Highly Parallel Algorithms for Visual Perception Guided Surface Remeshing

Lianping Xing, Xiaoting Zhang, Charlie C.L. Wang, *Senior Member, IEEE* and Kin-Chuen Hui

***Abstract*— This article presents highly parallel algorithms for remeshing polygonal models guided by cues from human visual perception. The remeshing framework is based on meshfree techniques for processing surface sample points. The benefit is twofold: it is robust to input models with problematic connectivity and the geometric processing on points is easier to run in parallel on GPUs. The visual perception information is extracted in the image space and then mapped back to the Euclidean space. Based on these cues, a saliency field is generated to re-sample the input model. Lastly, a new projection operator is developed to further optimize the distribution of re-sampled points. As the number of vertices on the resultant model is controlled by the down-sampled points, this remeshing framework can also be used in model simplification. Experimental results demonstrate that our algorithm can remesh diverse polygonal models to well-shaped triangular meshes with high visual fidelity.**

***Index Terms*— Parallel algorithm, visual perception, sampling, remeshing, simplification.**

I. Introduction

In computer graphics applications, polygon mesh has been a prevalent form of three-dimensional geometric shape representations. Mesh models can be created from various sources, such as modeling software and 3D range scans. However, due to the limitations of modeling methods, the resulting meshes though can capture 3D shapes accurately but may not provide satisfactory mesh quality. Some meshes may even contain defects such as gaps, holes, self-intersected triangles, etc. Remeshing technique is usually employed to improve both the quality of geometry and the connectivity of meshes. Depending on different target applications, the goal of a remeshing approach may vary. Commonly agreed properties of a good remeshing approach include:

- **General:** Its requirement on the quality of input models should be general. The algorithm can be applied to a variety of models such as orientable 2-manifold piecewise linear surfaces, polygon soup, etc.
- **Accurate:** It generates a mesh surface that is as close as possible to the input model, and the distribution of vertices on the resultant mesh leads to good element shape (e.g., nearly regular triangles). To achieve the goal of high accuracy, the vertices usually need to distribute adaptively according to some density functions.
- **Efficient:** The approach is fast to process huge models with massive number of polygons within a reasonable time.
- **Simple:** The algorithm is easy to implement.

In recent years, there are many research approaches of surface resampling aiming at generating a particular type of point distribution that captures the characteristics of the underlying model. Different patterns are produced in different approaches (e.g., uniform sampling, curvature adapted sampling and Poisson disk sampling). Most of the existing remeshing techniques take tremendous time in computation and sequential algorithms are conducted. It is hard for them to take advantage of the computational power provided on *Graphics Process Units* (GPUs). This motivates our work presented in this paper.

Three-dimensional models represented by polygonal meshes are now ubiquitously used in a large number of human-centered visual computing applications. The human perception cues have proved to be able to improve the reliability and robustness of geometry processing algorithms. There is a growing demand for incorporating insights from human perception into mesh processing. Researchers have shown that the comprehensibility of complex 3D models can always be greatly enhanced by guiding the attention of users to visually salient regions in low-level human vision. Due to its efficiency of visual persuasion in traditional art and technical illustration, visual perception technique has now been widely used in many computer graphics applications, including feature extraction and shape matching (e.g., [1]). However, there is no visual perception guided surface remeshing approach available in literature, which is the niche of our work.

Our remeshing framework generates quality mesh surfaces in three major steps: 1) visual feature extraction, 2) resampling and 3) samples optimization and meshing. Figure 1 illustrates the whole procedure of our remeshing framework. The processing starts from capturing the image snaps of the input model. After that, the visual saliency features are extracted in the image-spaces of these snaps. These visual saliency features are mapped back onto the input model as a set of 3D samples for visual saliency – called *visual saliency samples*. According to the visual saliency samples, a saliency field is generated on the input model, where the visual perception cues are preserved in the field. Governed by the saliency field, the input model is re-sampled so that the visually important regions have more samples. Lastly, the positions of sample points are optimized by applying a newly developed *Adaptive Weighted Locally Optimal Projection* (AWLOP) operators. Notice that, all the algorithms developed in these steps can be easily parallelized to run on GPUs. By the optimized samples, a mesh

Manuscript submitted in March 2013; revision submitted in August 2013.
All authors are with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, NT, Hong Kong. Corresponding Author: Charlie C.L. Wang (Tel: (852) 3943 8052; Fax: (852) 2603 6002; E-mail: cwang@mae.cuhk.edu.hk).

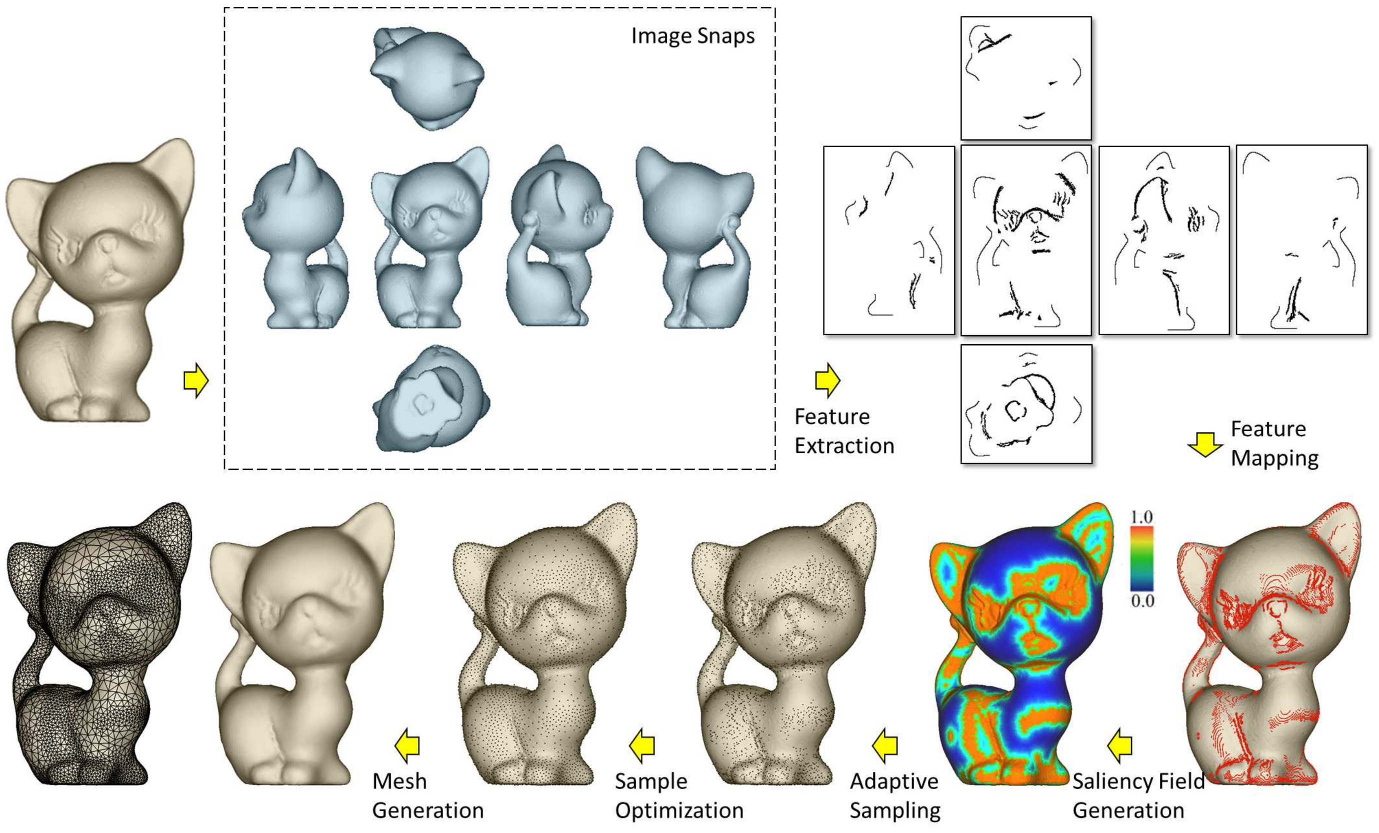


Fig. 1. The overview of our visual perception guided remeshing framework. The image snaps of an input model is first captured in six orthogonal views. The perceptual features are then extracted in the image space and mapped back to $\Re^3$ as *saliency points* – see the red ones on the model at the right of bottom row. After that, saliency field is generated and used to govern the adaptive sampling. Lastly, the sample points are optimally positioned by AWLOP operators and connected into a two-manifold mesh surface.

connectivity can be easily reconstructed by using the computational geometry techniques (e.g., Tight CoCone available at: www.cse.ohio-state.edu/%7Etamaldey/cocone.html).

The main contributions of our approach are as follows:

1) a visual perception guided surface remeshing framework which can generate mesh surfaces preserving the cues for human perception,
2) an adaptive sampling method that generates sample points according to the saliency field,
3) a point projection operator, AWLOP, for optimizing the distribution of sample points,
4) highly parallel algorithms used in the remeshing framework that can run on GPUs.

As a result, an effective and efficient pipeline for surface remeshing is developed.

The rest of our paper is organized as follows. After reviewing the related work in Section II, the method for visual saliency extraction is presented in Section III. Section IV describes the construction of saliency field and how it guides the resampling to be adaptive to visual saliency. The *Adaptive Weighted Locally Optimal Projection* operator is proposed in Section V to optimize the positions of sample points. Experimental results are shown and discussed in Section VI. Lastly, our paper ends with the conclusion section.

II. Related Work

Remeshing is a technique for improving mesh quality in many computer graphics applications (e.g. shape editing, animation, and numerical simulation). In recent years, it has received considerable attention and a variety of remeshing algorithms have been developed. Existing techniques can roughly be classified into two categories: the approaches that are computed in parametric domains (e.g., [2], [3]) and those that are directly generated on 3D surfaces (e.g., [4], [5]).

The key idea of parameterization based methods is to partition a parameter domain into sets of adjacent elements that have the same specific properties. Gu et al [2] proposed a technique that first cuts the surface into patches, then parameterizes it using a signal-adapted technique and finally represents the surface as a set of images that store the geometry and other attributes used for visualization. Surazhsky et al [3] introduced a remeshing algorithm based on local parameterization. However, parameterizing freeform models is challenging and always introduces severe distortions.

To alleviate the above problems, remeshing methods that directly take sampling on 3D meshes have been proposed. In an earlier work, Turk [4] proposed a re-tiling technique that applies an attraction-repulsion particle relaxation procedure to resample an input mesh. A curvature-aware adaptive sampling method was introduced in [6], which can help to produce high-quality meshes. Fuhrmann et al [5] presented a curvature adaptive remeshing algorithm which is based on *Weighted Centroidal Voronoi Tessellation* (WCVT). However, none of these works take visual perception into consideration. On the other aspect, these approaches are highly time-consuming and cannot be sped up by GPU-based computing.

III. Extraction of Perception Cues

There are a number of excellent approaches that generate different styles of depiction for 3D shapes according to visual requirements. Recent studies demonstrate that advanced line drawing techniques can effectively depict 3D shape and match the effectiveness of artist's drawings (ref. [7]). According to prior research, it is commonly agreed that a good depiction of 3D shape should include a wealth of other visual cues beyond contours. In this sense, the perception cues on a given model is not limited by its silhouettes. Some existing approaches (e.g., [1]) are following this thread of research. However, these methods fail to process models in the form of polygon soup. In this section, we borrow tools from computer vision to extract the perception cues in image space (i.e., the different views of input models). After that, the results are mapped back to 3D models as 3D visual saliency points. Features of graphics hardware are explored to speed up the extraction as well as mapping.

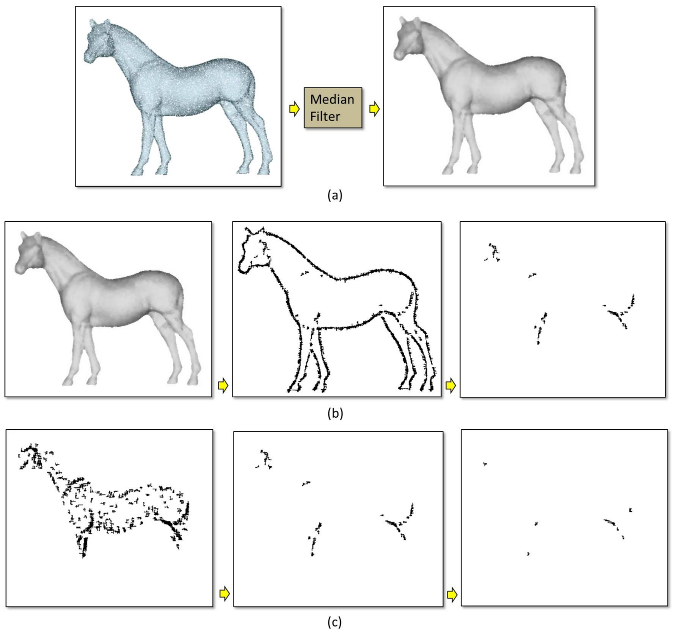


Fig. 2. Pre-processing and inner feature filter: (a) pre-processing to fill the holes on a problematic models, (b) applying the inner feature filter on an input gray-scale image can result in a binary image containing inner features, (c) when applying different threshold C, different amount of features will be extracted: (left) $C = 0$, (middle) $C = 3$ and (right) $C = 6$.

A. Visual saliency extraction in image space

We start the remeshing procedure by taking several snapshots of an input model from different views. When focusing on obtaining a mesh model have good visualization result, six orthogonal views are adopted to generate the snapshots (see Fig.1 for an example). The images can be efficiently obtained through hardware accelerated graphics pipeline (e.g., OpenGL).

Pre-processing for problematic models

Our framework allows the models with holes and other topological problems to be processed. The boundary of holes will be simply treated as small features if not being processed. To solve this problem, we apply a low-pass filter to fill the holes for those problematic models. Specifically, a median filter using a $k \times k$ aperture is applied. The value of k is chosen according to the level of noises presented on the input model. For highly noisy model, a larger k should be used. $k = 7$ works well in all our tests. An example is shown in Fig.2(a) to show the result of this pre-processing. Note that, before applying the median filter, an RGB image must be converted into a gray-scale one.

After applying the above pre-processing, visual saliency is extracted on each snapshot with the help of two filters: 1) inner feature filter and 2) silhouette feature filter.

Inner feature filter

The inner feature filter is based on spanning a Gaussian over the image. For each pixel (i, j), its pixel value in gray scale is denoted by $P(i, j)$. A $n \times n$ 2D Gaussian mask

$$G(i,j) = \alpha e^{\frac{(i-\frac{n-1}{2})^2 + (j-\frac{n-1}{2})^2}{(2\sigma)^2}} \tag{1}$$

is adopted in our inner feature filter, where the bandwidth parameter $\sigma = 0.3(\frac{n-1}{2} - 1) + 0.8$ and α is a scale factor chosen to let $\sum_{i,j \epsilon mask} G(i,j) = 1$. In all our tests, $n = 11$ and $\alpha = 0.0242$ is employed. Using this Gaussian mask, a weighted average image, $T(u, v)$, can be obtained by

$$T(u,v) = \sum_{i,j \epsilon mask} G(i,j)P(u+i, v+j) - C \tag{2}$$

with C being a threshold to control how many feature edges can be extracted. The resultant binary image containing the pixels of candidate features can be obtained by

$$\bar{F} = \{(u,v) | T(u,v) > P(u,v)\}. \tag{3}$$

Using a smaller value for C will have more pixels remained after taking the filter above. $C = 3$ is chosen in our implementation. The final inner features are determined by excluding the silhouette pixels. Figure 2(b) presents an example of applying the inner feature filter to the horse model, and Fig.2(c) shows the comparison by using different threshold C.

Silhouette feature filter

Thin-and-sharp features located on the silhouette (e.g, ears and legs of the horse model) are important for representing the shape of a 3D model. In this filter, these pixels will be extracted and added into the pixel set of visual saliency, $\bar{F}$. These features on silhouette are defined as foreground pixels that satisfy:

1) having a background pixels in its 8-neighbors, and
2) with relatively small distance to the skeleton of input model.

Therefore, to find the thin-and-sharp features on silhouette, we need to extract the skeleton of an input model in the image space and compute distances from the silhouette pixels to the skeleton. Both can be realized in parallel with the help of a highly parallel distance transformation algorithm.

Distance transformation uses a small mask, M, to propagate the distances over the image in an iterative way. At the beginning of the transformation, the distances D at the source pixels are assigned as zero while distances at other pixels are all initialized as infinity. Then, the distance value at each pixel (u, v) is updated in parallel by

$$D(u,v) = \min_{(i,j)\in M}\{D^{prev}(u+i, v+j) + D_M(i,j)\}, \quad (4)$$

where D^{prev} denotes the distance at a pixel in last iteration, and $D_M(i,j)$ gives the local distance from (i,j) to the center of a mask. When $b\times b$ mask is used, $D_M(i,j) = ((i-\frac{b-1}{2})^2 + (j-\frac{b-1}{2})^2)^{\frac{1}{2}}$. The update is run in parallel on all pixels for a few iterations until no distance value is changed, which is easy to be checked with the help of the *scan* primitives. Or a fixed number, m, of iterations are conducted. m is set as the diagonal length of the input image divided by the size b of the transformation mask M. Note that, a more accurate parallel distance transformation algorithm can also be used here. However, as accuracy is not a major concern in our filter, we employ the above algorithm that is easier to implement.

To extract the skeleton of an input model in the image space, we first compute a distance map from every foreground pixel to the silhouette pixels (by letting the silhouette pixels as sources in the above distance transformation algorithm). Four kernels (see Fig.3(a)) are applied across the distance map to extract the corresponding directional gradients. Among them, if a significantly large gradient is found at a pixel, the pixel is considered as belonging to the skeleton. Specifically, when $K_i(u,v)$ denotes the response of kernel i for a pixel (u,v), the skeleton pixels can be extracted by a filter as

$$\bar{S} = \{(u,v)|\max_{\forall i}\{K_i(u,v)\} > \lambda\} \quad (5)$$

with $\lambda = 8$ being the threshold for selecting the significantly large gradient. Using a larger λ will generate sparse points for the skeleton, while a smaller λ gives dense skeleton with many unwanted branches. An example of applying this skeleton extraction filter can be found in Fig.3(b).

Another distance map is generated by using the skeleton pixels as sources, the value of which on the silhouette pixels actually presents the feature size. With this information, we can detect the silhouette pixels with small feature size (i.e., less than 20 in our tests on 512×512 images). These pixels are added into the set of feature points. Figure 3(c) shows the result of extracting such thin-and-sharp features on a horse model.

Highly Parallel Implementation on GPU It is not surprise to see that implementing these image processing operators is not difficult. All the operators can be evaluated independently based on the neighboring pixels' information before applying the operators. As a result, they are realized as kernels running on GPU with the help of CUDA SDK library.

-1	-1	-1	-1	-1
0	0	0	0	0
2	2	2	2	2
0	0	0	0	0
-1	-1	-1	-1	-1

-1	0	2	0	-1
-1	0	2	0	-1
-1	0	2	0	-1
-1	0	2	0	-1
-1	0	2	0	-1

0	-1	-1	0	2
-1	-1	0	2	0
-1	0	2	0	-1
0	2	0	-1	-1
2	0	-1	-1	0

2	0	-1	-1	0
0	2	0	-1	-1
-1	0	2	0	-1
-1	-1	0	2	0
0	-1	-1	0	2

(a)

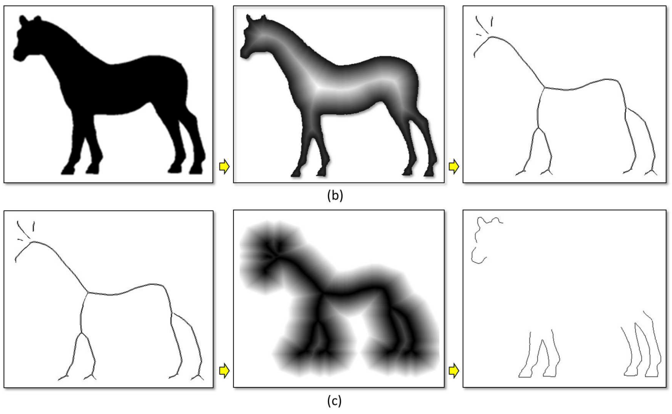


Fig. 3. Silhouette feature filter: (a) four kernels are used for the skeleton extraction on a silhouette distance map – from left to right: horizontal, vertical, 45-degree, and 135-degree edge extraction kernel, (b) the skeleton of an input model (left) can be extracted in the image space by the distance map (middle) of its silhouette, (c) the thin-and-sharp features of a model can be detected on the silhouette with the help of a distance map (middle) of its skeleton (left).

B. Image space to Euclidean space mapping

After detecting the visual saliency in image spaces, the set of feature pixels need to be mapped back to the Euclidean space to guide the following step of sampling. The mapping result can be efficiently obtained with the help of hardware accelerated graphics pipeline (e.g., OpenGL in our implementation). When taking snapshots from different views, we record the z-buffer value of every pixel. These z-buffer values can help to un-project every pixel in the set of feature points back to $\Re^3$ to serve as saliency points.

IV. Visual Perception Guided Sampling

Once saliency points are generated on a surface, a scalar saliency field needs to be built over the entire surface domain to govern the resampling procedure.

A. Saliency field generation

We aim at distributing a user-specified amount of samples over the mesh surface such that more points would be located on the visually salient regions. Specifically, the vertices of an input model near saliency points should have higher visual importance. The visual importance of a saliency point is set to 1.0 to represent the highest visual importance. Then, the values within $[0, 1)$ are assigned over the surface domain to all mesh vertices.

We use an advancing front method to generate the visual saliency field, which progressively moves a front L from the saliency points to their nearest neighboring vertices on the surface and then to farther vertices until all vertices of the input model have been travelled. Before the propagation, all saliency points and vertices of the input model are inserted into

Algorithm 1: SaliencyFieldPropagation

```
   Input: the set of saliency points $F$
1  and a model $M$
   Output: the propagated depths on vertices of $M$
2  Set depth-values of all points in $F$ as zero;
3  Set depth-values of all vertex on $M$ as $\infty$;
4  Insert all points of $F$ into the front $L$;
5  while $L \neq \emptyset$ do
6      foreach $v_i \in M$ in parallel do
7          if ($v_i$ is untravelled) AND ($v_i$ is the neighbor of a
           travelled point) then
8              Set $v_i$ as a candidate vertex of 'next-front';
9          end
10     end
11     Compact all 'next-front' vertices on $M$ into a new
       set $L'$;
12     foreach $v_k \in L'$ in parallel do
13         $d_{v_k} \Leftarrow \min_j\{d_{v_j} + 1\}$ for all neighbors $v_j$ of $v_k$;
14         Set $v_k$ as travelled;
15     end
16     Update the front as $L \Leftarrow L'$;
17 end
18 return;
```

a kD-tree to conduct the approximate nearest neighbor search. As a result, the neighborhood table can be constructed and copied to the GPU side to govern the propagation of saliency field. During the advancing process, the depth d_{v_i} of every vertex v_i which indicates the shortest 'distance' to its nearest saliency point is updated. The depths of all mesh vertices are initialized as $+\infty$ while the depths of saliency points are set to zero. The pseudo-code for parallel propagation is given as **Algorithm** *SaliencyFieldPropagation*.

Then the visual importance of every vertex v_i located on the mesh surface is set to be

$$I_{v_i} = \beta^{d_{v_i}}. \tag{6}$$

where $\beta \in [0,1]$. The value of β approximately indicates the smoothness of the saliency field. The larger the value is, a smoother field is given. For all the models shown in this paper, $\beta = 0.7$ is used and the number of neighbors is set to $k = 10$ to balance the speed and accuracy.

B. Adaptive sampling

We integrate the saliency field over the surface and obtain a visual quantity V_s as

$$V_s = \sum_{i=1}^{n} V_i \tag{7}$$

where n is the number of triangles, V_i is the visual quantity of the i-th triangle that can be obtained by

$$V_i = \frac{1}{3} A_i \sum_{j=1}^{3} I_{v_j} \tag{8}$$

with the triangle area A_i and the visual importance on vertices as I_{v_j}. Suppose n_s sample points are going to be generated on the input model, the number of samples, n_i, in the i-th triangle can be calculated by $n_i = \lfloor \frac{V_i}{V_s} n_s + 0.5 \rfloor$. In this formula, n_i has been rounded to an integer which therefore introduces a signed quantization error E_r. When the sampling procedure is taken triangle by triangle, the quantization error is accumulated and cannot be neglected. Therefore, the number of samples in the i-th triangle are corrected to

$$n_i = \lfloor \frac{V_i}{V_s} n_s + \sum_{r=1}^{i-1} E_r + 0.5 \rfloor \tag{9}$$

by considering the quantization error

$$E_r = \frac{V_r}{V_s} n_s - n_r \tag{10}$$

on all previously sampled triangles.

Uniform Triangle-Sampling The samples inside a triangle with three vertices: $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ can be generated with the help of barycentric coordinate $\mathbf{b} = (b_1, b_2, b_3)$ with $b_1, b_2, b_3 \in [0,1]$ and $b_1 + b_2 + b_3 \equiv 1$. First, two random numbers $r_1 \in [0,1]$ and $r_2 \in [0,1]$ are generated. Then, they are used to form a barycentric coordinate as

$$\mathbf{b} = (\min\{r_1,r_2\}, \max\{r_1,r_2\} - \min\{r_1,r_2\}, 1 - \max\{r_1,r_2\}) \tag{11}$$

so that the position of a new sample point in the triangle is given by $\mathbf{b}$. When the number of expected samples inside a triangle is small (e.g., $n_i < 5$), this simple uniform triangle-sampling can be employed as the distribution of samples according to the visual importance is mainly contributed by Eq.(9). However, when n_i becomes big, it is also expected that the distribution of sample points inside triangles follows the visual importance evaluated on vertices. Then, the adaptive triangle-sampling is employed.

Adaptive Triangle-Sampling As knowing the visual importance values I_1, I_2 and I_3 on the three vertices $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ of a triangle T, the expected distribution of samples can be formulated as a normalized function $J(\mathbf{b})$ with the help of barycentric coordinate $\mathbf{b} = (b_1, b_2, b_3)$ and $I(\mathbf{b}) = b_1 I_1 + b_2 I_2 + b_3 I_3$.

$$J(\mathbf{b}) = \frac{1}{\int_T I(\mathbf{b}) dT} I(\mathbf{b}), \tag{12}$$

where $\int_T I(\mathbf{b}) d\mathbf{T} = \frac{A}{3}(I_1 + I_2 + I_3)$, A is the area of the triangle. For a sample point following the expected distribution $J(\mathbf{b})$, the coordinates b_1 and b_2 are obtained one by one. $b_3 = 1 - b_1 - b_2$ can be determined thereafter.

By introducing a marginal density of b_1,

$$J_M(b_1) = \int_T J(b_1, b_2, 1 - b_1 - b_2) db_2,$$

the *cumulative density function* (CDF) of b_1 can be formulated as

$$F_1(x) = \int_0^x J_M(b_1) db_1. \tag{13}$$

According to the analysis given in [8], for a CDF, $F(\cdots)$ of a random variable x, if another random variable r comes

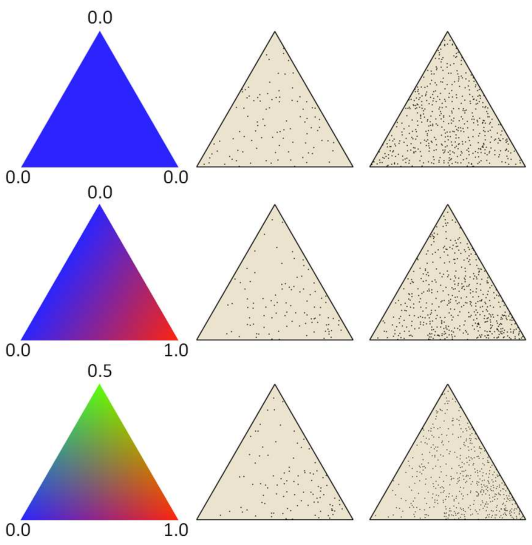


Fig. 4. Adaptive samples can be generated by the inversion method [8] according to the visual importance assigned on vertices of a triangle (left column). (Middle column) 100 samples are generated on triangles, and (right column) 500 points are sampled.

from uniform distribution in $[0,1]$, the random variable $z = F^{-1}(r))$ comes from the same distribution of x. Specifically, for a random draw $r_1 \in [0,1]$, x will follow the expected distribution of b_1 when $F_1(x) = r_1$ is enforced. That is, $b_1 = x$ can be determined by solving the following equation.

$$ax^3 + bx^2 + cx = (I_1 + I_2 + I_3)r_1 \tag{14}$$

with $a = I_2 + I_3 - 2I_1$, $b = 3(I_1 - I_2 - I_3)$ and $c = 3(I_2 + I_3)$. Given the value of b_1, the conditional distribution of b_2 is

$$J_C(b_2) = J(b_1, b_2, 1 - b_1 - b_2)/J_M(b_1).$$

Then, the CDF of b_2 is formulated as

$$F_2(y) = \int_0^y J_C(b_2)db_2 \tag{15}$$

Similarly, for a random draw $r_2 \in [0,1]$, y will follow the expected distribution of b_2 when $F_2(y) = r_2$ is enforced – that is the solution of

$$dx^2 + ex = f \tag{16}$$

with $d = \frac{1}{2}(I_2 - I_3)$, $e = I_1 b_1 - I_3 b_1 + I_3$ and

$$f = \tfrac{1}{2}((I_2 + I_3 - 2I_1)b_1^2 + 2(I_1 - I_2 - I_3)b_1 + I_2 + I_3)r_2.$$

The sampling method described above can efficiently generate samples such that more points would be produced at the region with high visual-importance, which follows the perception cues extracted in the image space. As shown in Fig.4, an adaptive distribution of samples can be generated by following the visual importance assigned at vertices.

Hybrid CPU/GPU Implementation The sampling procedure of our algorithm can be implemented in a hybrid CPU/GPU manner. First of all, how many points need to be sampled on each triangle are evaluated on CPU. Then, the triangles are classified into two groups: a group of triangles to be uniformly sampled and another group for those to be adaptively sampled. At last, the samples for each group of triangles are generated in parallel on GPU and inserted in a 1D array with the help of atomic operator on GPU.

V. Sample Distribution Optimization

The samples generated by above method follows the indication presented by the saliency field. However, on the other aspect, the samples are not distributed regularly enough to be linked to form triangles in good shape (i.e., with nearly equal angles on vertices). In this section, we propose an iterative method to further improve the regularity of sample distribution. After re-positioning the sample points, they are connected together to form a two-manifold triangular mesh by the Tight Co-Cone algorithm.

Our algorithm to re-position sample points is akin to the projection operators used in point-sample surfaces – specifically, the *Locally Optimal Projection* (LOP) operator. Given a data point-set $P = \{\mathbf{p}_j\} \subset \Re^3$, LOP projects a set of particles $X = \{\mathbf{x}_i\} \subset \Re^3$ onto the surface formed by the set P by approximating their L^1-medians. To improve the regularity of projected particles, Huang et al. [9] presented a *Weighted Locally Optimal Projection* (WLOP) operator, which introduces a new repulsion term to control the distribution of particles. As a result, the particles are pushed to places which have nearly equal distances to their neighbors. In other words, a uniform distribution of particles is obtained.

WLOP cannot be simply applied here to optimize the distribution of samples since it does not consider the guidance of saliency field. To solve this problem, we extend WLOP to an adaptive one – called AWLOP, which incorporates visual saliency values of projected points into WLOP. Specifically, the point-set P used here can be obtained from the input model M by either 1) uniformly sampling M when the number of vertices on M is small or 2) directly using the vertices of M if the number of vertices is large. The particles in X are actually the sample points obtained in Section IV. Every particle $\mathbf{x}_i \in X$ is moved to a new position by the formula below. Similar to WLOP, the update of position consists of two terms, where the first term attracts the particle to the given point set by the weighted local density

$$v_j = 1 + \textstyle\sum_{\mathbf{p}_l \in (P \setminus \{\mathbf{p}_j\})} \theta(\|\mathbf{p}_j - \mathbf{p}_l\|)$$

and the second term repulses the particles away from other particles by a particle-distribution density w_q. The updated position of $\mathbf{x}_i$ is

$$\begin{aligned}\mathbf{x}_i = &\textstyle\sum_{\mathbf{p}_j \in P} \mathbf{p}_j \frac{\theta(\|\mathbf{x}_i - \mathbf{p}_j\|)/v_j\|\mathbf{x}_i - \mathbf{p}_j\|}{\sum_{\mathbf{p}_j \in P}\theta(\|\mathbf{x}_i - \mathbf{p}_j\|)/v_j\|\mathbf{x}_i - \mathbf{p}_j\|} \\ &+\mu \textstyle\sum_{\mathbf{x}_q \in (X \setminus \{\mathbf{x}_i\})} (\mathbf{x}_i - \mathbf{x}_q) \frac{w_q \theta(\|\mathbf{x}_i - \mathbf{x}_q\|)/\|\mathbf{x}_i - \mathbf{x}_q\|}{W}\end{aligned} \tag{17}$$

where $W = \sum_{\mathbf{x}_q \in (X \setminus \{\mathbf{x}_i\})} w_q \theta(\|\mathbf{x}_i - \mathbf{x}_q\|)/\|\mathbf{x}_i - \mathbf{x}_q\|$, $\|\cdots\|$ is the L^2-norm, and $\theta(r) = e^{-16r^2/h^2}$ is adopted as in [9]. $\theta(r)$ is a rapidly decreasing smooth weight function with the support radius h defining the size of the influenced neighborhood. $\mu \in [0, 0.5)$ and h are served as two parameters selected by users to tune the performance of the operator.

Different from WLOP, our AWLOP defines the particle-distribution density, w_q, at a position $\mathbf{x}$ as a function of visual

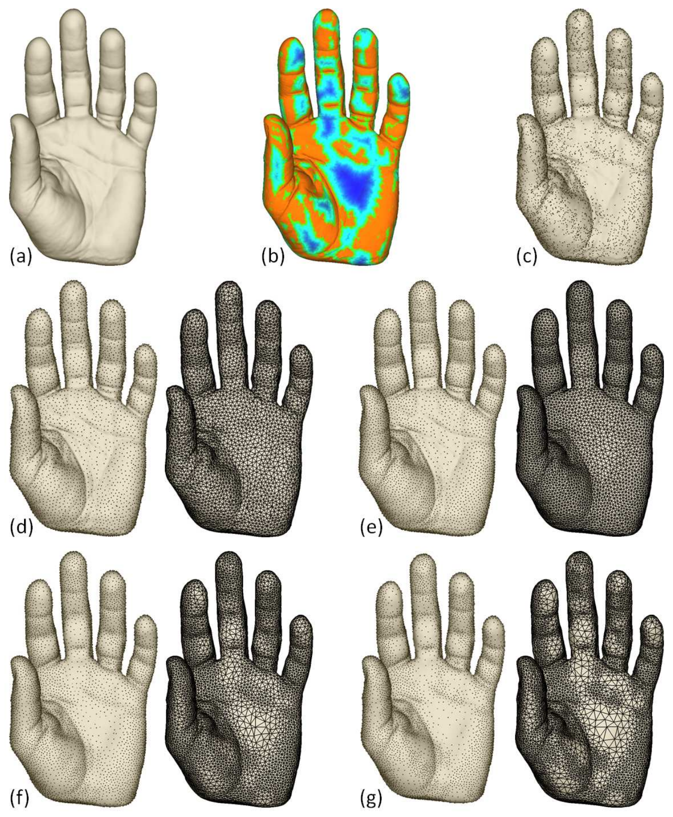


Fig. 5. For a hand model (a) with the visual saliency field shown in (b), the sampling method presented in Section IV can generate 10k points as shown in (c), the distribution of which is not regular. The results obtained by applying LOP (d) and WLOP (e) have uniformly distributed particles. The regular and adaptive distribution can be obtained by using our AWLOP (f) $w_q = 1/I(\mathbf{c}_q)$ (g) $w_q = 1/I^2(\mathbf{c}_q)$ as the particle-distribution density in the projection operator.

importance $I(\mathbf{c}_q)$ at the closest vertex of $\mathbf{x}_q$, $\mathbf{c}_q \in M$ on the input model M. From the formulation in Eq.(17), we can see that the attraction from the data point set P is conducted by the first term, and the repulsion forces between particles are adjusted by the particle-distribution density w_q in the second term. To let the distribution of particles guided by visual saliency, in the regions with high visual importance the repulsion forces should be less strengthened. Let w_q be inverse proportional to the value of $I(\mathbf{x}_q)$ can reflect this consideration, that is.

$$w_q = 1/I(\mathbf{c}_q). \tag{18}$$

For the other two parameters of AWLOP, μ and h, we set $\mu = 0.45$ and $h = 2L_{avg}$ with L_{avg} being the average distance between data points to their $k-$nearest neighbors. $k = 20$ is chosen here to balance the speed and the robustness.

Implementation Details and Comparisons The AWLOP operator can be implemented in a hybrid CPU/GPU program, where the neighborhood table of points in P and X are constructed and updated at the CPU side with the help of ANN library (available at: www.cs.umd.edu/%7Emount/ANN/). The iterative update of particles' positions is performed on GPU. As a result, the samples can be efficiently optimized to a regular distribution and meanwhile being adaptive according

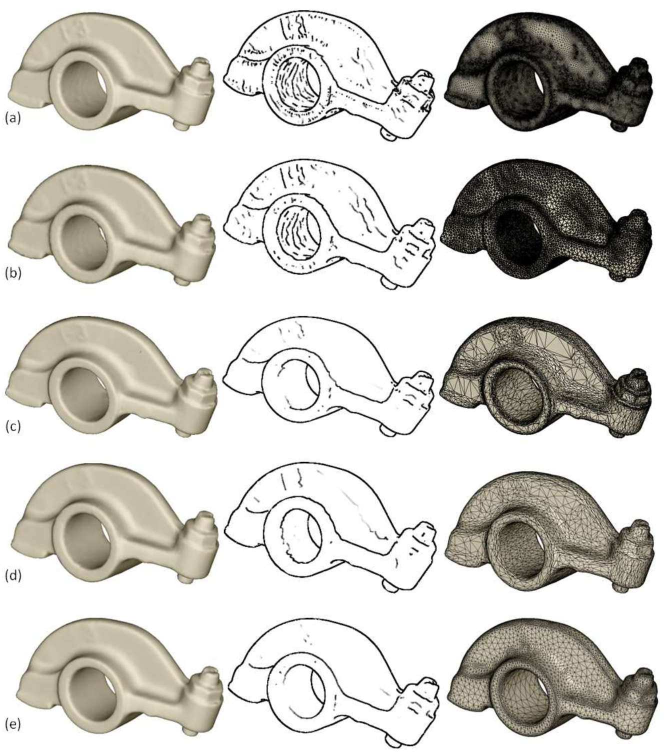


Fig. 7. Remeshing results of the Rockerarm model are shown here, where the original model (a) is remeshed from 121k vertices to 10k vertices by (b) our method, (c) mesh-saliency, (d) QEM and (e) Fuhrmann's approach respectively. The first column shows the shading models, the middle column gives the visual saliency extracted by [1], and the last column presents the corresponding mesh models.

to the visual saliency field. Figure 5 illustrates the comparison of results from LOP, WLOP and AWLOP. The input is a hand model with saliency field (Fig.5(b)) and the sample points generated by the method in Section IV (see Fig.5(c)). The distributions generated by LOP (Fig.5(d)) and WLOP (Fig.5(e)) tend to be uniform no matter what initial set shown in Fig.5(c) is. Our AWLOP can generate particles adaptive to visual saliency field as Fig.5(f) and (g). The particle-distribution density, w_q, is set to be $1/I(\mathbf{c}_q)$ in Fig.5(f) and $w_q = 1/I^2(\mathbf{c}_q)$ is employed in Fig.5(g). We can see that the distribution of the particles in Fig.5(g) is more adaptive to visual saliency field than Fig.5(f). However, it may lead to too sparse regions when the number of particles is very small. Therefore, in our all further tests, $1/I(\mathbf{c}_q)$ is used.

VI. Results and Discussion

We have implemented our algorithm using C++ language and CUDA SDK library. The implementation has been tested on various models. The results are obtained by running the algorithm on a PC equipped with Intel iCore-7 3.4GHz CPU + 8GB RAM. An implementation can be accessed at a URL link[1]. All tasks of our algorithm are run in parallel on the modern graphics hardware equipped with GeForce GTX 660 Ti GPU. With the help of highly parallel algorithms, remeshing of all examples can be finished in less than 10 seconds on our platform.

[1]www2.mae.cuhk.edu.hk/%7Ecwang/GPURemeshByVisualCues.html

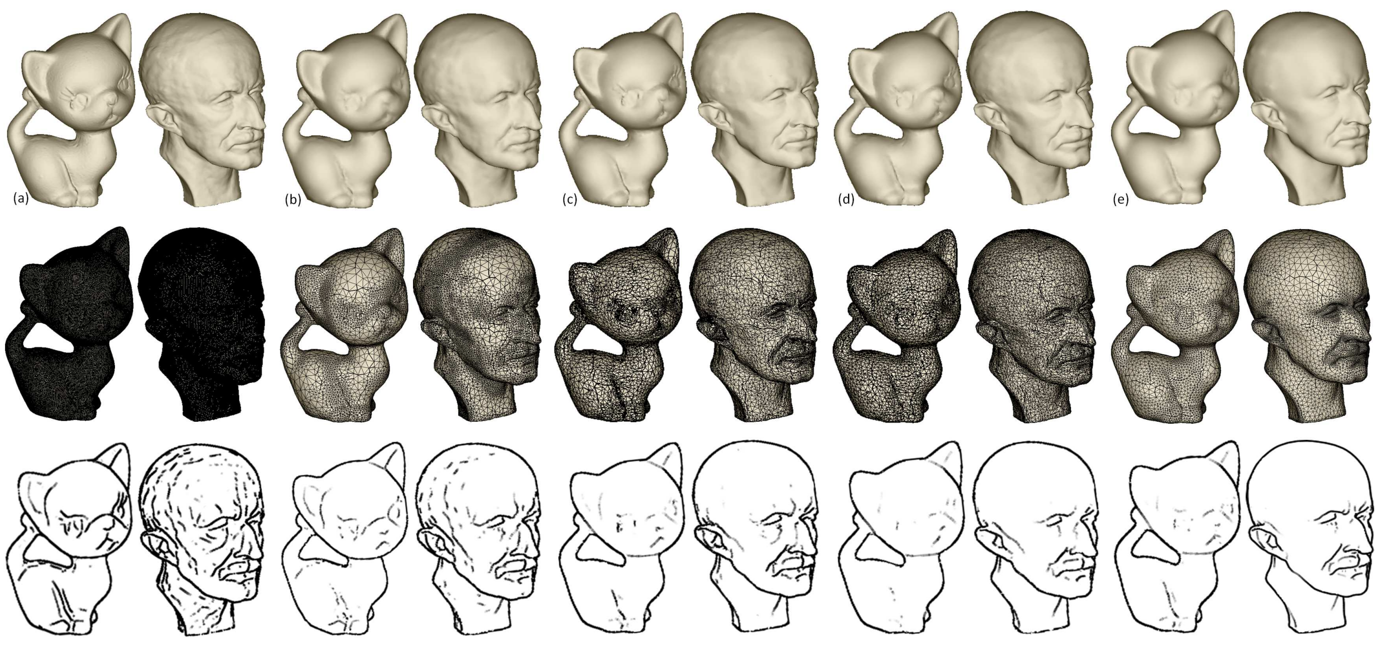


Fig. 6. The models (a) with about 137k vertices (Kitten) and 199k vertices (MaxPlanck) are remeshed to 5k and 10k vertices respectively by (b) our method, (c) mesh-saliency, (d) QEM and (e) Fuhrmann's approach. The line drawings generated by the method of [1] are shown in the bottom row.

TABLE I
COMPARISON OF RUNNING TIMES, VISUAL SIMILARITY ERRORS AND GEOMETRIC ERRORS.

Model	**Vertex Num.**		**Methods**	**Time**†	**Visual**	**Geometric Err.**‡	
	Input	**Result**		**(sec.)**	**Err.** [10]	**Max**	**Mean**
Kitten (Fig.6)	137k	5k	Ours	3.635	0.9247	0.834	3.56×10^{-2}
			Mesh-Saliency	$3,882\ (\times 1,068)$	0.9110	0.178	1.78×10^{-2}
			QEM	$9.752\ (\times 2.7)$	0.8984	0.189	1.44×10^{-2}
			Fuhrmann	$126.8\ (\times 35)$	0.9048	3.14	2.64
MaxPlanck (Fig.6)	199k	10k	Ours	5.384	0.9272	1.25	0.158
			Mesh-Saliency	$11,740\ (\times 2,180)$	0.9070	0.338	3.13×10^{-2}
			QEM	$112.0\ (\times 21)$	0.8936	0.640	2.77×10^{-2}
			Fuhrmann	$7,403\ (\times 1,375)$	0.9021	1.75	1.36
Rockerarm (Fig.7)	120k	10k	Ours	4.148	0.9536	2.80×10^{-3}	1.66×10^{-4}
			Mesh-Saliency	$4,539\ (\times 1,094)$	0.9401	9.05×10^{-4}	7.80×10^{-5}
			QEM	$9.361\ (\times 2.3)$	0.9399	9.19×10^{-4}	6.10×10^{-5}
			Fuhrmann	$26.21\ (\times 6.3)$	0.9367	3.75×10^{-3}	2.77×10^{-4}
Cane (Fig.8)	493k	20k	Ours	8.578	0.9655	1.37	0.143
			Mesh-Saliency	$104,781\ (\times 12,215)$	0.9535	0.602	4.03×10^{-2}
			QEM	$301.0\ (\times 35)$	0.9573	0.878	3.79×10^{-2}
			Fuhrmann	$654.0\ (\times 76)$	0.9334	11.5	6.94
Horse(Fig.9)	203k	10k	Ours	4.526	-	2.58×10^{-3}	9.10×10^{-5}
Igea(Fig.9)	806k	6k	Ours	5.755	-	9.33×10^{-4}	3.90×10^{-5}

†The numbers in parenthesis show the speedups of our method comparing to mesh-saliency, QEM and Fuhrmann's approach.
‡The geometric errors are measured by the publicly available Metro tool.

Figure 6 shows the remeshing results of a Kitten model (137k vertices) and a MaxPlanck model (with 199k vertices). The original models were downsampled to 5k vertices (Kitten) and 10k vertices (MaxPlanck) according to the visual saliency field. Finally, the sample points were optimized and meshed. We can observe from the result that even the model is downsampled to less than 4% of its original complexity, the resultant shape still looks similar and the distribution of its vertices is strictly adapted to its visual saliency field. More experimental results are shown in Figs.7 and 8. We compare our results with the results of QEM-based mesh simplification [11], saliency-based mesh simplification [12] and the remeshing method proposed by Fuhrmann et al. in [5] (with the contrast factor 1.5) – all based on our implementations. To compare the quality about how well these methods are able to preserve the visual perception, we render the original model and the remeshed models into line drawings by one of the state-of-the-art [1], where the non-photorealistic rendering apporach relies on the visual perception. By the rendering results shown in Fig.7-8, we can see that our algorithm is the best one to

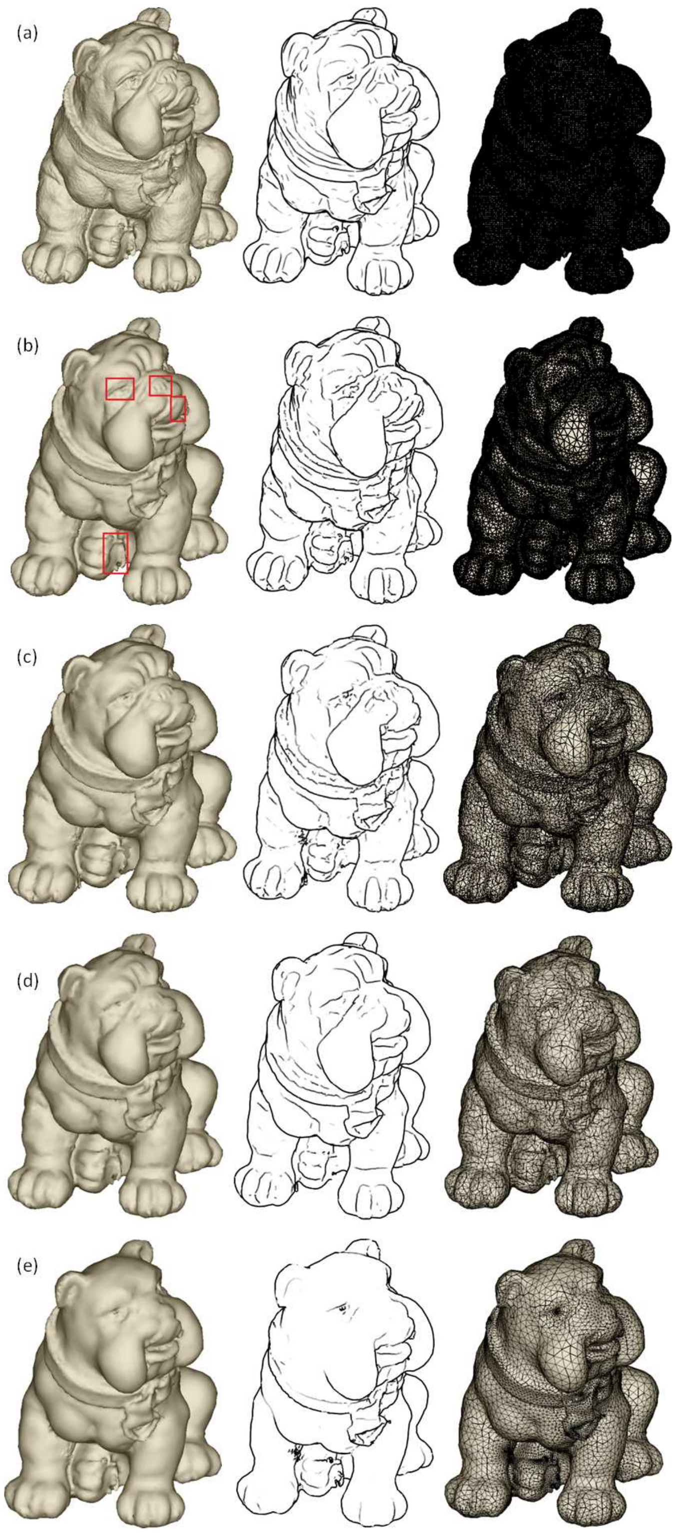


Fig. 8. The Cane model (a) with almost 1M triangles is remeshed to 20k vertices by (b) our method, (c) mesh-saliency, (d) QEM and (e) Fuhrmann's approach. The first column shows the shading models, while the middle column illustrates the extracted line drawings and the last column gives the corresponding mesh models. Our method can preserve the visual saliency better (see the eyes, nose, tongue and toe of the model which have been marked out in (b)). Moreover, our remeshing framework takes only about 8 seconds with the help of highly parallel computational power provided by GPU.

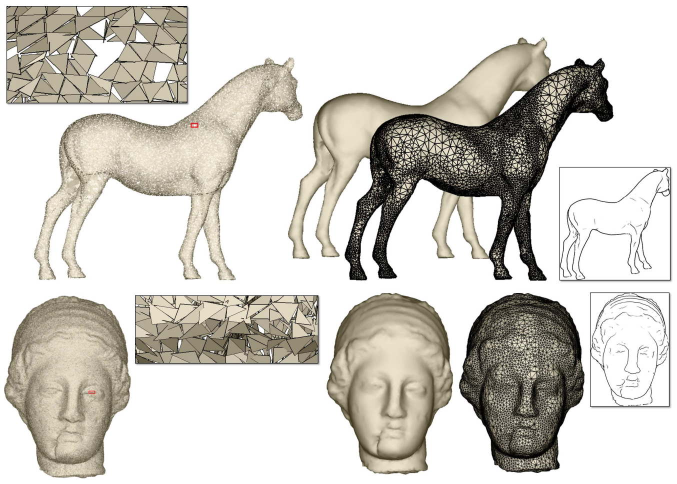


Fig. 9. Remeshing results of the polygon soup models: (top row) a horse model with 204k vertices are remeshed into a two-manifold triangular mesh with 10k vertices, and (bottom row) an Igea model with 806k vertices can be remeshed into a mesh with 10k vertices while preserving the visual saliency.

preserve visual saliency on a remeshed model.

Quantitative statistical analysis and comparisons are given in Table I. Using the non-photorealistic rendering result of input models as the reference, the rendering results of models remeshed by different methods are compared. The quantitative similarity between the images are measured by an image quality comparison metric simulating *Human Vision System* (HVS) in [10]. A higher image similarity indicates a model with more visual perceptional features remained. Moreover, the comparisons of time are also listed in the table. When the input model has a large number of vertices, our method is much faster than other methods as many steps of our remeshing pipeline are highly parallelized and run on GPUs. In summary, our method gives the best remeshed results (with respect to visual saliency) in the fastest speed. On the other hand, the geometric error analysis is also given in Table I using the publicly available Metro tool[2]. From statistics, we can find that the geometric error is not consistent with the visual saliency measurement. The mean and max errors of our approach are similar but greater than that of QEM [11] and saliency-based method [12]. This also proves that for applications focusing on the quality of rendering results, visual cues may not be fully represented by existing geometric error metrics (e.g., [11], [12]).

Moreover, this remeshing framework can also be applied to the mesh models having topology problems (see the zoom-views of polygon soup models in Fig.9). This property is provided because our algorithms do not strongly rely on the local connectivity of the input models. Figure 9 shows the remeshing results generated on two polygon soup models.

VII. Conclusion and Future Work

In summary, we proposed a visual perception guided surface remeshing framework for general polygon models. Firstly, the visual saliency extractions in different views are conducted

[2]http://vcg.isti.cnr.it/activities/surfacegrevis/simplification/metro.html

in the image space, which are mapped to 3D saliency points thereafter. After that, a scalar saliency field is constructed, where the field is conducting the resampling pattern adapted to visual requirements. Finally, the positions of sample points are optimized and the new mesh is reconstructed. In this paper, we exploit features of graphics hardware to accelerate the computation. Moreover, the remeshing framework based on sample points has many tasks being able to run in a highly parallel manner on GPU. As a result, our remeshing framework can generate visual perception guided results in a very efficient way on a variety of models.

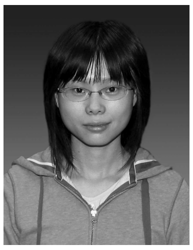
In this paper, the visual saliency measurement is extracted in the image-space, so that the selections scheme of image snapshots will influence the subsequent visual saliency measure. Six orthogonal views are used to generate the snapshots for the subsequent visual saliency extraction in 2D image space, which may miss important features that cannot be seen in the orthogonal views. At this implementation, the orientation of a model is interactively selected by users. After specifying the orientation of a model in the front-view, all the rest five views can be generated automatically. One of our future work is to automate the orientation selection by using visual cues. Techniques for adaptive viewpoint selection will also be explored in our future work to further improve the fidelity of models processed by our remeshing pipeline. Lastly, any other extraction algorithms based on visual perception cues can be easily plugged into our framework – i.e., the framework is not limited by the particular type of perception cues.

Acknowledgment

This research is supported by the HKSAR RGC/GRF Grant (Ref.: CUHK/417109) and CUHK Direct Research Grant (Ref. CUHK/2050518 and Ref. CUHK/2050492). The horse polygon soup model is used by the courtesy of Tao Ju. The Kitten, Rockerarm, MaxPlanck and Cane models are downloaded from the AIM@SHAPE shape repository.

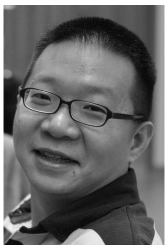
References

[1] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive contours for conveying shape," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 848–855, 2003.
[2] X. Gu, S. J. Gortler, and H. Hoppe, "Geometry images," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 355–361, 2002.
[3] V. Surazhsky, P. Alliez, and C. Gotsman, "Isotropic remeshing of surfaces: A local parameterization approach," in *Proceedings of 12th International Meshing Roundtable*, 2003, pp. 215–224.
[4] G. Turk, "Re-tiling polygonal surfaces," in *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, 1992, pp. 55–64.
[5] S. Fuhrmann, J. Ackermann, T. Kalbe, and M. Goesele, "Direct resampling for isotropic surface remeshing," in *Proceedings of the Vision, Modeling, and Visualization Workshop*, 2010, pp. 9–16.
[6] Y. Miao, R. Pajarola, and J. Feng, "Curvature-aware adaptive resampling for point-sampled geometry," *Computer-Aided Design*, vol. 41, no. 6, pp. 395–403, 2009.
[7] F. Cole, K. Sanik, D. DeCarlo, A. Finkelstein, T. Funkhouser, S. Rusinkiewicz, and M. Singh, "How well do line drawings depict shape?" in *Proceedings of ACM SIGGRAPH 2009 papers*, 2009, pp. 28:1–28:9.
[8] W. Hörmann, J. Leydold, and G. Derflinger, *Automatic Nonuniform Random Variate Generation*. Springer Berlin Heidelberg, 2004.
[9] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or, "Consolidation of unorganized point clouds for surface reconstruction," *ACM Transactions on Graphics*, vol. 28, no. 5, pp. 176:1–176:7, 2009.
[10] Y.-J. Liu, X. Luo, Y.-M. Xuan, W.-F. Chen, and X.-L. Fu, "Image retargeting quality assessment," *Computer Graphics Forum*, vol. 30, no. 2, pp. 583–592, 2011.
[11] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997, pp. 209–216.
[12] C. H. Lee, A. Varshney, and D. W. Jacobs, "Mesh saliency," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 659–666, 2005.



Lianping Xing received the BSc degree in computer science from Harbin Institute of Technology, China and the MS degree in computer science from Tianjin University, China. She is currently a PhD candidate in the Department of Mechanical and Automation Engineering, at The Chinese University of Hong Kong. Her research interests include computer graphics, CAD and geometric processing.

Xiaoting Zhang is currently a Ph.D. student at the Department of Mechanical and Automation Engineering, the Chinese University of Hong Kong. She gained her M.Phil. degree in Mechanical Engineering and Automation from the Harbin Institute of Technology. Her current research interests include geometric modeling, image processing and computer vision.



Charlie C.L. Wang is currently an Associate Professor at the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong. He gained his B.Eng. in Mechatronics Engineering from Huazhong University of Science and Technology, M.Phil. and Ph.D. in Mechanical Engineering from The Hong Kong University of Science and Technology. He is a Fellow of ASME. Dr. Wang serves on the editorial board of a few journals including Computer-Aided Design, ASME Journal of Computing and Information Science in Engineering, and International Journal of Precision Engineering and Manufacturing. His research interests are geometric modeling, design and manufacturing, and computational physics.

Kin-Chuen Hui received his B.Sc and Ph.D in Mechanical Engineering in 1979 and 1990 respectively from the University of Hong Kong. Before joining the Chinese University of Hong Kong in 1992, he was a consultant in the CAD Services Centre of the Hong Kong Productivity Council. He is currently a Professor of the Mechanical and Automation Engineering Department at the Chinese University of Hong Kong, and is the director of the Computer-Aided Design Laboratory. He is an editorial board member of the Journal of Computer-Aided Design. He is a member of the Institution of Mechanical Engineers, British Computer Society, Hong Kong Computer Society, and a fellow member of the Hong Kong Institution of Engineers. He worked closely with the local industry, and was a founding member of the Hong Kong Game Industry Association. He served as Chairman of the Hong Kong Game Industry Association in the period 2009 - 2012, and is currently serving as Honorary Consultant of the same association.