myjournal manuscript No.
(will be inserted by the editor)

Visual Saliency Guided Textured Model Simplification

Bailin Yang · Frederick W.B. Li · Xun Wang · Mingliang Xu

Received: date / Accepted: date

Abstract Mesh geometry can be used to model both object shape and details. If texture maps are involved, it is common to let mesh geometry mainly model object shapes and use texture maps modelling most object details, optimising data size and complexity of an object. To support efficient object rendering and transmission, model simplification can be applied to reduce the modelling data. However, existing methods do not well consider how object features are jointly represented by mesh geometry and texture maps, having problems in identifying and preserving important features for simplified objects. To address this, we propose a visual saliency detection method for simplifying textured 3D models. We produce good simplification results by jointly processing mesh geometry and texture map to generate a unified saliency map for identifying visually important object features. Results show our method offers a better object rendering quality than existing methods.

Keywords visual saliency · textured model · model reduction · simplification · texture space optimization

Bailin Yang
School of Computer Science & Information Engineering, Zhejiang Gongshang University, China

Frederick W.B. Li
School of Engineering and Computing Sciences, University of Durham, United Kingdom

Xun Wang
School of Computer Science & Information Engineering, Zhejiang Gongshang University, China

Mingliang Xu
College of computer science, University of Zhengzhou, China

1 Introduction

Textured three-dimensional (3D) objects are widely used in many graphics applications, including 3D games [44, 22], online 3D object virtual exhibition and virtual environments. Model simplification is a key approach to reduce modelling data of 3D objects, improving object rendering and transmission performance, which is particularly useful when a lot of objects are involved in an application.

Most existing model simplification methods are designed for 3D meshes, where they implicitly assume that object shapes and details are only modelled by meshes. Common approaches include minimising local surface distortion based on error metrics [10], preserving salient regions that stand out from their surrounding context [20], and applying perceptual principles [6]. However, texture maps are popular to involve in object modelling for the sake of improving realism, taking the responsibility for modelling most of the object details and letting mesh geometry focus on modelling object shapes. Therefore, mesh-based model simplification methods may become ineffective for reducing modelling data. A naïve approach to address this problem is adopting texture map simplification methods. However, popular methods such as DXTC (S3TC) and down-sampling (e.g. mipmapping) only consider the pixel information of a texture map without taking into account its relation to a 3D mesh. Even if we further consider mapping distortion [1] or projected size [11] of texture maps for simplification, as how mesh and texture map features jointly represent an object is not being considered, important object features may still not be able to identify and preserve. Therefore, significant visual distortion may be induced to simplified objects.

To support textured 3D object simplification with important features preserved, we have developed methods for identifying visual saliency and simplifying both mesh geometry and texture maps. Our main contributions include:

- An inverse mapping method is developed to allow mesh geometry, texture map, and the projection between them to be jointly processed, and that all captured saliency information can be consolidated into a unified visual saliency map to control textured 3D object simplification.
- A new local filter window is introduced to capture geometrically and topologically sufficient local mesh vertices for saliency evaluation, in which local features from both geometry and texture map domains can be properly captured even when objects are well simplified.
- We have introduced a local entropy feature identifying local saliency based on colour changes to enable capturing of visual features without complex signals.
- A novel relaxation and regularisation algorithm is developed to redistribute texel information allocating more texture map space to visually salient regions. This supports texture map optimisation, in which important features are preserved and induced distortion are minimised.

The rest of the paper is organized as follows. Section 2 discusses the existing work. Section 3 and 4 describes our new saliency detection method and depicts how it is applied to simplify textured meshes. Section 5 and 6 shows our experiment results and concludes our paper.

2 Related Work

Model simplification reduces object complexity, improving rendering and transmission performance and offering better user response time for graphics applications. As objects are typically modelled by 3D meshes and texture maps, methods have been developed to simplify these two different representations accordingly, which mainly comprise mechanisms for reducing object data and criteria for minimising the deviation of a simplified object from its original one. As visual quality of graphics outputs has been increasingly catching people's attention, researchers recently focus more on developing techniques to preserve object features during simplification according to human perception.

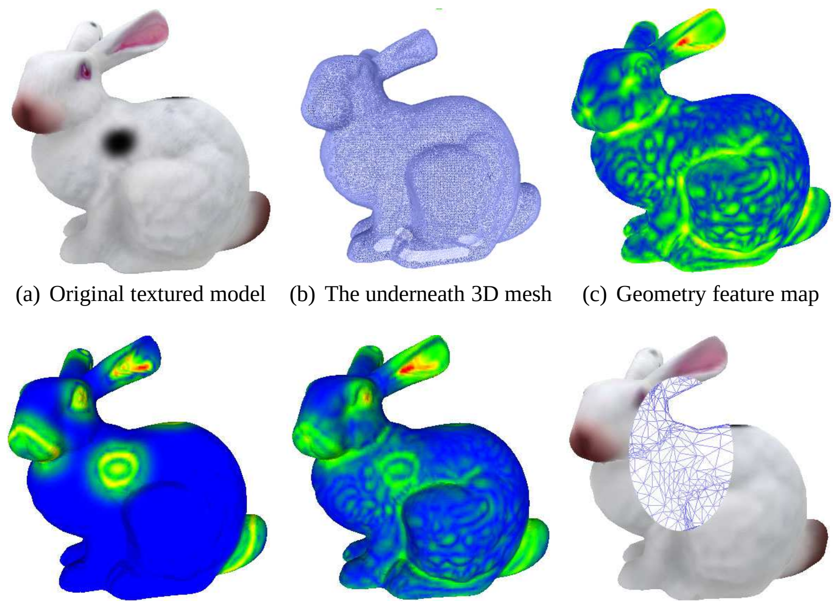
2.1 Model Simplification

To simplify 3D meshes, progressive meshes [12] proposed edge collapse and vertex split operations to modify the data size of a 3D mesh by adding or deleting mesh vertices and adjusting the connectivity of the affected vertices. Quadric error metric [10] was then developed to efficiently measure cumulative surface distortion introduced by mesh simplification. Methods developed at this stage focused on minimising the geometric error induced by simplification. However, such an approach does not consider how human perceive an object. For instance, it may make no difference for a user to perceive an object in either its original or simplified form, when the environment lighting is very low in contrast. Therefore, researchers turned their focuses on generating similar rendered results for simplified objects against their original ones rather than focusing only on geometric difference minimisation. Luebke and Hallen [26] adopted the contrast sensitivity function (CSF), developing an empirical perceptual model to measure the human perceived quality of the rendered output from a simplified object. Qu and Meyer [34] developed a visual masking technique based on the Sarnoff visual discrimination metric and the visual masking tool in JPEG 2000, supporting mesh reduction according to surface texturing, light variation, surface reflectance, etc. Menzel and Guthe [28] alternatively developed a perceptual model by measuring vertex-based BRDF parameterisation of a rendered model. It also considered contrast as a key factor to adjust the acceptable visual difference of a local region of a simplified object from its original one.

In terms of texture maps simplification, a de facto standard DXTC (S3TC) [14] divides an image into 4×4 pixel blocks, where all pixels in each block are approximated by two base and two derived colours. Similarly, *i*PACKMAN [38], being a part of the OpenGL ES API, reduces image data by approximating each pixel in a block with certain base colours. It additionally stores luminance information for modulation in order to enhance output image quality. Alternatively, texture map simplification can be done by down-sampling. A popular method is mip-mapping, which reduces the size of an image with simple fixed ratios regardless the image content. In contrast, [4] conducted user studies to determine the proper amount of down-sampling applying on different images. Besides the global methods as described above, texture map simplification can also be performed to local regions. [13] recursively resized texture map regions by balancing the amount of frequency content in each region. [1] determined the frequency signal in local texture map regions, stretching or shrinking a region based on its importance indicated by the frequency signal. [27] considered the projected model-space area of a texture map region to decide its degree of down-sampling and additionally performed a perceptual down-sampling process to reduce regions with less varied image content. Interestingly, although the texture map distortion induced by simplification may significantly affect a model's visual appearance, this has not been studied extensively. In addition, existing methods only directly relate the importance of a texture map region to the image colour and complex (or high frequency) content, but this may not always be the case as revealed in Fig. 8.(c).

2.2 Visual Saliency

Visual saliency detection is a bottom-up (stimulus-driven) approach to model human visual attention. It analyses the


(a) Original textured model (b) The underneath 3D mesh (c) Geometry feature map

(d) Gray entropy feature map (e) Final saliency map (f) Simplified textured model

Fig. 1 A bunny model and its visual saliency. (a) Texture colours are used to model eyes, mouth and black dapples on the body. (b) Geometric curvature variation is used to model eyes, mouth and legs. (c) The geometry feature map generated from the 3D mesh shows that black dapples are not captured. (d) Gray entropy feature map can clearly identify eyes, mouth and black dapples. (e) The final saliency map generated can capture both features modelled by mesh geometry and texture maps, where warmer colours (reds and yellows) indicate higher saliency and cooler colours (blues) indicate lower saliency. (f) A simplified textured bunny model is generated with our method.

visual characteristics of each element in a content domain to locate important parts [5]. It is rooted in image processing [32,16] and has been extended to mesh processing [20]. Specifically, visual saliency detection methods look for local features, which stand out from their surrounding context.

To detect salient regions of a 3D mesh, mesh saliency [20] applied the center-surround mechanism [16]. It is done by evaluating distinct parts of a local region in a domain with multiple scales to identify important domain features, together with applying the feature integration theory to process surface curvature information. To enhance the result, Gal et al. [9] additionally incorporated shape parameters, such as variance in curvature, number of curvature changes and size of a salient region relative to the entire mesh, to perform saliency detection. Thereafter, Liu et al. [24] applied Morse theory to handle animated or noisy meshes. Bulbul et al. [2] considered motion, colour and luminance information of a mesh for evaluating saliency. However, some object features may come from texture maps, such as the black dapples of the bunny model shown in Fig. 1. To process texture maps, Balmelli et al. [1] designed a local frequency map based on wavelet analysis to extract texture map saliency and carried out relaxation to optimize a texture map. Tang et al. [39] considered texture mapping distortion, texture sample reusability and saliency information generated from texel colour and luminance to compress a texture map. Yang et al. [43] considered texture map as part of the features of a textured 3D model and defined saliency-based feature maps for the texture map in order to support model simplification. Recent methods [21,37,3] attempted to detect mesh saliency by incorporating global features, which analyse vertex distance, object structure (limb-like vs. round shape), shape characteristics through log-Laplacian spectrum, and user selected schelling points. Alternatively, [11] made use of the number of visible texture map segments and their projected sizes to formulate visual saliency.

Saliency detection has many applications. It can improve 3D model examination by evaluating view-points through Jensen-Shannon divergence [7] to reveal important model features. It can also identify and emphasise important parts of a volumetric model [18], or can modify the normal vector of each mesh vertex according to the variation in its luminance in order to enhance the visual appearance of important mesh features [30]. One step further beyond important feature identification is applying saliency detection to illustrate objects with the extracted features, supporting object visualisation. For instance, [29] extracts view-dependent ridge-valley feature lines to form object illustration. On the other hand, saliency detection can be generalised to identify important 3D scene parts by processing model space motion and depth information as well as image space colour and luminance information [25]. A further application is real-time tracking of visually attended 3D scene models, where [24] performed this based on both image features (colour and luminance) and 3D model features (depth, model size, and model motion).

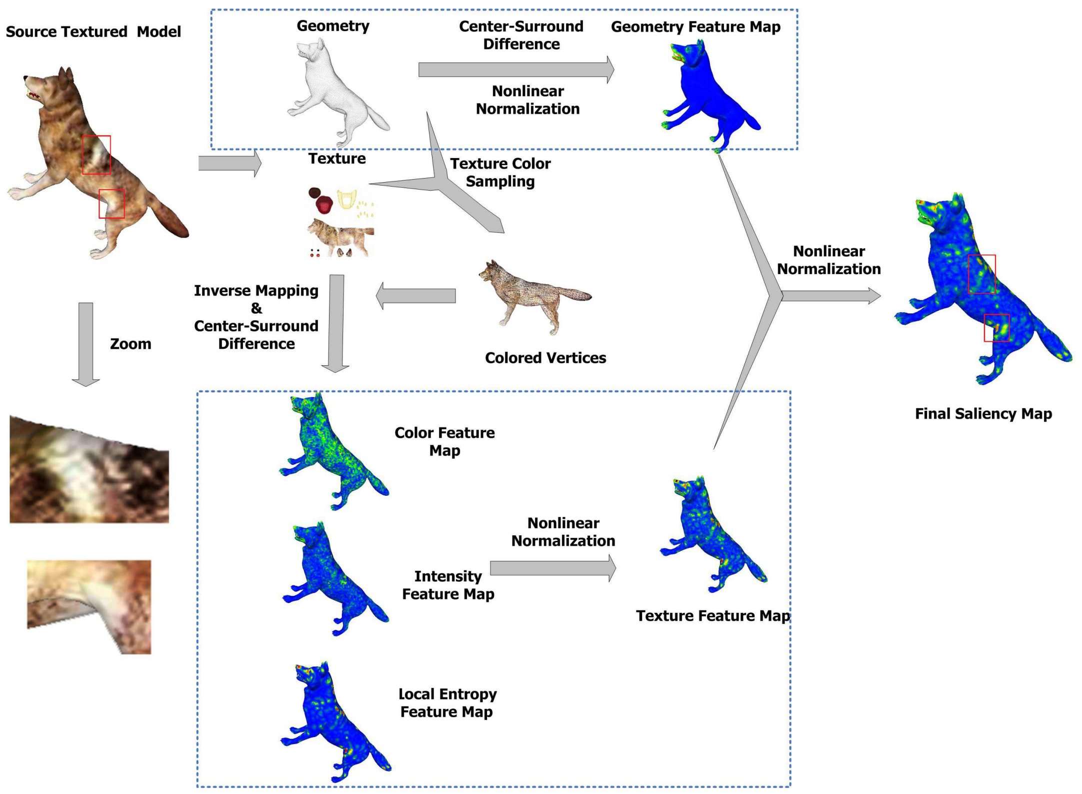


Fig. 2 Saliency region detection for textured model.

Saliency-based approach is attractive as existing methods demonstrated its possibility of identifying important features from both the mesh and the texture map domains. However, it is challenging to properly perform model simplification on textured objects, as mesh and texture map information should be treated as an integral part rather than separate components for identifying important object features properly. In addition, we should be aware that object features may come from either mesh geometry or texture maps or their combinations, and how much features come from each type of modelling data is object dependent, which is decided by an object designer. Finally, important object features may not be necessarily represented by complex signals.

3 Our Method

3.1 Overview

Our method aims to simultaneously process mesh geometry and texture map, generating a unified saliency map for properly identifying important object features to support model simplification. Fig. 2 depicts the saliency map construction process. As shown in the upper part of the figure, we perform center-surround difference on the mean curvature of mesh vertices to produce a geometry feature map. To capture how the object features modelled by the texture map builds on top of the mesh geometry, we inversely project the 3D mesh to the texture map, determining the distortion induced by texture mapping. The result is encoded as a texture distortion map. As shown at the bottom part of the figure, we determine object features from the texture map based on intensity, colour and local entropy [17] and produce corresponding feature maps. Finally, we apply nonlinear normalisation, combining all the above feature maps and generating the final saliency map. We implement model simplification by adapting the QSlim algorithm [10] to support mesh reduction and introducing a new relaxation and regularisation algorithm to support texture map optimisation.

3.2 Local Feature Extraction

Identifying locally distinct object features is the core idea of a visual saliency based method. Such features can be properly determined if sufficient local object information can be collected for processing. In image processing, a simple square filter window is sufficient for identifying locally distinct colour features, due to the regular connectivity of image pixels. Existing mesh saliency methods, such as [20], extend such an idea to collect local mesh features using a spherical filter window, where the radius r is determined by the diagonal length of the object bounding box. However, since mesh connectivity is usually irregular, where long and narrow triangles may exist (ref. Fig. 3), a spherical window may fail including all topologically connected neighbouring vertices $NS(V)$ of a candidate vertex V for processing. For example, only the three blue vertices in Fig. 3 are included and the green vertices are not, even they are directly connected to V. Consequently, local geometry features, such as mean curvature, cannot be properly determined.

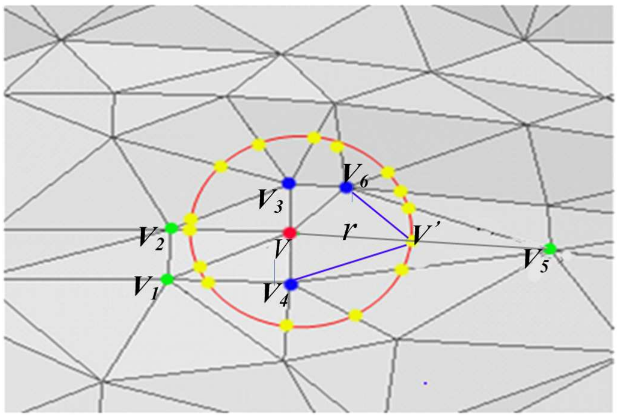


Fig. 3 Filter window construction

To address this problem, we define a new local filter window $W_r(V)$ based on both geometric distance and mesh connectivity to capture sufficient vertex samples for each V. As shown in Fig. 3, besides the three direct connected vertices (blue vertices) captured by a typical spherical filter window, $W_r(V)$ additionally captures the intersection points (yellow vertices) between the spherical filter window and the edges connecting to V, and the topologically connected neighbouring vertices (green vertices). The new local filter window allows us to capture more relevant local model information. This improves the quality of the locally distinct mesh features being determined. More importantly, this also allows the new local filter window to properly capture locally distinct features from both the texture map and its distortion induced by texture mapping. Despite our new filter window can capture local object features more precisely, which is optimal for simplification purpose, as it is mesh triangle size dependent, there might be a concern of being inconsistent in saliency computation on different detailed versions of an object from the perceptual point of view. This issue is not fundamental to our proposed method, but it is worth investigating further.

To estimate an intersection point, we take a Gaussian length-weighted average on its one-ring neighbouring vertices, e.g. the one-ring neighbouring vertices of the intersection point V' are V, V_4, V_5, V_6. Such an average produces better estimated intersection points than using a simple interpolation [19], as it assigns higher weights to vertices with higher proximity. An intersection point V' is computed as:

$$V' = \frac{1}{\sum_{i=1}^{n} W_i} \sum_{i=1}^{n} W_i \cdot V_i \tag{1}$$

where V_i are the one-ring neighbouring vertices of V'. Each weight $W_i = e^{-kd_i^2}$, representing the contribution of each neighbouring vertex to V', and is inversely proportional to the distance d_i between V_i and V'. In our implementation, the Gaussian coefficient k are empirically set to 1, 1.2, 1.5 and 2 at four scales for generating good results. Our new local filter window ensures sufficient local information can always be captured for processing, particularly when a mesh is simplified, having relatively sparse vertices.

3.3 Geometry Feature Computation

To capture important geometry features from a mesh, we perform Gaussian-weighted average on mean curvatures of mesh vertices under different mesh scales, producing scale-dependent feature maps and combining them into a geometry feature map. With the new local filter window, the geometry feature map construction is summarised as follows:

– **Step 1:** Compute the mean curvature $\xi(V_i)$ at each vertex V_i ($i = 1...n$, where n is the number of mesh vertices) based on [40].
– **Step 2:** Determine the neighbouring vertex set $NS(V_i)$ for each vertex V_i by the local filter window, comprising all vertices embraced in the filter window, intersection points between the filter and the mesh, and all topological directly connected neighbouring vertices of V_i.
– **Step 3:** Calculate the Gaussian-weighted average $GW(V_i)$ of $NS(V_i)$ at different scales.
– **Step 4:** Compute a scale-dependent feature map $G_l(m,n)$ by taking the difference between two $GW(V_i)$, which are generated at two scales m and n.
– **Step 5:** Apply the nonlinear normalisation [16] to combine all $G_l(m,n)$ forming the final geometry feature map $\overline{G}$. This step preserves only features that are important throughout all different mesh scales.

Note that the Gaussian-weighted average of the mean curvature at V_i is computed by:

$$GW(V_i, r) = \frac{\sum_{x \in NS(V_i)} \xi(x) \exp[-\|x - V_i\|^2/(2r^2)]}{\sum_{x \in NS(V_i)} \exp[-\|x - V_i\|^2/(2r^2)]} \tag{2}$$

and the absolute difference between the Gaussian-weighted average of a pair of fine and coarse scales, (r_m, r_n), in ϖ is:

$$G_l(m,n) = |GW(V_i, r_m) - GW(V_i, r_n)| \tag{3}$$

Since many 3D objects are not sphere-like or even extremely irregular in shape, it is not satisfactory if we follow [20], simply using the bounding box diagonal of a 3D mesh as the radius r to define the spherical filter window. Instead, we compute r based on the triangle sizes of a mesh. We determine cr, which is the radius of the circumcircle of the equilateral triangle with area $A(T_i)$, where $A(T_i)$ is the average area of all triangles in the mesh. Consequently, we define $r = 2^l \cdot cr$ as the radius of the spherical filter window where $l \in 1..4$, and obtain four filter window pairs ϖ:$\{2cr, 4cr\}$, $\{4cr, 8cr\}$, $\{8cr, 16cr\}$, $\{16cr, 32cr\}$.

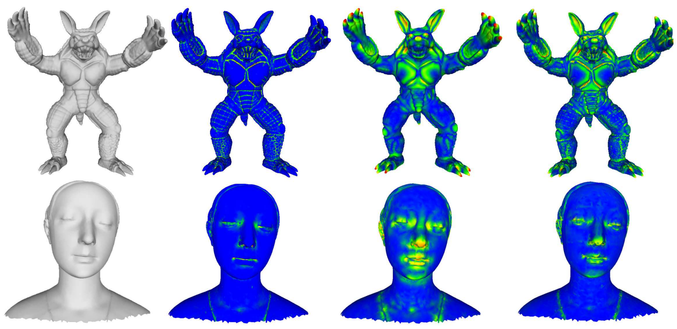


Fig. 4 Comparison among different salient region detection methods based on two 3D meshes: Armadillo and Laurana. The first column shows the original meshes. The other columns show the results from Mean Curvature, Lee's method [20], and our method, respectively.

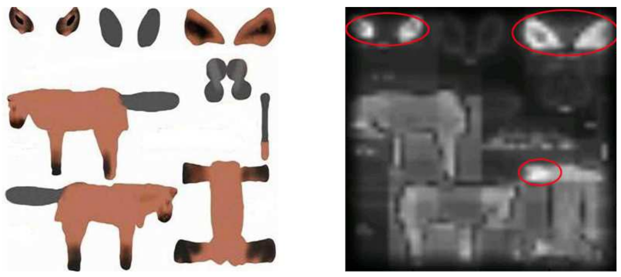


Fig. 5 Left: The texture map of a horse model. Right: Corresponding region saliency detected by Itti's method [16]. Regions picked out by red circles have stronger saliency, being higher in brightness.

Fig. 4 shows the saliency maps generated by different methods. It shows applying mean curvature alone cannot capture some salient regions, e.g. the knee of the armadillo. [20] can identify more object features, but the salient regions obtained are usually too large to be precise. In contrast, as demonstrated by the detected armadillo's knee and the eyes, mouth and nose of the Laurana model, our method identifies object features more precisely since we involve more locally relevant vertices for saliency computation.

3.4 Texture Feature Computation

Texture map adds visual features to a 3D object by using colours and patterns. For example, in Fig. 6, several colours are used to form circular patterns on the texture map representing the eyes of the girl model, while the corresponding mesh geometry underneath does not make obvious contribution to represent the eyes. On the other hand, there is a white fur stripe characterising the wolf model as shown in Fig. 7. The stripe is surrounded by areas with complicated patterns but itself is simple in colour. Most existing saliency detection methods [16] fail identifying the white fur stripe as important, since they consider region importance is mainly characterised by local colour difference. To address the problem, we propose using both colour information and local entropy to identify object features. We also allow people to control the weight of each type of these information when detecting saliency against objects with different designs. Introducing local entropy is also critical, since a texture map is not always continuous in content but is usually constituted of separate sub-textures as shown on the left hand side of Fig. 5, making typical saliency detection methods incapable to properly identify salient regions. On the right hand side of the same figure, it shows that Itti's saliency method [16] can only identify horse eyes and legs as salient, which is far from enough. Besides texture content, our method also consider texture mapping distortion as a factor to determine saliency of texture map regions, which is particularly critical when an object is being simplified.

Inverse Mapping: Our method treats 3D mesh vertices as the basis reference when computing saliency maps from different domains (mesh geometry or texture map) or between domains (projection between mesh and texture image). This makes producing a final unified saliency map feasible when different domains of information are involved, and particularly helps incorporate texture mapping distortion for evaluating model saliency.

Inverse mapping is an operation for computing how a texture map is being referred by the relevant 3D mesh vertices. As depicted in Fig. 6, for each vertex V_i, we capture its corresponding texel TE_{V_i} together with the surrounding texels TE_x by performing inverse mapping through the local filter window. In the figure, a textured girl model in the 3D space shows on the top-left, where its inverse mapping to the 2D texture space shows on the top-right. The close-up views of the left eye of the girl model in its 3D space and the inversely mapped 2D texture space are shown at the bottom of the figure. These views show that the triangles $\{T_n + T_e\}$ enclosed by the sphere s are inversely projected to become new triangles $\{T_n^{'} + T_e^{'}\}$ enclosed by the ellipse e. The computation is done as follows:

– **Step 1:** For each candidate vertex V, we apply a sphere S with center V and radius r to capture a set of intersection points, using these points to generate a set of triangles $\{T_n\}$. We also obtain a set of triangles $\{T_e\}$ connecting vertex V.

– **Step 2:** We then inversely project $\{T_n\}$ and $\{T_e\}$ from 3D geometry space to 2D texture space, forming a new triangle set $\{T_n^{'}+T_e^{'}\}$, which is embraced by the ellipse e with center $V^{'}$ (ref. Bottom-right of Fig. 6).
– **Step 3:** For each triangle $T^{'}$ in $\{T_n^{'}+T_e^{'}\}$, we apply the classical edge walking algorithm [8], which is usually used in the rasterisation procedure of a 3D graphics pipeline, to obtain texel samples. Number of texel samples obtained increases with the triangle area. As shown at the bottom-right of Fig. 6, the red dots represent some obtained texel samples of the triangle $T^{'}$.
– **Step 4:** We then obtain all texel samples inside the ellipse e by repeating Step 3, and perform the Gaussian weighted filtering operation on these texel samples.

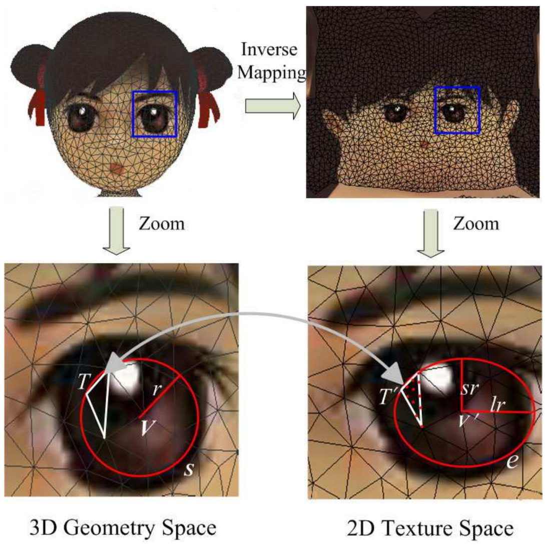


Fig. 6 Inverse mapping and texture sampling.

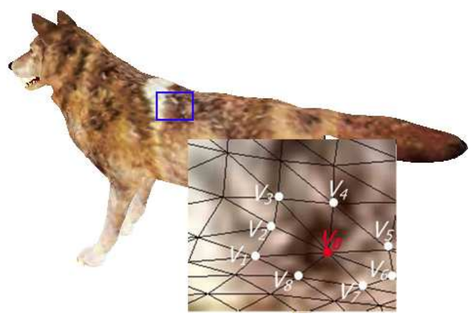


Fig. 7 Textured wolf model: In the highlighted part, V_0 is a candidate vertex and $\Delta V_0 V_i V_{i+1}$ $(i \in 1 \ldots 7)$ are its connecting triangles, where texture colour arbitrarily varies over each triangle.

Visual Saliency: We compute visual saliency by both attributes associated with individual texels and patterns (attribute changes) within a local region into account. To implement, we pick intensity and colour as the individual texel attributes to collect, while we evaluate pattern by local entropy. We also perform bilinear interpolation to compute all saliency values across the surface of each projected triangle.

The intensity I of each texel is $I(TE_{V_i}) = \frac{r+g+b}{3}$, where r, g, and b are the red, green and blue colour values of each texel, respectively. The colour C is represented by red-green (RG) and blue-yellow (BY) opponencies [41]: $RG(TE_{v_i}) = \frac{r-g}{max(r,g,b)}$, $BY(TE_{v_i}) = \frac{b-min(r,g)}{max(r,g,b)}$, which can better interpret how human perceive colours. To avoid large fluctuations of the texture colour opponency values at low luminance, while $max(r,g,b) < 0.01$, the $RG(TE_{V_i})$ and $BY(TE_{V_i})$ are set to be zero.

To capture pattern, we introduce *Texel Gray Entropy* (GE) based on the Shannon entropy, which evaluates signal complexity by measuring the gray colour homogeneity of the texels surrounding a given texel TE_{V_i}. It is defined as:

$$GE(TE_{V_i}) = -\sum_{gc\in\ CG} P(gc)\log_2 P(gc), \tag{4}$$

$$P(gc) = \frac{\sum_{TE_j \in NT}(GR(TE_j) == gc)}{\sum_{NTE_{V_i}}} \tag{5}$$

where TE_j is a neighbouring texel around TE_{v_i} and NT is the $n \times n$ neighbouring texel set. $\sum_{NTE_{v_i}}$ is the number of NT associated with TE_{v_i}. There exist C different gray colour values, forming the gray colour set CG. Suppose gc is a gray colour in CG, $P(gc)$ is the probability of gc in NT, and $GR(TE_j) = 0.30r + 0.59g + 0.11b$ representing the gray value of each TE_j. We do not separately compute local entropy against each r, g, b, avoiding slow computation.

Texture feature map: Texture feature map is constructed as follows:

Firstly, we apply a 3×3 Gaussian filter to smooth a texture image, making it robust to noise.

Secondly, we perform inverse mapping to obtain both the corresponding texel TE_{v_i} and surrounding texels TE_x for each mesh vertex, using the local filter window. In contrast to typical methods, such as [16], the filter is applied to the original texture map instead of its scaled-down versions for obtaining high quality features [31].

Thirdly, we compute the Gaussian-weighted average for each of the intensity I, colour $RG + BY$ and local gray entropy EG of each texel TE_{v_i} under different scales using the following equation:

$$XW(TE_{v_i}, r, lr, sr) = \frac{\sum_{TE_x \in W_{r,lr,sr}} ICT(TE_x)EXP}{\sum_{TE_x \in W_{r,lr,sr}} EXP}$$

(6)

where $EXP = exp[-\|TE_x - TE(v_i)\|^2/(2r'^2)]$, and r' is the distance from TE_x to TE_{v_i}.

$XW(TE_{v_i}, r, lr, sr)$ represents a function for computing the Gaussian-weighted average, which can be substituted by $IW(TE_{v_i}, r, lr, sr)$, $CW(TE_{v_i}, r, lr, sr)$ or $GEW(TE_{v_i}, r, lr, sr)$, which evaluates the Gaussian-weighted average of intensity, colour or local gray entropy, respectively. r is the radius of the local filter window (sphere) s, and lr, sr are the long and short radius of the projected filter window (eclipse) e (ref. Fig. 6). $ICT(TE_x)$ is a function of texel attributes, which can be substituted by $I(TE_x)$, $RG(TE_x)+BY(TE_x)$ or $GE(TE_x)$ when evaluating IW, CW or GEW, respectively.

Fourthly, we use the following equation calculating the sub-feature maps, including Intensity Feature Map $\overline{I_l(m,n)}$, colour Feature Map $\overline{C_l(m,n)}$ and Local Entropy Feature Map $\overline{E_l(m,n)}$. ICE is a function of Gaussian-weighted average, which can be substituted by IW, CW or GEW when evaluating $\overline{I_l(m,n)}$, $\overline{C_l(m,n)}$ or $\overline{E_l(m,n)}$, respectively.

$$\begin{aligned}\overline{ICE_l(m,n)} = |&ICE(TE_{v_i}, r_m, lr_m, sr_m)- \\ &ICE(TE_{v_i}, r_n, lr_n, sr_n)|\end{aligned} \quad (7)$$

where pair (r_m, r_n) is one of the pairs in ϖ.

Then, each of the $\sum_{l=1}^{4}\overline{I_l}$, $\sum_{l=1}^{4}\overline{C_l}$ and $\sum_{l=1}^{4}\overline{E_l}$ is normalised (denoted as $N()$). These normalised items are summed up forming the texture feature map $\overline{T}$,

$$\overline{T} = \gamma_1(N(\sum_{l=1}^{4}\overline{I_l})+N(\sum_{l=1}^{4}\overline{C_l}))+\gamma_2 N(\sum_{l=1}^{4}\overline{E_l}), (\gamma_1+\gamma_2=1) \quad (8)$$

Finally, we combine the texture feature map $\overline{T}$ and the geometry feature map $\overline{G}$ forming the final saliency S:

$$S = \lambda_1 N(\overline{T}) + \lambda_2 N(\overline{G}), (\lambda_1 + \lambda_2 = 1) \quad (9)$$

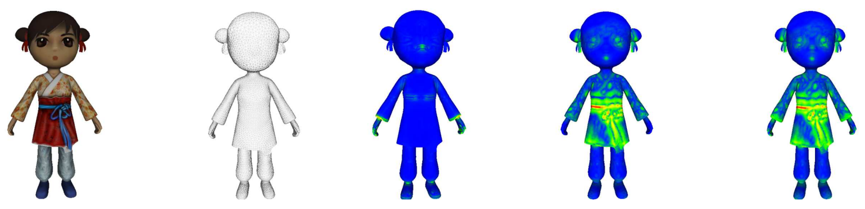
To produce proper texture feature maps $\overline{T}$ for different textured objects, we may manipulate the weights in Eq. 8. If an object mainly relies on colour information stored at individual texels modelling its details, e.g. the girl model in Fig. 8(a), we should increase the weight of γ_1. In contrast, if attribute changes play an important role of object modelling, e.g. the vase model in Fig. 8(c) has a distinguished white horse pattern, which stands out from its surroundings, we should increase γ_2 instead. In our experiments, γ_1 and γ_2 are set with one of the values 0.25,0.5 or 0.75. On the other hand, if the 3D mesh constructing an object does not possess too much object details, we should increase the weight of λ_1 in Eq. 9, as texture map consequently plays a more important role in object modelling. We set the values of λ_1 and λ_2 similar to those used for setting γ_1 and γ_2.

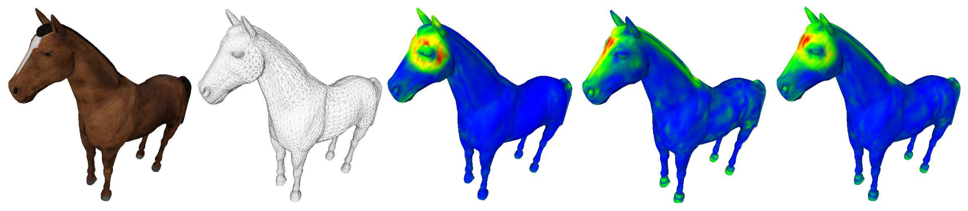
Discussions: Setting the above weights is not straightforward. For example, the bunny model in Fig.1 is dominated by its geometry features, such as ears, mouth and thigh regions, while its texture map does not significantly contribute to the final model representation, despite having some birthmarks on its body. In this case, λ_2 should be increased. In contrast, it is popular for online games to gain runtime performance by representing objects with simplified meshes but with good quality texture maps. Similarly, due to model design, if object features are mainly modelled by texture maps, e.g. the eyes of the girl model in Fig. 6, it means features from the texture map is more important than mesh features. In both cases, we have to set λ_1 larger. Alternatively, mesh and texture map importance may vary according to user attention, such as light-of-sight or object-user view distance. Therefore, generating weighting factors automatically may be complicated. Itti et al. [15] had investigated this problem and suggested using supervised learning as an approach to combine saliency information properly. However, it also found that achieving a good result in combining saliency information is scenario or application dependent. As we want to keep the generality of our method and a good application performance, we therefore adopt a simple weighted linear sum approach for combining saliency information. In addition, introducing the above weights makes our method capable in handling different types of textured objects. Future work should be carried out to investigate an automatic way setting the values of these weights.

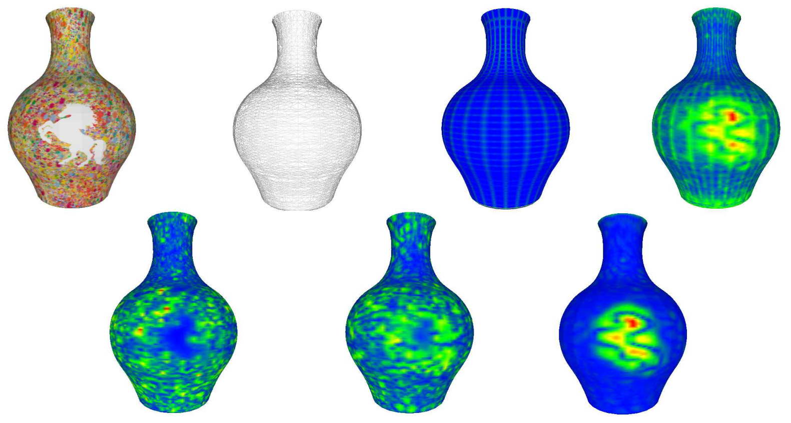
Experiment outputs: Fig. 8 shows some experiment outputs from our method. For each of the girl and the horse models as shown in Figs. 8(a) and (b), starting from left to right, we show the original textured model, the underneath mesh model, the generated geometry feature map, the texture feature map and final saliency map. For each of the vase and wolf models as shown in Figs. 8(c) and (d), we show in the first row the original textured model, the underneath mesh model, the generated geometry feature map and the final saliency map, respectively. The second row shows the saliency maps for intensity feature, colour feature and local gray entropy feature, respectively.

Fig. 8(a) shows that the head of the girl model is modelled by a smooth 3D mesh, not possessing rich curvature variance. If we simply consider the geometry feature map, we cannot capture the eyes, nose and mouth parts as important. Instead, our final visual saliency map, which incorporates texture features, can successfully identify these parts as important. Similarly, for the horse model as in Fig. 8(b), we have identified the black mane on its back and a white region on its head as salient regions.

When we consider the vase model in Fig. 8(c), which has a white horse pattern at the center over the vase's surface


(a) Girl


(b) Horse


(c) Vase

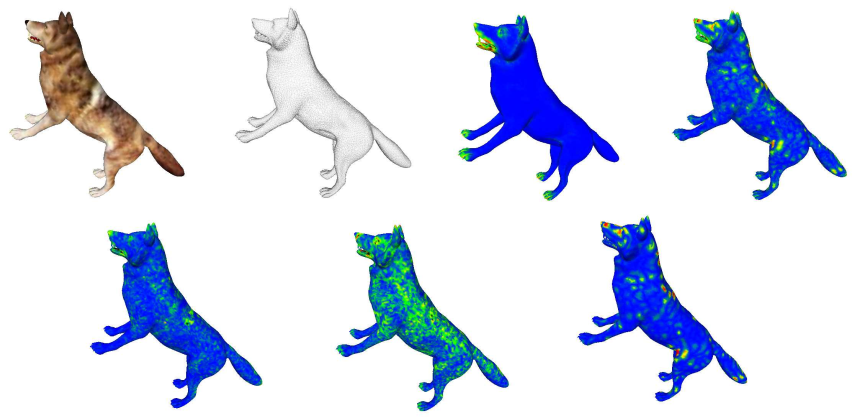

(d) Wolf

Fig. 8 Experiment results on four models. The figure shows the source textured model, its 3D mesh and different feature maps generated.

surrounding by a complex background, since the white horse is not complex in signal, it cannot be detected as salient if only intensity and colour features are considered as in existing methods. We additionally consider the local gray entropy feature, successfully detecting such a salient region. Our method is also successful in detecting the white fur stripes on the back and at the left groin of the wolf model in Fig. 8(d) as salient regions.

4 Saliency-Driven Model Simplification

Our method jointly considers features from 3D mesh and texture map for data reduction. To simplify the mesh geometry, we extend the model simplification method [10] by incorporating a visual saliency factor to determine the order of edge collapse. To simplify the texture map, we introduce a novel texture relaxation and regularisation method for reducing the texture map resolution, which is guided by our saliency detection method.

4.1 Mesh Simplification

We modify QSlim [10], weighting the quadrics by mesh saliency. Compute the weight W of each vertex V_i as:

$$W(V_i) = 1.0 + N(\omega(V_i))^{\zeta} \tag{10}$$

where $\omega(V_i)$ is calculated as:

$$\omega(V_i) = \begin{cases} \kappa S(V_i), & if \quad S(V_i) \geq \eta, \\ S(V_i), & if \quad S(V_i) < \eta \end{cases} \tag{11}$$

$S(V_i)$ is the saliency of each vertex V_i. In order to preserve vertices with larger saliency values throughout the process, $S(V_i)$ is amplified nonlinearly, and that when $S(V_i)$ of some vertices exceed certain values η, they will be amplified by a factor κ. In addition, we can change ζ to adjust the importance of geometry saliency in the final weight. The weight W will increase with the geometry saliency value $\omega(V_i)$. If the parameters κ, η and ζ are set larger, more salient vertices can be preserved during simplification. Practically, their values cannot go very large, otherwise the base mesh vertices cannot be preserved due to their low saliency values. We empirically set η, ζ and κ to be 30th percentile saliency, 3, and 20, respectively, for obtaining good results. In addition, $W(V_i)$ is normalised to within $[0 \ldots 1]$.

4.2 Texture Map Optimisation and Simplification

Texture map optimisation is a process to arrange more texture space holding visually important regions, allowing these regions to preserve longer during texture map simplification. The texture feature map introduced in the last section determines such important regions. However, a texture map can be distorted as its texels may be nonuniformly sampled across the mesh surface due to parameterisation. We then take texture map distortion into account, forming another factor to determine region importance. We have also developed a novel Relaxation & regularisation algorithm, which deforms a texture map allocating proper space to hold different texture map regions based on their importance, avoiding unexpected distortions as those appeared in [1].

4.2.1 Synthesis of Visual Importance Map

To evaluate the saliency value ρ of each texel, we inversely project all mesh triangles onto the 2D texture map domain and perform bilinear interpolation over the surface of each projected triangle. The output is a gray image called Saliency Texture Map (STM) as shown in Fig. 9(a).

Texture deviation is measured by a 3×2 Jacobian matrix [35], which produces singular values representing the amount of parameterisation distortions. When a singular value is smaller than 1, it indicates the parameterisation of the corresponding texture domain part is contracted and its sampling frequency being reduced. Here we denote a singular value as ϕ. The smaller the value of ϕ is, the greater the sampling frequency should be arranged for a triangle in the texture domain. We obtain the smallest singular value for each corresponding triangle of a vertex v, averaging these values to form the final singular value for v. We then execute a bilinear interpolation operation over the surface of each projected triangle in the texture map domain, obtaining all singular values to form the Texture Distortion Map (TDM) as shown in Fig. 9(b).

Finally, we combine both STM and TDM, forming a visual Importance Map (IM) with normalised weights as shown in Fig. 9(c). We can get the importance value of each texel by $\sigma(u, v) = N(\rho(u, v)) \cdot N(\phi(u, v))$. Fig. 9(c) shows that our final visual Importance Map of the horse model can recognise salient regions, such as the mouth, eyes, feet and mane on the back of the horse. We also recognise the mouth, nose and eyes of the human head model as visually important as shown in Fig. 10(e).

4.2.2 Relaxation and regularisation

A relaxation algorithm was proposed in [1] to reorganise texture image content by balancing the content frequency distribution. Fig. 10 (a) shows visually important regions are evenly distributed over the frequency map of the texture image of the human head's model, where texture parameterisation produced is supposed not to deform severely. We attempted to perform relaxation following [1] with our importance map (ref. Fig. 10(e)) describing the model saliency. Unfortunately, as shown in Fig. 10(d), severe artifacts are produced around the mouth part of the human head model, because some regions of the texture parameterisation produced are deformed heavily after relaxation (ref. Fig. 10(b)), leading unfavourable results.

To overcome this problem, we have developed an effective Relaxation & Regularisation algorithm, guiding by our visual importance map IM. Prior to relaxation, we divide texture map into several regular grids and obtain a visual importance value of each texel from the IM. As shown in Fig. 10(e), our importance map can obtain far more accurate salient regions, including the eyes, nose and mouth, than using the frequency map from [1] (ref. Fig. 10(a)). For relaxation, we shrink grid cells with visually unimportant texels and stretch those cells with visually important texels, reserving more space to visually important regions. At this point,

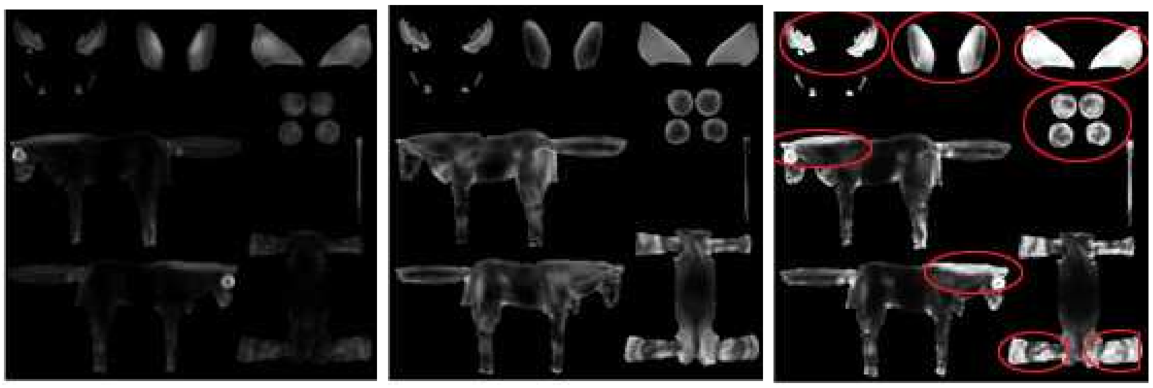
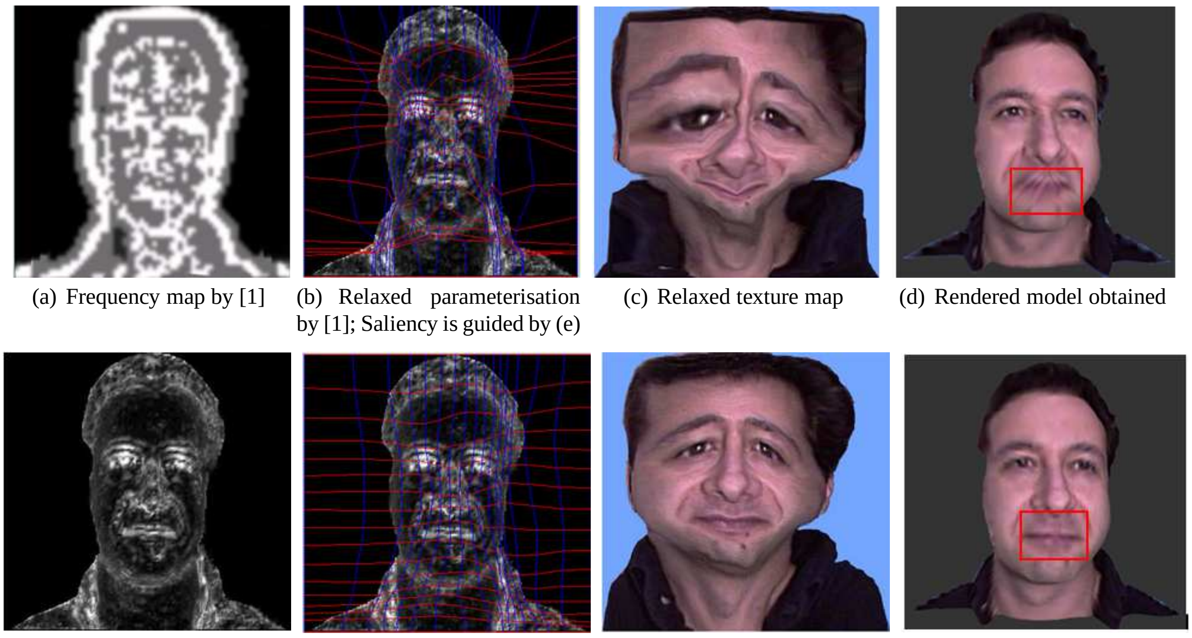

(a) STM (b) TDM (c) IM

Fig. 9 STM, TDM and IM for the horse texture map.


(a) Frequency map by [1] (b) Relaxed parameterisation by [1]; Saliency is guided by (e) (c) Relaxed texture map (d) Rendered model obtained

(e) Our importance map (f) Relaxed and regularised parameterisation (g) Relaxed texture map (h) Our rendered model

Fig. 10 This figure shows the frequency and the importance map of a model produced by Balmelli's method [1] and our method. It also shows the produced texture map parameterisation and image from both methods. Rendered models indicated our method produces a better result.

grid cells holding visually extremely important or unimportant texels may still be heavily deformed. To avoid such a problem, we introduce a geometrically motivated shape regularisation method. For regularisation, we treat the relaxed rectangular grid cells as rigid areas and carry out an as-rigid-as-possible deformation for the relaxed grid. As a result, the deformed grid cells will be adjusted making the entire grid more regular. As shown in Fig. 10 (f)-(h), grid cells after Relaxation & Regularisation did not severely deform, producing a better rendering result. We now describe the entire process in detail.

1) Relaxation

In our relaxation method, each grid cell of a texture map comprises some texels and four boundary vertices. The importance value σ for each boundary vertex can be obtained from IM. To relax the texture image, we minimise the following objective function:

$$E = \sum_{\{i,j\} \in e} WI_{ij} \|v_i - v_j\|^2 \tag{12}$$

where v_i and v_j are two boundary vertices of a grid cell, forming an edge e. WI_{ij} is the weighting factor determined by averaging the importance value σ of the quads sharing the edge e. While σ is constant, the energy function is a quadratic function of the grid cell vertex positions. Minimisation of this energy function can be transformed into a classical least-squares minimization problem. In our implementation, the relaxation algorithm is a successive approximation algorithm, such that the new position of a vertex V_i^{n+1} is determined by adding a displacement $\Delta(V_n)_i$ to the corresponding current position V_i^n:

$$V_i^{n+1} = V_i^n + \Delta(V_n)_i \quad i = 1 \ldots N \tag{13}$$

$$\Delta(V_n)_i = \sum_{j\in ni} \mu_{ij}(v_j - v_i) - \{\sum_{j\in ni} \mu_{ij}|(v_j - v_i)|^2\} \frac{\nabla\sigma(v_i)}{2\sigma(v_i)} \tag{14}$$

where $\mu_{ij} = \alpha_j\sigma(v_j)/\sum_{j\in ni}\alpha_j\sigma(v_j)$.

2) As-Rigid-as-Possible regularisation

For regularisation, we define a nonlinear energy function to preserve the rigidity of regular grid cells while approximating the deformed grid cells with their original shapes before relaxation. The algorithm is as follows:

Given a quadrilateral grid cell QG, the initial and deformed positions of its four vertices are vp_i and vq_i, where $i \in \{1 \ldots 4\}$. If the deformation of this grid cell is rigid, the optimal rigid transformation A is found by matching all initial vertex positions vp_i against the deformed vertex positions vq_i, such that:

$$vq_i - vq_c^i = A(vp_i - vp_c^i) \tag{15}$$

where vp_c^i and vq_c^i are the initial and deformed rotation centers. As we knew, the deformation could not be rigorously rigid. We find an optimal rigid transformation A that fits Eq. 15 in a least-squares sense, i.e. we minimise:

$$E(QG) = \sum_{i\in QG(i)} \|vq_i - vq_c^i - A(vp_i - vp_c^i)\|^2 \tag{16}$$

In fact, this energy function is a shape matching problem. In the 2D case, A has the analytical expression [36],

$$A = \frac{1}{\mu_s}\sum_{i\in QG(i)} (\hat{vp}_i \quad -\hat{vp}_i^{\perp}) \begin{pmatrix} -\hat{vq}_i^T \\ -\hat{vq}_i^{\perp T} \end{pmatrix} \tag{17}$$

where

$$\mu_s = \sqrt{(\sum_i \hat{vq}_i^T \hat{vp}_i)^2 + (\sum_i \hat{vq}_i^T \hat{vp}_i^{\perp})^2} \tag{18}$$

with $\hat{vp}_i = vp_i - vp_c^i$ and $\hat{vq}_i = vq_i - vq_c^i$, where $\perp$ is a 2D vector operator such that $(x, y) \perp = (-y, x)$.

3) Implementation The Relaxation and Regularisation algorithm is implemented as follows:

– **Initialisation**: Take the texture map image TI with size $i\times j$ as a regular mesh M and divide TI into $m\times n$ small regular grid cells G, where each covers $(i \times j)/(m \times n)$ texels. In our implementation, the size of $m \times n$ is assigned to be $1/16$ of $(i \times j)$.
– **Relaxation**: Relax each grid cell G of mesh M. For each vertex v_i^n of cell G, its relaxed position v_i^n is determined by Eqs. 13-14 above. If the $\Delta(V_n)_i$ is not changed significantly, we perform the **Mesh Adjustment** operation, otherwise we perform **regularisation**.
– **Regularisation**: After getting the relaxed mesh, we modify the positions of boundary vertices of each grid cell G with the as-rigid-as-possible regularisation method in Eqs. 16-18. In addition, we take the regularised grid as an input into the Relaxation phase.
– **Mesh Adjustment**: According to the alteration of grid cell vertices, adjust the regular mesh M forming an irregular mesh IrM.
– **Texel Projection**: Inversely project all texels covered by IrM into M through sampling. The sampling frequency for each single grid cell is equal to $(i \times j)/(m \times n)$. We can then obtain the new relaxed texture map image TI'. Note that we should record the mapping relation of the source image TI and the relaxed image TI'.

5 Experimental Results

We firstly compare the simplification results for 3D geometry models without texture maps obtained by QEM[10], mesh saliency [20] and our method. We then show the visual quality for textured models when we simplify the geometry and resize the texture image using three different texture resizing methods: 1) apply QEM and direct texture image resizing without texture space optimization ($QEMRD$), 2) apply mesh saliency [20] and texture image resizing with texture space optimization [1] ($SALOPT$), and 3) our method. To our knowledge, there is no existing work directly comparable to our method due to its uniqueness, and that we alternatively compare our method with QEMRD and SALOPT, which are based on combining a mesh and a texture map simplification methods, in order to improve the fairness of the comparison. Table 1 shows geometry face number and texture image resolution for the original and the reduced version of the testing models. Note that the vase, horse, girl, and wolf are textured models while the last two, Armadillo and Laurana, are purely mesh models without texture map.

5.1 Simplification for Mesh Model

Fig. 11 shows the simplification results of two mesh models obtained from: QEM [10], mesh saliency [20] and our method. The Laurana and Armadillo models are simplified to contain only 3% and 1% of polygons of their original models, respectively. The results shows that visually important regions, such as armadillo's mouth and knee, and Laurana's eyes, nose and mouth, are better preserved by our method, while the results from QEM [10] are the worst. This is because our method can successfully detect visually important regions as shown in Fig. 4. Our method also allow adjustments to the saliency weighting factor κ as in Eq. 11, making salient regions to preserve much longer than other model parts during simplification.

5.2 Simplification for Textured Model

This part presents results of a visual quality comparison and an objective measurement on the rendered outputs of some reduced textured models produced by $QEMRD$, $SALOPT$ and our method.

5.2.1 Visual Quality Comparison

For the comparison, we present screenshots of the rendering results of three different model reduction methods against four textured models (ref. Fig. 12). We found that our new method achieves better visual appearance for the textured models than the other two methods. In the vase and wolf models (ref. Fig. 12(a) and (d)), the white horse pattern at the center of the vase model and the white furs on the back of the wolf model are simple in colour. Since our salient detection method takes into account the local entropy feature, we successfully detect such a pattern as visually important. Consequently, the boundary of this horse pattern in vase model and the white furs on the back of the wolf model are well preserved even after model simplification. For the horse model (ref. Fig. 12(b)), we can also capture the white hair on the head as a salient region. In addition, for the horse, girl and wolf models (ref. Figs. 12(b)-(d)), our method also obtains eyes more clearly than $QEMRD$ and $SALOPT$, mainly because the curvature around the eyes of the girl, horse and wolf models does not vary significantly. Since $QEMRD$ and $SALOPT$ rely only on geometry information to capture salient areas, they do not regard these features as visually important and they will be easily lost during model simplification.

For texture image reduction, when applies $QEMRD$ and $SALOPT$, mapping errors appear on the legs of the horse model (ref. Fig. 12(b)) if texture image is resized aggressively, since such resizing will get rid of some texels. The mapping error is more serious for the horse model since the reduced versions of their corresponding texture images are small. As shown in Table 1, the texture image's size of the reduced horse model is just 32×32 pixels.

Specifically, as shown at the leg of the horse model (within the two red rectangles) (ref. Fig. 12 (b)), $QEMRD$ induces quite significant artifacts since it simply resizes texture images without considering any saliency information. In general, $SALOPT$ induces less error than $QEMRD$ because it takes saliency into account. However, the artifacts still appear in the horse model when $SALOPT$ is applied. This is because the texture optimization operation of $SALOPT$ causes severe deformation to the texture image due to a relaxation process and that some texels are lost as mentioned in section 4.2.2. In contrast, we have introduced a geometrically motivated shape regularisation method to adjust the results obtained from our relaxation process, minimising such a deformation problem. In addition, our method has reorganised a texture map making more texture space to hold visually salient texels. This helps reduce possible texture distortion as well as allowing visually salient texels preserved longer during model simplification.

Table 1 Geometry face numbers and texture image sizes (pixels) of the original and reduced versions of the test models

	Original Model	Reduced Model	Ratio(Reduced/Orignal)
Vase	33216	1800	5.4 %
	1024×1024	64×64	0.4%
Horse	18363	1181	6.4%
	512×512	32×32	0.4%
Girl	46368	3178	7.3%
	1024×1024	64×64	0.4%
Wolf	29892	2856	7.9%
	1024×1024	64×64	0.4%
Armadillo	346K	3445	1%
Laurana	129K	3870	3%

5.2.2 Objective Comparison

We adopt the peak signal-to-noise ratio ($PSNR$) and multi-scale structural similarity index($MS-SSIM$) [42] to objectively measure the rendered quality of the textured models. $PSNR$ is still the most commonly used metric for judging visual fidelity of images, but $MS-SSIM$, a kind of statistical metric, is considered the most accurate [33]. To provide a more comprehensive measurement, we take several images of each model from different viewpoints, computing $PSNR$ and $MS-SSIM$ values for all images and averaging these $PSNR$ and $MS-SSIM$ values to form a single error measurement for each model.

Before computing $PSNR$, $MS-SSIM$ and H, we should assign the reference model and the reference image set. In our experiment, we use the original model as the reference model. The reference image set is captured from this reference model. We carefully arrange the viewpoints, such that an even coverage of image samples is collected. We adopt the small rhombicuboctahedron [23] to evaluate $PSNR$ and $MS-SSIM$ values.

We also measure the colour distribution (CD) of the texture image used in a model, where a higher CD value in-

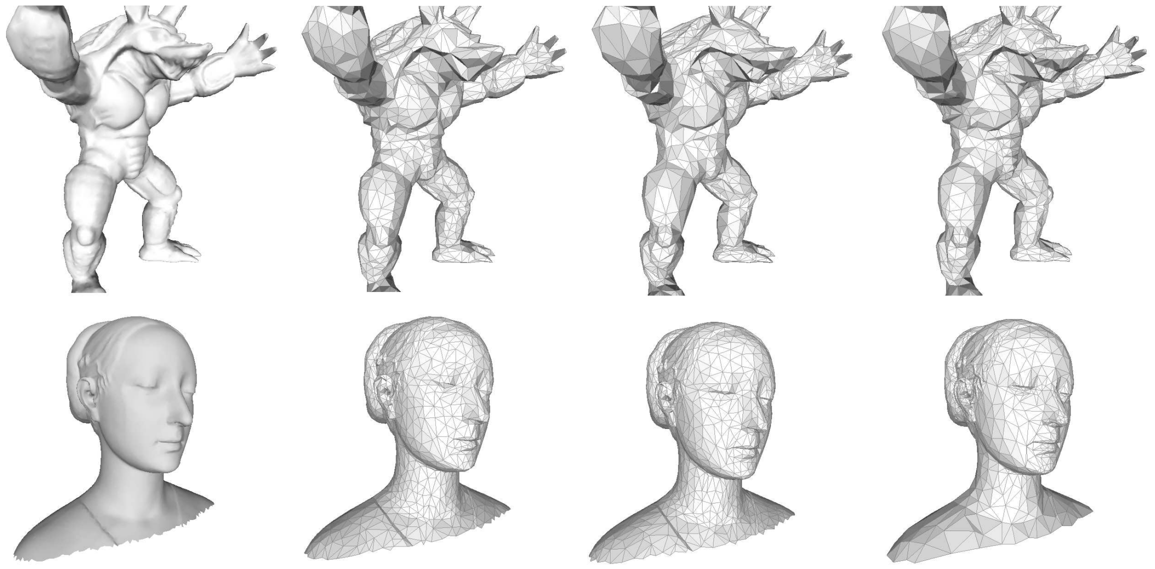


Fig. 11 Compare simplification results on the two models, Laurana and Armadillo (column 1), by running QEM [10], Lee's [20] and our method (columns 2 to 4, respectively).

dicates that the texture image comprises a wider range of colours, i.e. more complicated. For example, the vase model has a higher CD than the girl model, while the CD value is low for the horse and raptor models. To compute the CD, in the implementation, we scan through a whole texture image and find out all the colours on it.

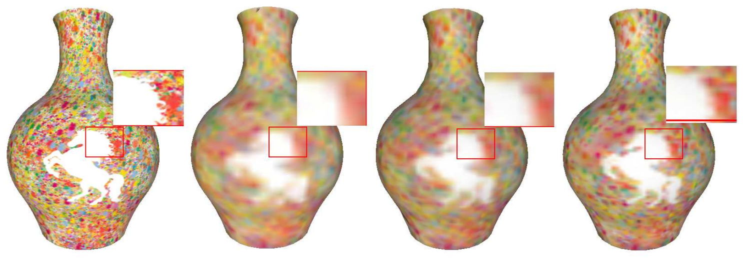
Table 2 shows the models produced by our method have larger $PSNR$ and $MS-SSIM$ values than those generated by the other two methods, meaning that our method performs better, as the rendered outputs of the simplified textured models produced by our method are very similar to the reference image set. Experiment results for the horse model show that $PSNR$ of our method is just 0.13 higher than that of $SALOPT$ and much higher than that of $QEMRD$. Similarly, the $MS-SSIM$ for our horse model is also just 0.0121% higher than that of $SALOPT$. This is because the CD value of the horse model is relatively low, since most of the texture space is occupied by similar colours. As such a texture image is not complicated in terms of either the colour information or the pattern on the image surface, its features are unlikely to be lost significantly, even after simplification. Therefore, our method makes no significant improvement over $SALOPT$ in such a situation. Unlike $SALOPT$, $QEMRD$ does not consider saliency information of a texture image but simply reduces the texture image directly with the bilinear filtering operation. Consequently, texels of a texture image are unlikely to be retained after simplification, making such a method difficult to obtain good results. For the wolf model, we achieve similar experimental results for both $PSNR$ and $MS-SSIM$.

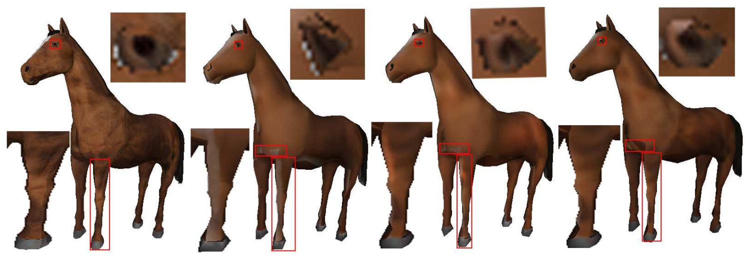
6 Conclusion

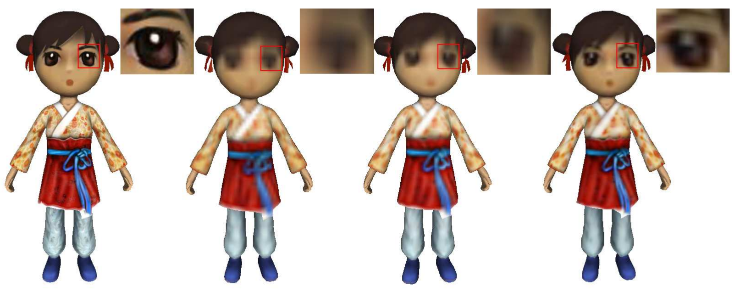
This paper has presented a salient region detection method to support textured model simplification. The method relies on a unified visual saliency map that we have developed to identify important model features, which are jointly formulated by meshes, texture maps and the projection between them, guiding how a model is to be simplified. We have also improved the quality of local saliency identification by a newly developed local filter window. In terms of model simplification, we have introduced a novel relaxation and regularisation algorithm to preserve important visual features from a texture map longer during model simplification. Results show that we can well preserve salient regions even when models are extremely simplified. In the future, we would like to take into account lighting and other surface properties, such as normal map, for determining salient regions of a textured model, since they also play important roles in modelling the appearance of a 3D object in modern graphics applications. We also plan to extend this saliency detection method to identify visually important objects in large-scale virtual environments.

Table 2 $PSNR$ and $MS-SSIM$ of three different methods

		$QEMRD$	$SALOPT$	Our Method	CD
Vase	$PSNR$	29.7443	32.2861	34.1341	101371
	$MS-SIMM$(%)	0.8661	0.8870	0.9325	
Horse	$PSNR$	38.6988	40.9702	41.1091	7894
	$MS-SIMM$(%)	0.9010	0.9221	0.9342	
Girl	$PSNR$	39.0627	41.5688	42.9627	99473
	$MS-SIMM$(%)	0.9295	0.9584	0.9793	
Wolf	$PSNR$	38.1002	39.7663	40.6375	16167
	$MS-SIMM$(%)	0.9142	0.9761	0.9979	


(a) Vase


(b) Horse


(c) Girl

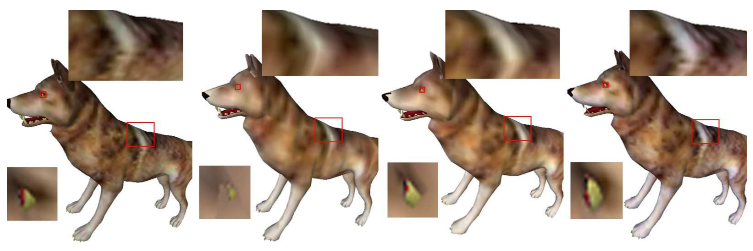

(d) Raptor

Fig. 12 Visual quality comparison among four different reduced textured models. For each model, screenshots obtained from the original textured model, $QEMRD$,$SALOPT$ and our method, are shown respectively.

Acknowledgements This work was partly supported by the National High Technology Research and Development Program of China (863 Program, Grant No. 2013AA013701), Zhejiang Province Natural Science Foundation for Distinguished Young Scientists (Grant No. LR12F02001), and National Natural Science Foundation of China (Grant Nos. 61170214, 61170098), .

References

1. Balmelli, L., Taubin, G., Bernardini, F.: Space-optimized texture maps. Computer Graphics Forum **21**, 411–420 (2002)
2. Bulbul, A., Koca, C., Capin, T., Güdükbay, U.: Saliency for animated meshes with material properties. In: Proc. Applied Perception in Graphics and Visualization, pp. 81–88 (2010)
3. Chen, X., Saparov, A., Pang, B., Funkhouser, T.: Schelling points on 3d surface meshes. ACM Transactions on Graphics (TOG) **31**(4), 29 (2012)
4. Cheng, I., Bischof, W.F.: A perceptual approach to texture scaling based on human computer interaction. In: Short Paper of Euro-

graphics 2006, pp. 1–6 (2006)
5. Connor, C., Egeth, H., Yantis, S.: Visual attention: Bottom-up versus top-down. Current Biology **14**(19), 850–852 (2004)
6. Corsini, M., Larabi, M., Lavoué, G., Petrík, O., Váša, L., Wang, K.: Perceptual metrics for static and dynamic triangle meshes. In: Eurographics 2012-State of the Art Reports, pp. 135–157. The Eurographics Association (2012)
7. Feixas, M., Sbert, M., Gonzalez, F.: Information-theoretic framework for viewpoint selection and mesh saliency. ACM Trans. on Applied Perception **6**(1), 1–23 (2009)
8. Fuchs, H., Golddfeather, J., Hultquist, J.: Fast spheres, shadows, textures, transparencies, and image enhancements in pixel-planes. In: Proc. Siggraph, pp. 109–116 (1985)
9. Gal, R., Cohen-OR, D.: Salient geometric features for partial shape matching and similarity. ACM Transaction on Graphics **25**(1), 130–150 (2006)
10. Garland, M., Heckbert, P.: Surface simplification using quadric error metric. In: Proc. ACM SIGGRAPH, pp. 209–215 (1997)
11. Gonzlez, C., Castell, P., Chover, M., Sbert, M., Feixas, M., Gumbau, J.: Simplification method for textured polygonal meshes based on structural appearance. Signal, Image and Video Processing **7**(3), 479–492 (2013). DOI 10.1007/s11760-013-0450-5
12. Hoppe, H.: Progressive meshes. In: Proc. ACM SIGGRAPH, pp. 99–108 (1996)
13. Hunter, A., Cohen, J.D.: Uniform frequency images: Adding geometry to images to produce space-efficient textures. In: Proc. IEEE Visualization, pp. 243–250 (2000)
14. Iourcha, K., Nayak, K., Zhou, H.: System and method for fixed-rate block-based image compression with inferred pixel values. US patent 5 pp. 956–431 (1999)
15. Itti, L., Koch, C.: Feature combination strategies for saliency-based visual attention systems. Journal of Electronic Imaging **10**(1), 161–169 (2001)
16. Itti, L., Koch, C., Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. IEEE Trans. on Pattern Analysis and Machine Intelligence **20**(11), 1254–1259 (1998)
17. Kadir, T., Brady, M.: Saliency, scale and image description. Internation Journal of Computer Vision **45**, 83–105 (2001)
18. Kim, Y., Varshney, A.: Saliency guided enhancement for volume visualization. IEEE Trans. on Visualziation and Computer Graphics **12**(5), 925–932 (2006)
19. Lavouë, G.: A local roughness measure for 3d meshes and its application to visual masking. ACM Trans. on Applied Perception **5**(4), 1–23 (2009)
20. Lee, C.H., Varshney, A., Jacobs, D.W.: Mesh saliency. In: Proc. ACM SIGGRAPH, pp. 659–666 (2005)
21. Leifman, G., Shtrom, E., Tal, A.: Surface regions of interest for viewpoint selection. In: Proc. IEEE computer vision and pattern recognition (CVPR), pp. 414–421 (2012)
22. Li, F.W.B., Lau, R.W.H., Kilis, D., Li, L.W.F.: Game-on-demand: An online game engine based on geometry streaming. ACM Trans. Multimedia Comput. Commun. Appl. **7**(3), 19:1–19:22 (2011)
23. Lindstrom, P.: Model simplification using image and geometry-based metrics. Ph.D. thesis, Georgia Institute of Technology (2000)
24. Liu, Y.S., Liu, M., D. Kihara, K.R.: Salient critical points for meshes. In: Proc. ACM Solid and Physical Modeling, pp. 277–282 (2007)
25. Longhurst, P., Debattista, K., Chalmers, A.: A gpu based saliency map for high-fidelity selective rendering. In: Proc. Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa, pp. 21–29 (2006)
26. Luebke, D.P., Hallen, B.: Perceptually-driven simplification for interactive rendering. In: Proc. Eurographics Workshop on Rendering Techniques, pp. 223–234. Springer-Verlag (2001)
27. Martinez, J., Andujar, C.: Space-optimized texture atlases for 3d scenes with per-polygon textures. In: Proc. Pacific Graphics, pp. 14–23 (2010)
28. Menze, N., Guthe, M.: Towards perceptual simplification of models with arbitrary materials. Computer Graphics Forum **29**(7), 2261–2270 (2011)
29. Miao, Y., Feng, J.: Perceptual-saliency extremum lines for 3d shape illustration. The Visual Computer **26**(6-8), 433–443 (2010)
30. Miao, Y., Feng, J., R., P.: Visual saliency guided normal enhancement technique for 3d shape depiction. Computers & Graphics **35**(3), 706–712 (2011)
31. Montabone, S., Soto, A.: Human detection using a mobile platform and novel features derived from a visual saliency mechanism. Image and Visual Computing **28**, 391–402 (2009)
32. Moze, M.C., Sitton, M.: Computational Modeling of Spatial Attention. UCL Press (1998)
33. Ponomarenko, N., Battisti, F., Egiazarian, K., Astola, J., Lukin, V.: Metrics performance comparsion for color image database. In: Proc. 4th intnational workshop on video processing and quality metric for consumer electronics, pp. 14–16 (2009)
34. Qu, L.J., Meyer, G.W.: Human detection using a mobile platform and novel features derived from a visual saliency mechanism. Image and Visual Computing **28**(3), 391–402 (2008)
35. Sander, P.V., Snyder, J., S. J. Gortler, e.a.: Texture mapping progressive meshes. In: Proc. ACM SIGGRAPH, pp. 409–416 (2001)
36. Schaefer, S., McPhail, T., Warren, J.: Image deformation using moving least squares. ACM Transaction on Graphics (SIGGRAPH 2006) **25**(3), 533–540 (2006)
37. Song, R., Liu, Y., Martin, R.R., Rosin, P.L.: Mesh saliency via spectral processing. ACM Transactions on Graphics (TOG) **33**(1), 6:1–6:17 (2014)
38. Ström, J., Akenine-Möller, T.: ipackman: High-quality, low-complexity texture compression for mobile phones. In: Proc. the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, HWWS ’05, pp. 63–70 (2005)
39. Tang, Y., Zhang, H.X., Q. Wang, e.a.: Importance-driven texture encoding based on samples. In: Computer Graphics International, pp. 169–176 (2005)
40. Taubin, G.: Estimating the tensor of curvature of a surface from a polyhedral approximation. In: Proc. IEEE International Conference on Computer Vision, pp. 902–907 (1995)
41. Walther, D., Koch, C.: Modeling attention to salient proto-objects. Neural Networks **19**(9), 1395–1407 (2006)
42. Wang, N., Simoncelli, E., Bovik, A.: Multiscale structural similarity for image quality assessment. In: Proc. the 37th IEEE Asilomar Conference on Signals, Systems and computers, pp. 1398–1402 (2003)
43. Yang, B., Wang, X., Li, F.W.B.: Salient region of textured 3d model. In: Proc. Pacific Graphics, pp. 78–84 (2010)
44. Yoon, D., Kim, K.J.: 3d game model and texture generation using interactive genetic algorithm. In: Proceedings of the Workshop at SIGGRAPH Asia, pp. 53–58 (2012)