Strokes generated by the MAR system for the input in Fig. 5.

In Fig. 13, the MAR approach has been applied to the silhouettes illustrated in Fig. 5. Note that the high-frequency noise has been removed and that the accuracy of the stroke has been maintained.

The images provided in this work demonstrate that the output of this system is suitable for pen-and-ink silhouette illustration. Values from 0.0 to 0.4 are used for e, depending on the detail in the original mesh used. A discussion of the performance of this system is provided in Sec. VII.

C. Chain Size

The silhouette chains sometimes need to be modified before applying MR filters. In the decomposition process, a dataset C^{k+1} is decomposed into a coarse approximation C^k and details D^k. If the length of C^{k+1} is n, then the length of C^k will be $n/2$ when the chain loops or $2 + n/2$ otherwise. Since the length of the chain must be a whole number, C^{k+1} must have a length divisible by two for decomposition. The MAR approach handles this by adding a single point to any chain of odd length before performing any level of decomposition. This point is added at the second last position in the chain, interpolated between its neighboring points. During reconstruction, the system removes these extra points once they are reconstructed.

D. Resolution-Independent Strokes: Smoothing/Coarsening

In the MAR approach, reconstruction can proceed to a higher level-of-detail than the original chain to smooth the silhouette strokes or can stay at a low level-of-detail to coarsen the strokes. For the higher levels of detail, since there are no details D^k associated with the strokes, this is accomplished simply by eliminating QD^k in Equation 6 to create:

$$C^{k+1} = PC^k \tag{7}$$

The P filter is actually a subdivision matrix. Thus, using P alone increases the smoothness of C^k. This is useful when the input mesh has a low number of triangles or when one wants to view the silhouettes from a larger mesh closely and rough

edges are not desired in the final output. This is illustrated for a mesh with a lower number of triangles in Fig. 14.

Fig. 14. The rightmost two **ape** images from Fig. 11. Here the system performs reconstruction on corrected strokes above the original level of detail. This results in a smoothing effect, making the geometry of the underlying mesh less apparent.

With the MAR approach, the user has control over the number 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. A discussion of the results from using these different approaches is provided in Sec. VII.

VI. Stylizing

The MAR approach uses two steps to create appealing strokes that approximate real hand-drawn pen-and-ink illustrations. First, silhouette chains that appear too coarse can automatically be smoothed using the resolution-independent chains (Sec. V-D). The second step to stylize the strokes is to use the angled-bisector strip presented by Northrup and Markosian [10]. This method converts the silhouette chains into triangle strips which simulate pen strokes that vary in width (controlled by the distance from the eye to each point in the stroke). Points in the chain closer to the eye are stylized as wide portions of the stroke while points farther away produce narrower portions.

The MAR approach provides two options for Hidden Line Removal (HLR). The first method works in image-space and is very efficient, however it fails when used with silhouettes from coarse meshes. The second method uses an image-space technique to produce more accurate results. Both methods are described in the next two sections.

A. Object Space HLR

For object-space HLR, the method from the original Edge-Buffer system [23] is used. This method first renders the polygonal mesh in white and then draws the silhouettes in black. Thus, any strokes on the back-facing side of the surface will be occluded by the white mesh. To ensure that the mesh does not partially occlude edges on the front side of the surface, the silhouette edges are displaced away from the mesh slightly.

While this approach is acceptable for the Edge-Buffer system, it does not always work with the strokes generated by the MAR approach because these processed strokes do not adhere exactly to the mesh (see discussion in Sec. VII). Thus, strokes that should be visible may be moved slightly behind the mesh and thus be improperly occluded and strokes that should be invisible may be moved enough so that they are seen. This effect is negligible for dense meshes, but is increasingly noticeable for coarser meshes (Sec. VII). This is why portions of the silhouette are missing in Figs. 15 (middle) and 19. Some strokes from the local filters in Fig. 21 (*bottom-left*) are also improperly handled at the cat's paws and ear.

A simple solution for this problem is to displace the strokes slightly towards the eye. This approach has been used for all of the result images in Sec. VII. Unfortunately, depending on the geometry of the mesh, this approach might not work for all parts of the stroke (Figs. 15(middle), 19, 21). Specifically, if multiple silhouettes exists at very close depths, this method may incorrectly display occluded silhouettes. Furthermore, if the mesh has many sharp features, this might not reveal the complete stroke in certain places. In these situations, a stronger form of HLR is required, as is presented in the next section.

B. Image Space HLR

Hidden line removal can also be accomplished with an image-space approach. In this approach, the assentation is that if a raw, unprocessed edge[2] is visible, then its corresponding processed edge (or edges, if the processed silhouette chain is at a higher level of detail than the unprocessed chain) is visible. To determine if an unprocessed silhouette edge is visible, an *ID buffer* is used [10]. **Unprocessed** silhouette edges are drawn in unique colours in the ID buffer along with a white version of the mesh to perform occlusion (Fig. 15, left). Unique colours are used so that a list of visible unprocessed edges can be created by analyzing each pixel of the ID buffer. Once the system has built this list, it draws the following processed edges generated by the MAR approach:

- processed edges whose corresponding unprocessed edge was found to be visible in the ID buffer
- r non-visible edges between two visible edges; where r is related to the size of the errors; In the MAR approach, $r = 2$ is used.

The extra r edges are drawn because the MAR approach changes vertex positions in the chain; therefore, small groups of invisible edges before processing, usually those found at error positions, will become visible after processing. These must be drawn so that small breaks do not appear in the stroke.

This method provides more accuracy than the object-space approach, however it is computationally more expensive. The extra computation time required is directly linked to the number of pixels that must be analyzed. This step takes on average an extra 90 milliseconds for an 800 by 800 pixel display.

VII. Results and discussion

The MAR approach can generate error free strokes with minor user input for most meshes. Results generated by the

[2]An unprocessed edge, is an unmodified silhouette edge extracted from the polygonal mesh before the MAR approach has been applied.

Fig. 15. *Top:* raw silhouette edges, each with a unique colour. *Middle:* results of object-space HLR on processed strokes. *Bottom:* results of image-space HLR on the processed strokes.

system are illustrated in Figs. 17- 23. The MAR process achieves fast computation rates including preprocessing (building the Edge-Buffer) and rendering (chaining, multiresolution filtering, and stroke stylization). Furthermore, the multiresolution methods employed [14], [15] operate quickly and can produce resolution-independent silhouettes.

MAR is more suitable for finer, denser meshes as it might remove important detail from meshes with a low polygon count. For these coarse meshes, MAR presents a tradeoff between feature-preservation and quality of filtering (directly controlled by the value e). It can sometimes be impossible to remove errors from silhouettes of simple meshes without losing stroke accuracy (Fig. 19). A solution to this is to subdivide these meshes using a method such as Catmull-Clark or Doo-Sabin subdivision before extracting and correcting silhouettes.

Running times are provided in Table III for various polygonal meshes for the local and global multiresolution approaches, following with a discussion of the quality of the results with notes on mesh size, user input, the global and local approaches and the different filter types. Finally, the MAR approach is compared to other error removal approaches [13], [12], [10], [8].

Model	Fig.	Polygons	Num. Chains	Chain Length	Local Time	Global Time
Ox	19	652	11.4	16.3	0.414	1.55
Ape	13	1490	35.0	24.8	0.742	7.17
Beaver	12	2286	13.8	39.1	0.613	3.669
Face	22	2940	24.5	23.8	0.922	7.01
Kleopatra	23	4092	8.6	38.9	0.437	3.207
Cat	21	7819	41.7	27.9	2.241	39.84
Toutalis	17	12796	30.4	20.1	1.065	13.671
Inner ear	18	32702	99.4	27.1	9.589	155.73
Foot	20	46045	180.6	29	60.007	77.195

TABLE III
RUNNING TIMES (IN MILLISECONDS) FOR ERROR-CORRECTION FOR THE MESHES ILLUSTRATED IN THIS ARTICLE. THESE RESULTS ARE PLOTTED IN FIG. 16.

A. Timing

Table III provides running times for the system, when two levels of decomposition and reconstruction for local and global cubic B-Spline filters are used. The provided computation times were gathered using a 2.65 GHz Pentium 4 with OpenGL/ATI Radeon 9700 graphics and 1 gigabyte of RAM running Windows XP.

These results are averaged from 256 tests with silhouette chains extracted at random viewing directions and are plotted in Fig. 16. These results illustrate that the MAR approach is efficient; meshes less than about 20000 faces usually run in real time and larger meshes, such as the foot (Fig. 20), run at interactive speeds. The speed of the multiresolution filters is determined by the number and size of the silhouettes extracted.

These results also reveal that the global approach takes more time to operate than the local approach. This is because global multiresolution requires solutions to Equations 4 and 5. Fortunately, the results for the global approach are still realtime or interactive for the small to medium-sized meshes of sizes up to 30,000 triangles displayed in Table III. The added accuracy of global methods is not required for high resolution meshes (of size greater than about 10,000 faces) because the strokes from large meshes adhere well to model. This is due to the higher resolution and smaller average error size in the silhouette chains extracted from these meshes.

B. User Input

Meshes with more than about 10000 faces require little or no user-input (Figs. 17, 18 and 20). For these meshes, error free strokes with no accuracy loss can often 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 employed $e <= 0.1$ meshes larger than 10000 faces. To generate accurate strokes for smaller meshes (Figs. 11, 19) or to accommodate small sharp features on larger meshes (those only defined by several triangles), the global multiresolution approach must be used (see next

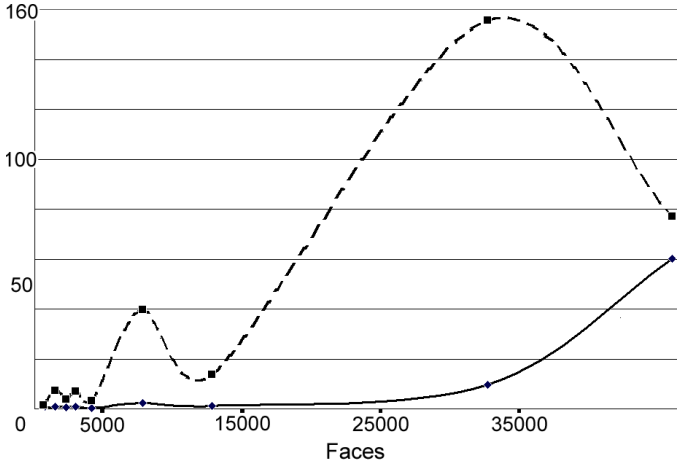

Fig. 16. A plot of the MAR approach's timings versus the number of faces in the polygonal meshes provided in Table III. The solid line represents the average execution time for the local multiresolution approach. The dotted line represents the average execution time for the global approach. The expense of the global approach is always higher than the local approach, but appears to be highly variable with respect to the length and number of silhouettes from the mesh.

section for a detailed discussion) with precise input for the amount of details and the number of decomposition and reconstruction steps. It is in these situations that varying e can result in a noticeable tradeoff between error removal and feature preservation. Accurate strokes can be generated for smaller meshes if the mesh is subdivided before extracting silhouettes; however, this requires a subdivision preprocess and will not generate an accurate error-free silhouette for the original mesh—the silhouette of the subdivided mesh will be corrected.

C. Global Multiresolution VS. Local Multiresolution

The cubic B-Spline, Dyn-Levin and Chaikin filter matrices (Figs. 8-10) have been tested with local and global multiresolution methods.

Visually, the global method produces more accurate results, an effect most noticeable for meshes with a smaller number of faces and edges. Global results are directly compared to local results in Figs. 21 and 23. Note in Fig. 21 that although the local method (Fig. 21 *bottom-left*) appears to solve most of the errors highlighted in the raw image (Fig. 21 *top*), it loses accuracy in several areas, notable especially around the cat's front paws and ear. This image was rendered using the object-space HLR approach and the loss of accuracy is the reason that some strokes improperly hidden and revealed. The global approach (Fig. 21 *bottom-right*) maintains accuracy, removes all identified errors and is accurate enough so that the object-space HLR approach handles all parts of the silhouette curves properly. In Fig. 23, local and global silhouettes for all three filters are shown at an alternate angle. In these images, the underlying mesh is also rendered with shaded back-facing polygons to reveal how accurate the processed silhouettes are. Also, Fig. 23 displays local results (left-column) and global results (right-column). In the cubic B-Spline local example (top-left), the processed stroke loses the mesh significantly (most noticeable at the middle part of the silhouette). The cubic B-Spline global approach does not suffer from this loss of accuracy.

As presented in Sec. VII-A, the expense of global methods increases with the size and number of silhouettes. Fortunately, the visual accuracy improvement is usually only useful for smaller sized meshes (Figs. 11, 13, 12, 19, 21) where the solution to the systems used in the global approach can be found quickly.

In this article, we employed local methods for Figs. 13 (*left column*), 17, 18, 20, and 21 (*bottom-left*). We employed global methods for Figs. 11, 13 (*right column*), 12, 19 and 21 (*bottom-right*). Figs. 22 and 23 illustrate the results of executing both the local and global multiresolution methods for each filter implemented in the system.

D. Cubic B-Spline Subdivision VS. Dyn-Levin Interpolation VS. Chaikin Subdivision

There are three different sets of multiresolution filters [14], [15] implemented in the MAR approach (providing the A,B,P,Q matrices, Sec. V): cubic B-Spline subdivision, Dyn-Levin interpolation and Chaikin subdivision. Each of the filters produce slightly different results when used to remove errors from the silhouette curves. For large meshes with dense details, the different effects of the filters are not particularly apparent because the starting silhouette is very detailed and the errors are not large compared to the correct detail in the chain. Thus, it is difficult to see regions where the filters produce different results for detailed meshes, unless about three or more levels of decomposition are used. It is also difficult to classify the results of the filters for coarse meshes because, various input data can produce drastically different results. Despite this, it is important to understand the general differences between the filters to choose which filter to try first when correcting errors in silhouettes from coarse meshes and for large meshes when viewed closely.

The cubic B-Spline filters are based on a subdivision scheme and the points generated are C^2 continuous curves. This is the highest continuity of the three methods and makes appealing smoothed strokes. This continuity comes from the B-Spline filters' larger mask width. This larger width means that errors are removed quickly. However, this also means that the cubic B-Spline approach is more prone to inaccuracy over the other filters because it is a subdivision approach and the mask uses more influence from neighboring points (observe this in the local case for cubic B-Spline in Fig. 23). Thus slightly higher values for e might be required with these filters. Fig. 22 is provided to illustrate this effect. In this image, strokes generated with the cubic B-Spline filters (left column) give the best impression of the face. The other two methods generate less appealing bumpy results, most noticeable in the error-filled temple areas.

The Dyn-Levin filters are created from a different subdivision method which is based on interpolation. Thus, strokes processed with this method adhere better to the original mesh than the B-Spline based methods. Although a larger mask width (see Samavati and Bartels [14]) allows the Dyn-Levin

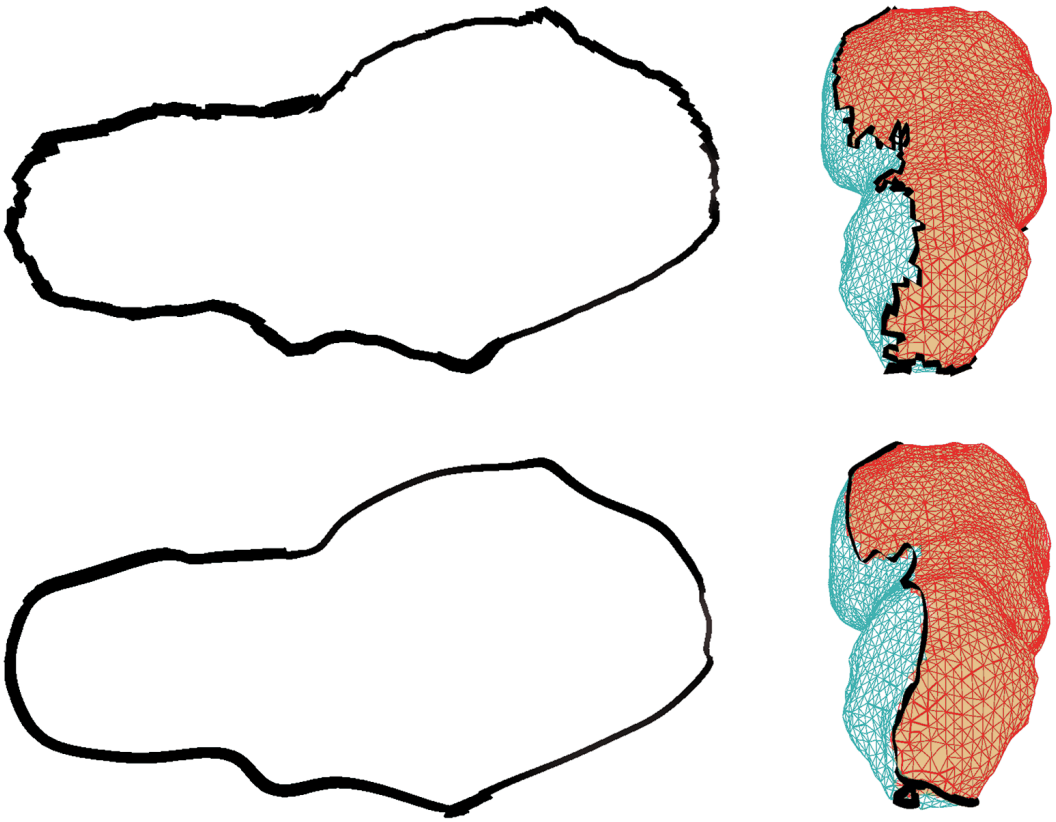

Fig. 17. *Top*: Original silhouette from a model of the **Toutalis asteroid**. *Bottom*: Results after processing with the MAR system. In this session, two levels of global cubic B-Spline decomposition and reconstruction with $e = 0.1$ were used. Shaded polygons are back-facing.

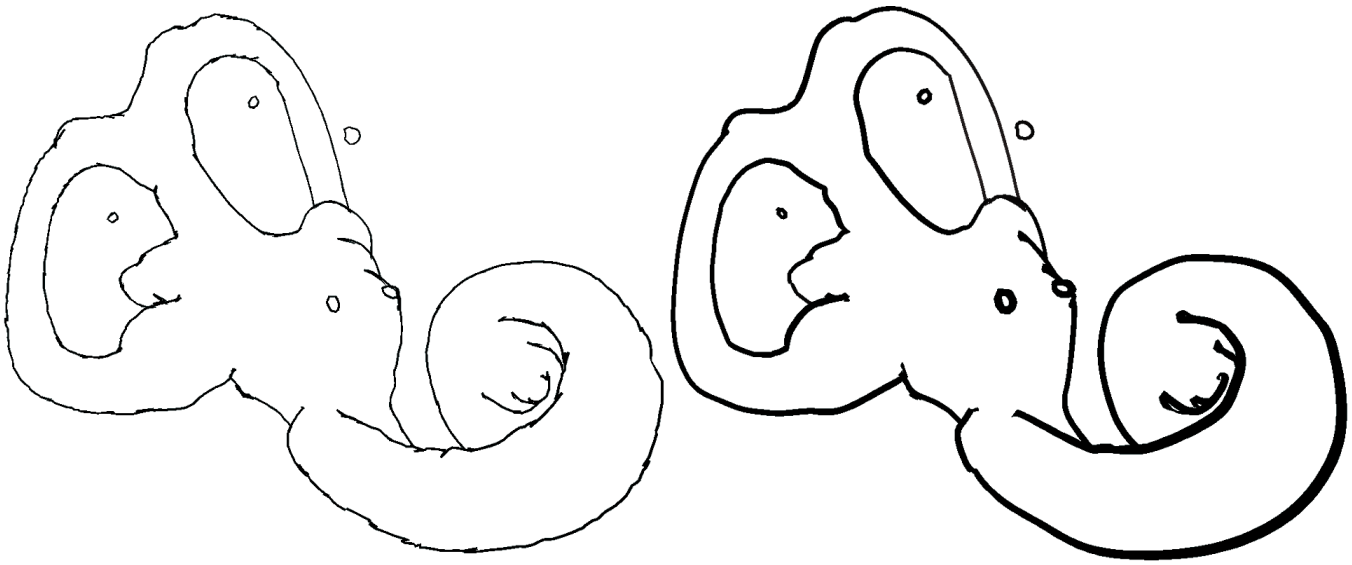

Fig. 18. *Top*: Original silhouettes from an **inner-ear** model. *Bottom*: the results after processing two levels of decomposition and reconstruction with local cubic B-Spline filters and $e = 0.0$. Stroke thickness varies in this image as a function of depth.

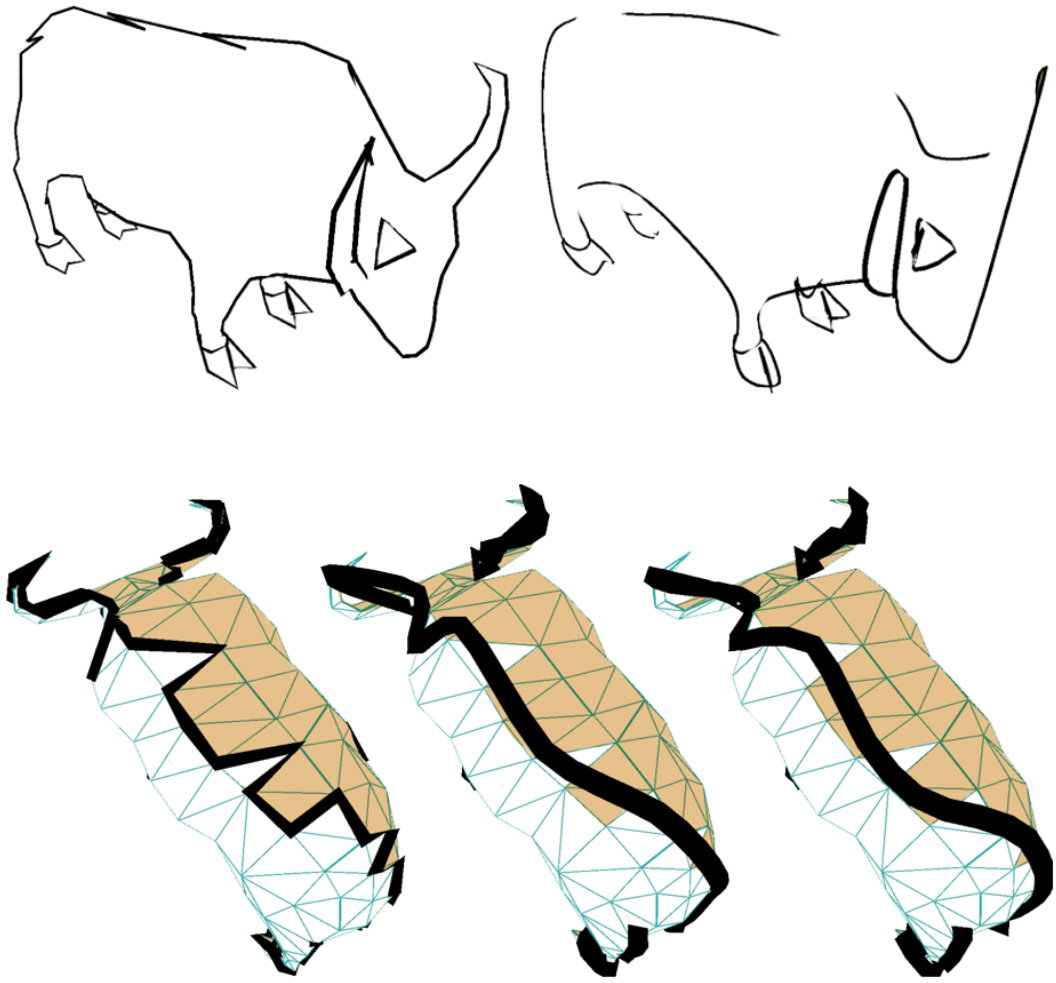

Fig. 19. *Top-Left*: raw silhouette from a coarse **ox** mesh. *Top-Right*: processed strokes with global cubic B-Spline filters. *Bottom row*: alternate views of the silhouette with the original strokes (*left*), processed strokes with global cubic B-Spline filters (*middle*) and global Chaikin filters (*right*). In both cases, two levels of decomposition and reconstruction are used with $e = 0.35$. This is an example of the system producing poor accurate output: the corrected strokes do not adhere well to the original mesh due to the low resolution of the initial silhouettes.

approach to remove errors with about the same settings as the cubic B-Spline approach, this method can exaggerate some features as a side effect of the interpolation. Observe this exaggeration in the middle column in Fig. 22, where in the local case, the filter has pointed the head slightly in one direction and in the global case, the filter has distorted the right temple slightly. These filters should be used when accuracy of strokes is desired over quality of the final image.

The Chaikin subdivision filters are the fastest to execute due to the fact that their masks are the narrowest [14]. The Chaikin filters provide a quadratic B-Spline subdivision which offers C^1 continuity. The result of using these filters on silhouette chains is somewhere between use of cubic B-Spline subdivision and Dyn-Levin. The processed strokes adhere better to the mesh than with cubic B-Spline, but not as accurately with as Dyn-Levin. This is visible in Fig. 23 (left-column). It is important to note that in order to remove errors with the Chaikin filters, fewer details and sometimes an additional step of decomposition and reconstruction must be used in the MAR pipeline. Thus, after removing errors, strokes processed with Chaikin filters usually adhere *less* accurately to the mesh than with cubic B-Spline. These problems are due to the limited scope of the Chaikin mask. In Fig. 22, all strokes are processed with one level of decomposition and $e = 0.2$. The strokes processed using the Chaikin filters (right-column) curve inwards and outwards strangely, still following the implying some of the "zig-zags" in the silhouette (this is most noticeable in the temples). Due to these problems, these filters are not recommended for use with this system when accuracy becomes important for low quality meshes.

For smaller meshes where accuracy becomes an issue, the cubic B-Spline filters should be used for the best looking strokes and the Dyn-Levin filters should be used for the most accurate strokes. When larger meshes are used, cubic B-Spline is recommended, unless fast results are required. In this case, the Chaikin filters should be used because its smaller mask width lowers execution times.

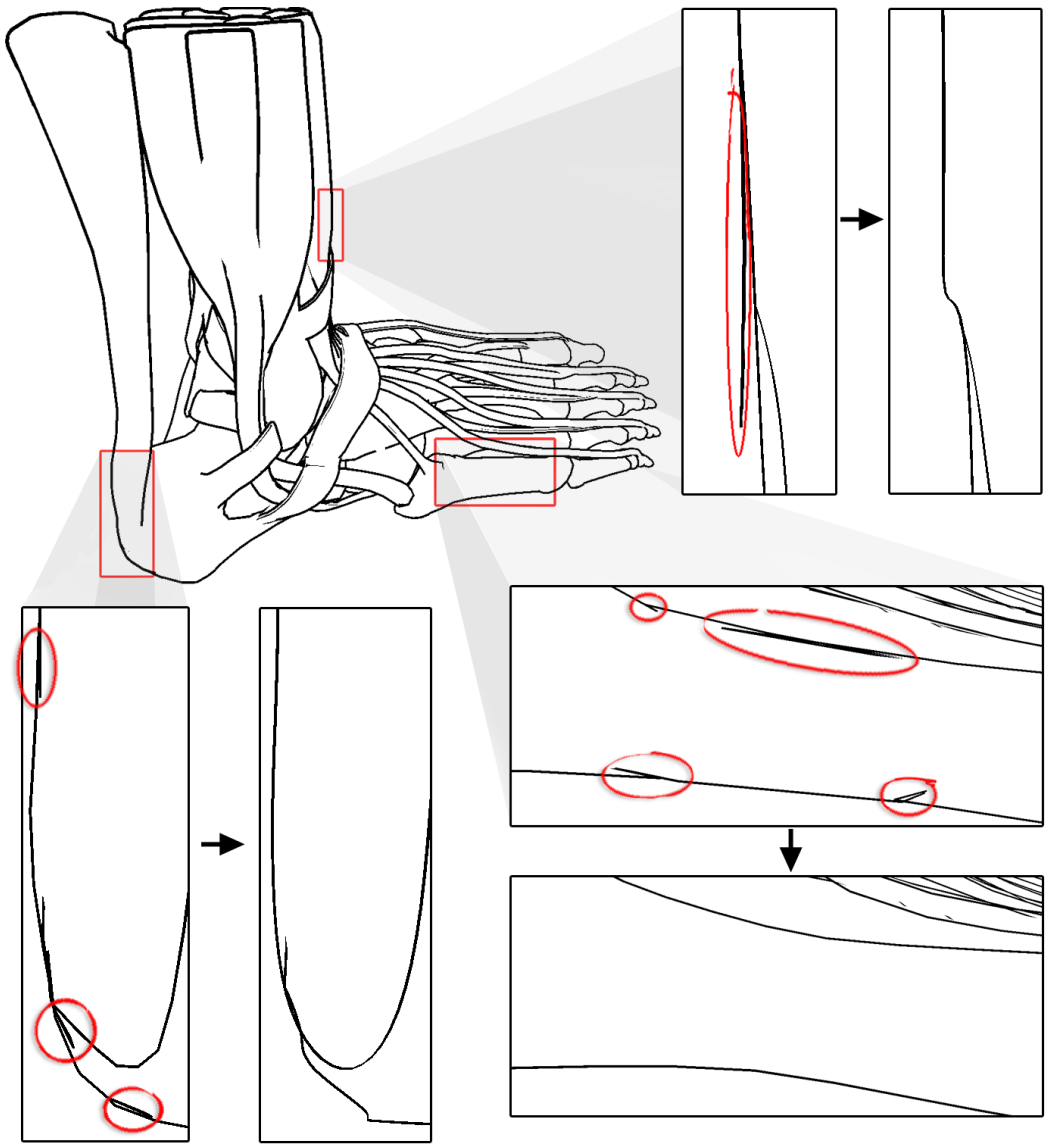

Fig. 20. A detailed **foot** mesh. Removing silhouette errors on large meshes is more important when zooming in on the mesh. Errors are circled for three enlarged areas. These images use two levels of decomposition and reconstruction, $e = 0.2$ and local cubic B-Spline filters.

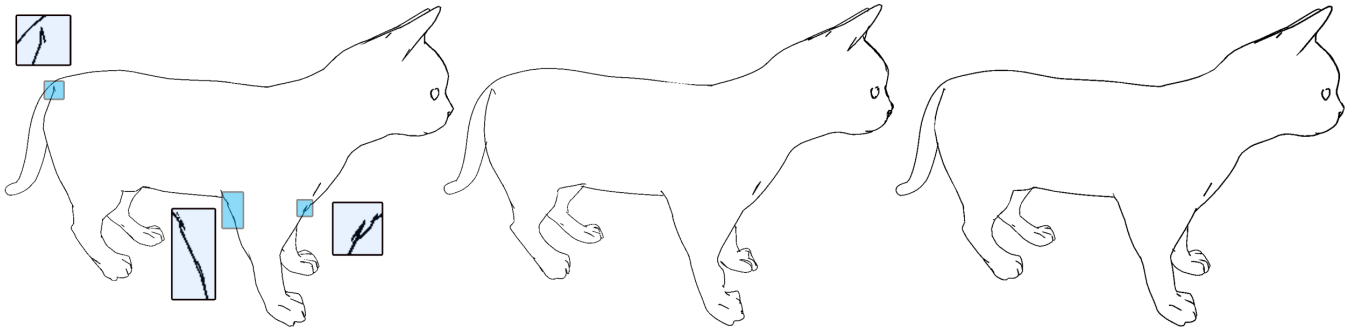

Fig. 21. *Top*: Raw silhouettes extracted from a **cat** mesh. *Bottom-left*: local B-Spline subdivision with two steps of decomposition and reconstruction 30% details included. *Bottom-right*: global B-Spline with the same settings.

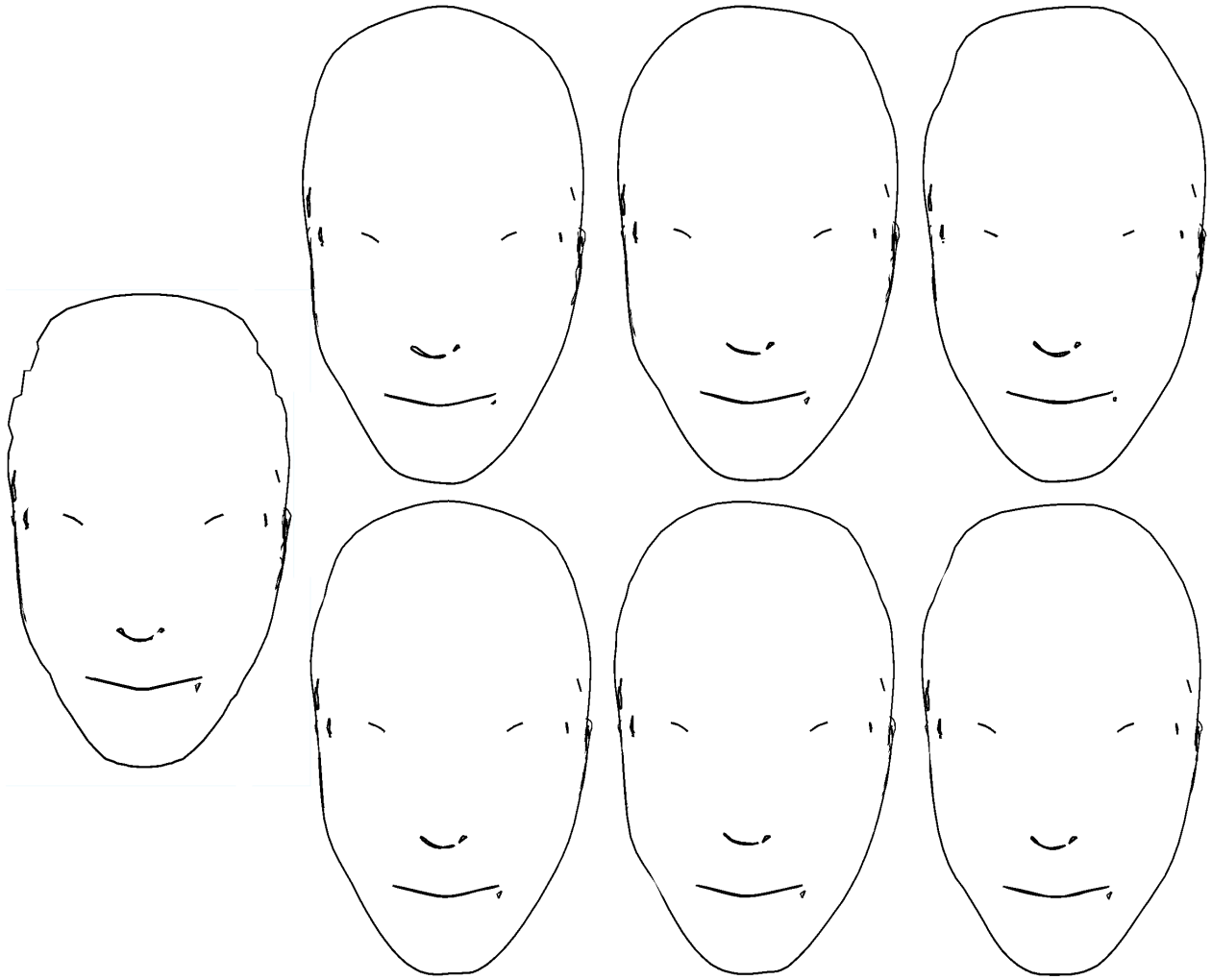

Fig. 22. *Left*: Silhouette from a **head** mesh with several large errors at the temples and six results of running, from left to right, B-Spline, Dyn-Levin and Chaikin filters on the silhouettes using one level of decomposition and reconstruction with 20% details included. The top row uses local filters and the bottom row uses global filters.

E. Comparison to Previous Work

As detailed in Sec. II-E, four methods have already been presented for silhouette error correction. Of these methods, there are two main approaches. Northrup and Markosian [10] and Isenberg et al. [8] use the approach of correcting raw silhouette edges extracted from the polygonal mesh. Corrêa et al. [13] and Hertzmann and Zorin [12] ignore raw silhouette edges from the mesh and to generate new, better edges to approximate the silhouette.

The MAR approach, like Isenberg et al. [8] and Northrup and Markosian [10], corrects silhouette edges directly from the mesh. It offers an improvement over their approaches because it provides a general solution. In other words, since the method is evaluated evenly over the complete chain, no errors are missed. Isenberg et al. [8] and Northrup and Markosian [10] require a series of error-cases and corresponding solutions which occasionally miss errors. Another important distinction between the systems is that the MAR approach corrects errors in all strokes (Fig. 24, middle), while Northrup and Markosian's [10] method and Isenberg et al. [8] only correct visible strokes. This is useful when some sort of transparency stylization is desired, however it can mean processing many extra silhouettes for noisy meshes, such as those produced by range-scans (Fig. 24). A significant drawback of the MAR approach to Isenberg et al. [8] and Northrup and Markosian [10] is that it cannot generate accurate strokes for coarse meshes (Sec. VII-B, Figs. 21, 19), while their approaches are not affected by the detail of the mesh.

Hertzmann and Zorin [12] and Corrêa et al. [13] provide techniques that generate new, more suitable edges for silhouettes. Hertzmann and Zorin [12] provide a more general version of these two methods which will now be discussed. In their approach, silhouettes are generated by estimating the exact position on the polygonal mesh that the silhouette would intersect if the polygonal mesh were a smooth surface. Both the MAR approach and Hertzmann and Zorin's approach generate sub-polygon silhouettes close to the "actual" silhouette for the polygonal mesh (if it was a smooth surface) and do not miss individual errors. Furthermore, both approaches have problems with coarse meshes. Hertzmann and Zorin's approach experiences problems performing hidden line removal for these meshes because their interpolated strokes

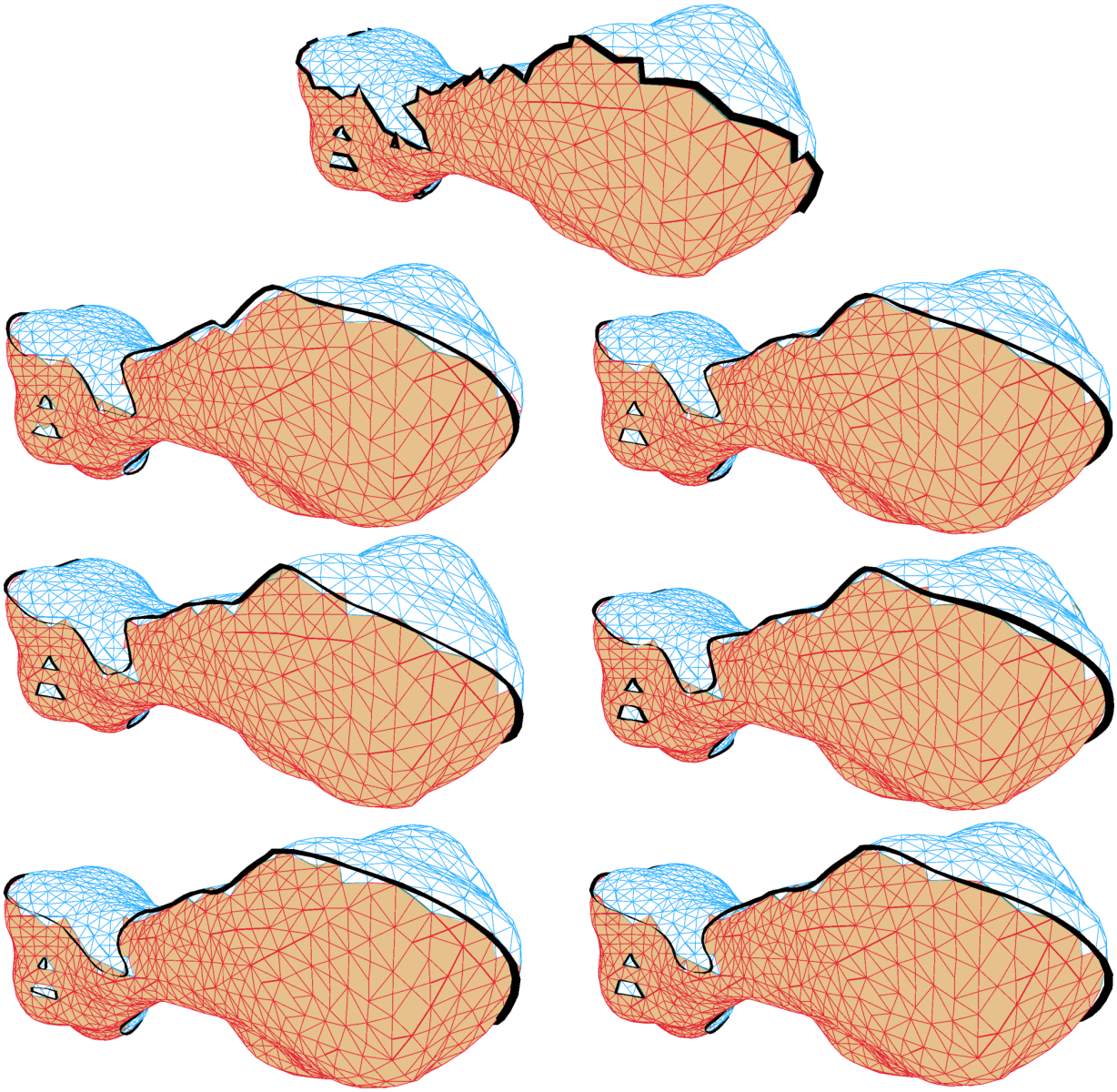

Fig. 23. An analysis of the effects of various filters viewed from an angle alternate to that used to extract silhouettes from the **Kleopatra asteroid** model. *Top*: The raw silhouettes. Shaded faces are back facing. *Left-column, from top to bottom*: Local B-Spline, Dyn-Levin and Chaikin approaches. *Right-column*: Global filters in the same order. In all images, $e = 0.0$ is used with two levels of decomposition and reconstruction.

might go over a back face and become invisible due to z-buffer occlusion. In the case of the MAR approach, issues of stroke accuracy arise for coarse meshes (Sec. VII-B, Figs. 21, 19). The primary difference between these two approaches is that Hertzmann and Zorin's approach generates edges exactly on the mesh while the MAR approach smoothes edges to various levels of accuracy controlled by the user.

The following scheme should be used to choose between the approaches. If completely accurate corrected silhouettes are desired, Hertzmann and Zorin's [12] approach should be used because it is guaranteed to produce accurate error-corrected chains. If resolution independent strokes smoothed independent of mesh geometry are desired, the MAR approach should be used. The only case where this does not hold is for very coarse meshes where the approaches presented by Isenberg et al. [8] or Northrup and Markosian [10] will produce the best results.

VIII. Conclusions and Future Work

The MAR approach efficiently computes 3D stylized, smooth, error-free silhouettes in a pen-and-ink style. Users can provide input to the system to determine the type of silhouette strokes generated. A comparison between real artwork in this style and the results of the system are displayed in Fig. 25. Our approach represents an improvement over previous works because:

- it provides resolution-independent silhouettes not bound to the geometry of the mesh. This means the silhouettes can be smoothed or coarsened to a different resolution from the raw silhouette and provide a more realistic

pen-and-ink style. Smoothing is important to create natural looking silhouettes and is critical for a realistic effect when one views silhouettes from detailed meshes closely, or when one views silhouettes of simple meshes. Coarsening is useful to create simpler strokes from very complicated meshes, benefiting systems such as Kirsanov et al. [26];

- it does not require specialized error/solution cases to remove errors and thus provides a more general solution than previous techniques.
- it generates sub-polygon strokes (closer to the real location of the silhouette) for arbitrary meshes efficiently.

A. Limitations and Future Work

The MAR approach has several limitations which present opportunities for future research. First, stroke accuracy can be lost with coarse meshes. In these cases, only artistic smooth strokes can be generated. Processes to maintain accuracy should be explored. Perhaps an approach that varies the amount of detail included along the chain during reconstruction could be applied to this problem. Although subdividing the mesh before extracting and correcting silhouettes solves this problem, this solution only provides a modified smooth silhouette (not the exact silhouette for the original mesh). Another limitation is the lack of a robust and efficient Hidden Line Removal approach. New methods for HLR must be investigated for the MAR approach. Finally, a limitation of all object-space methods is that they do not provide a way to eliminate duplicate silhouette chains that occur in noisy meshes, such as those produced by range-scans (Fig. 24).

A formal evaluation has not yet been performed for the MAR approach. Results have been evaluated:

1) in terms of performance, by comparing to previous methods and evaluating execution times (Sec. VII-A)
2) visually, by comparing them directly to images generated by pen-and-ink (Fig. 25)

A detailed evaluation of these techniques requires expertise in the human-computer interaction area and is considered beyond the scope of this article.

Another area of future research is to use a MAR-like multiresolution pipeline to stylize interior strokes as an improvement to traditional B-spline or low-pass filtering methods [31], [11]. The MAR approach could also be combined with the approach presented by Kalnins et al. [32] for coherent silhouettes.

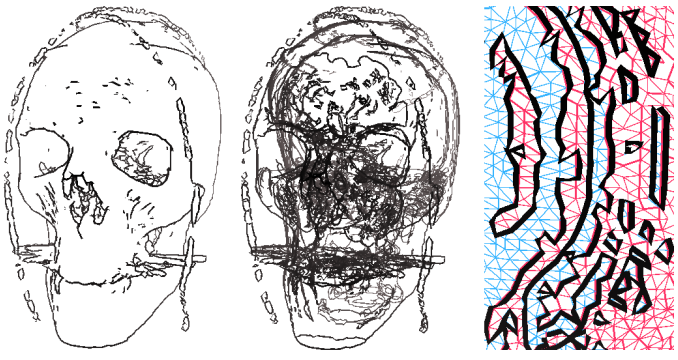

Fig. 24. Silhouettes extracted from a detailed scan mesh of a skull. *Left:* with Hidden Line Removal (HLR). *Middle:* without HLR. *Right:* a side view of the silhouette edges extracted.

Fig. 25. Comparing real pen-and-ink silhouette rendering to the results of the MAR approach. *Top-row:* a real illustration of a seagull and images of a similar level of detail generated with the system. *Bottom-row:* a real illustration of a plant (*left*), and images of several plants generated with the system. *Plant meshes courtesy Martin Fuhrer*.

REFERENCES

[1] W. Crane, *Line Form*. George Bell Sons, London, 1900.
[2] G. Simmons, *The Technical Pen*. Watson-Guptill Publications, 1992.
[3] E. Hodges, *The Guild Handbook of Scientific Illustration*. Van Nostrand Reinhold, 1989.
[4] M. Salisbury, "Ink-based pen-and-ink illustration," in *PhD Thesis*, 1997.
[5] C. Christou, J. J. Koenderink, and A. J. van Doorn, "Surface gradients, contours and the perception of surface attitude in images of complex scenes." Perception, 1996, pp. 701–713.
[6] T. Isenberg, B. Freudenberg, N. Halper, S. Schlechtweg, and T. Strothotte, "A developer's guide to silhouette algorithms for polygonal models," in *IEEE Computer Graphics and Applications*, 2003, pp. 28–37.
[7] S. Whitaker, *The Encyclopedia of Cartooning Techniques*. Running Press, Philadelphia, 1994.
[8] T. Isenberg, N. Halper, and T. Strothotte, "Stylizing Silhouettes at Interactive Rates: From Silhouette Edges to Silhouette Strokes," *Proc. of Eurographics 2002*, vol. 21, no. 3, pp. 249–258, Sept. 2002.
[9] L. Markosian, M. Kowalski, S. Trychin, L. Bourdev, D.Goldstein, and J. Hughes, "Real-time non-photorealistic rendering," in *Proc. of SIGGRAPH '97*, 1997, pp. 415–420.
[10] J. Northrup and L. Markosian, "Artistic silhouettes: A hybrid approach," pp. 31–38, 2000.
[11] M. Sousa and P. Prusinkiewicz, "A few good lines: Suggestive drawing of 3d models," *Proc. of Eurographics 2003*, vol. 22, no. 3, pp. 327 – 340, 2003.
[12] A. Hertzmann and D. Zorin, "Illustrating smooth surfaces," in *Proc. of SIGGRAPH 2000*, K. Akeley, Ed. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000, pp. 517–526.
[13] W. Corrêa, R. Jensen, C. Thayer, and A. Finkelstein, "Texture mapping for cell animation," in *Proc. of SIGGRAPH '98*, 1998, pp. 435–446.
[14] R. Bartels and F. Samavati, "Reversing subdivision rules: Local linear conditions and observations on inner products," in *Journal of Computational and Applied Mathematics, vol. 119, Issue 1-20*, 2000, pp. 29–67.
[15] F. Samavati and R. Bartels, "Multiresolution curve and surface representation by reversing subdivision rules," in *Computer Graphics Forum, Vol. 18, No. 2*, 1999, pp. 97–120.

[16] T. Saito and T. Takahasi, "Comprehensible rendering of 3d shapes." in *Proc. of SIGGRAPH'90*, 1990, pp. 197–206.
[17] A. Hertzmann, "Introduction to 3d non-photorealistic rendering: Silhouettes and outlines," in *Non-Photorealistic Rendering (SIGGRAPH '99 Course Notes)*, 1999.
[18] O. Deussen and T. Strothotte, "Computer-generated pen-and-ink illustration of trees," in *Proc. of SIGGRAPH 2000*, 2000, pp. 13–18.
[19] B. Gooch, P. Sloan, A. Gooch, P. Shirley, and R. Riesenfeld, "Interactive technical illustration," in *1999 ACM Symposium on Interactive 3D Graphics*, 1999, pp. 31–38.
[20] J. Rossignac and M. van Emmerik, "Hidden contours on a frame-buffer," in *Proc. 7th Eurographics Workshop Computer Graphics Hardware*, 1992, pp. 108–204.
[21] P. Rustagi, "Silhouette line display from shaded models," in *Iris Universe*, 1989, pp. 42–44.
[22] R. Raskar and M. Cohen, "Image precisiton silhouette edges," in *Proc. 1999 ACM Symp. Interactive 3D graphics*, S. Spencer, Ed. ACM Press, 1999, pp. 135–140.
[23] J. Buchanan and M. Sousa, "The edge-buffer: A data structure for easy silhouette rendering," in *Proc. of NPAR 2000*, 2000, pp. 39–42.
[24] P. V. Sander, X. Gu, S. J. Gortler, H. Hoppe, and J. Snyder, "Silhouette clipping," in *Proc. of Siggraph 2000*, K. Akeley, Ed. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000, pp. 327–334.
[25] A. Finkelstein and D. H. Salesin, "Multiresolution curves," in *Proc. of SIGGRAPH'94*. ACM Press, 1994, pp. 261–268.
[26] D. Kirsanov, P. V. Sander, and S. J.Gortler., "Simple silhouettes over complex surfaces," in *Proc. of First Symposium on Geometry Processing 2003*, 2003, pp. 102–106.
[27] E. Stollnitz, T. Derose, and D. Salesin, "Wavelets for computer graphics: Theory and applications," in *Morgan Kaufmann*, 1996.
[28] F. Samavati and R. H. Bartels, "Local b-spline wavelets," in *Biometric 2004, University of Calgary*, 2004.
[29] R. H. Bartels, G. H. Golub, and F. F. Samavati, "Some observations on local least squares," in *Stanford University, Tech. Rep. SCCM-04-09*, 2004.
[30] M. Brunn, M. Sousa, and F. Samavati, "Capturing and re-using artistic styles with reverse subdivision-based multiresolution methods," in *Department of Computer Science Tech. Report, University of Calgary*, 2004.
[31] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive contours for conveying shape," in *Proc. SIGGRAPH 2003*, 2003, pp. 848–855.
[32] R. Kalnins, P. Davidson, L. Markosian, and A. Finkelstein, "Coherent stylized silhouettes," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 856–861, July 2003.

Kevin Foster holds a B.Sc. with a specialization in Video Game programming and a M.Sc. both in Computer Science from the University of Calgary. His research interests include non-photorealistic rendering, implicit surfaces, computer animation, natural phenomena simulation, realtime programming and digital music. His M.Sc. research focusses on non-photorealistic pen-and-ink extraction and stylization techniques for 3D polygonal surfaces. He has also written several research papers covering pen-and-ink techniques for implicit surfaces. At the university of Calgary, he has taught several courses, including programming, graphics and animation classes and he has held programming positions at several Calgary companies (CMT Systems Ltd., Cana Construction).

Mario Costa Sousa is an Assistant Professor in the Department of Computer Science at the University of Calgary. He holds a M.Sc. (PUC-Rio, Brazil) and a Ph.D. (University of Alberta) both in Computer Science. He performs research in non-photorealistic rendering (NPR), illustrative visualization, 3D modelling and volumetric display software. His current focus is on research and development of NPR methods for 3D model construction/analysis, natural media simulation, rendering techniques and systems for computer-generated illustrative visualization and composition.

Faramarz F. Samavati is an Assistant Professor in the Department of Computer Science at the University of Calgary. He received his Ph.D. from Sharif University of Technology (Tehran, Iran) in 1999. His research interests are computer graphics, geometric modelling, scientific visualization and computer vision. His current research is focused on multiresolution and subdivision methods, sketch based modelling and non-photorealistic rendering.

Brian Wyvill received his PhD from the University of Bradford in 1975 and continued his interest in computer animation as a research fellow at the Royal College of Art. After working as an animation consultant for projects such as some scenes from the film 'Alien', he joined the University Calgary faculty in 1981. He is now a full professor and current interests include implicit modelling, non-photo realistic rendering and sketch based modelling. He is a member of ACM, CGS, and Eurographics, and editorial boards of the Visual Computer and the Journal of Shape Modelling.