UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RODRIGO BARNI MUNARETTI

Perceptual Guidance in Mesh Processing and Rendering Using Mesh Saliency

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Prof. Dr. João Luiz Dihl Comba
Advisor

Porto Alegre, May 2007

CIP – CATALOGING-IN-PUBLICATION

Barni Munaretti, Rodrigo

Perceptual Guidance in Mesh Processing and Rendering Using Mesh Saliency / Rodrigo Barni Munaretti. – Porto Alegre: PPGC da UFRGS, 2007.

73 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2007. Advisor: João Luiz Dihl Comba.

1. Mesh saliency. 2. Deformable meshes. 3. Mesh segmentation. 4. View-dependent rendering. I. Comba, João Luiz Dihl. II. Título.

Consistency is the last refuge of the unimaginative.
— OSCAR WYLDE

ACKNOWLEDGEMENTS

We thank Nicolas Clément for the Grunt and Orc datasets, the Stanford Graphics Lab for the Bunny and Armadillo models, Cyberware for the Isis, Goddess1, Goddess2 and Ganesh models and the AIM@SHAPE Shape Repository for the Hand dataset. The Laçador dataset was kindly provided by LsDM - UFRGS (www.ufrgs.br/ndsm). We also thank Amitabh Varshney for the help and support in understanding and expanding upon the concept of Mesh Saliency.

This work is supported by a scholarship by CNPq.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

GPU	Graphics Processing Unit
FBO	Framebuffer Object
BVI	Base Vertex Index Texture
CVI	Contracted Vertex Index Texture
SCI	Shared Cluster Indices Texture
VCO	Vertex Contraction Order Texture
CSO	Cluster Stop Point Texture
FBOP	Framebuffer Object for Contraction Processing
FBOPw	FBOP write surface
FBOPr	FBOP read surface
FBOC	Framebuffer Object for Data Format Conversion
MAXORD	Maximum contraction order for vertex contraction

LIST OF SYMBOLS

$\mathscr{S}_i(v)$	Single-pose mesh saliency value for vertex *i*
$\mathscr{S}_{oi}$	Multi-pose mesh saliency value for vertex *i*
$\mathscr{S}_{bi}$	Base mesh saliency value for vertex *i*
$\mathscr{S}_{i,j}$	Mesh saliency value for vertex *i* in pose *j*
$\mathscr{S}_{c_0}$	Starting saliency value for cluster *c*
$\mathscr{S}_{ci}$	Cluster-normalized saliency value for vertex *i* in cluster *c*
$\mathscr{S}_{mesh}$	Accumulated saliency for the entire mesh
$max(\mathscr{S}_c)$	Maximum saliency value detected during growth phase for cluster *c*
$\mu(\mathscr{S}_c)$	Mean saliency value for cluster *c*
$\mathscr{C}(v)$	Mean curvature value for vertex *v*
$G(\mathscr{C}(v), \sigma)$	Gaussian-weighted average of the mean curvature at neighborhood σ for vertex *v*
$N(v, \sigma)$	Local vertex neighborhood at scale σ
M_b	Base mesh for multi-pose saliency computation
P	Set of all poses or deformations for multi-pose saliency computation
p	Pose or deformation, element of set *P*
$\mathscr{W}$	Single-pose saliency weight map
$\mathscr{W}_o$	Multi-pose saliency weight map
λ_c	User-defined multiplier for seed selection pool
λ_{perc}	User-defined percentage for contraction distribution

λ_n	User-defined normal vector contribution factor
C	User-defined cluster count
$\mu(\overrightarrow{n_c})$	Mean cluster normal for cluster c
$\overline{x_c}$	Spatial centroid for cluster c
r_c	Radius of cluster bounding sphere for cluster c
R_{global}	Global resolution target for vertex contraction
R_{local}	Local resolution target for vertex contraction
A_{est}	Estimated screen-space visibility
p_{cam}	Camera position
I_{est}	Visual Importance estimation
$VC(v_i)$	Contracted vertex index for vertex i

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Considerations on perceptual information are quickly gaining importance in mesh representation, analysis and display research. User studies, eye tracking and other techniques are able to provide ever more useful insights for many user-centric systems, which form the bulk of computer graphics applications. In this work we build upon the concept of Mesh Saliency — an automatic measure of visual importance for triangle meshes based on models of low-level human visual attention — improving, extending and integrating it with different applications.

We extend the concept of Mesh Saliency to encompass deformable objects, showing how a vertex-level saliency map can be constructed that accurately captures the regions of high perceptual importance over a range of mesh poses or deformations. We define *multi-pose saliency* as a multi-scale aggregate of curvature values over a locally stable vertex neighborhood together with deformations over multiple poses. We replace the use of the Euclidean distance by geodesic distance thereby providing superior estimates of the local neighborhood. Results show that multi-pose saliency generates more visually appealing mesh simplifications when compared to a single-pose mesh saliency.

We also apply Mesh Saliency to the problem of mesh segmentation and view-dependent rendering, introducing a technique for segmentation that partitions an object into a set of face clusters, each encompassing a group of locally interesting features. Mesh Saliency is incorporated in a propagative mesh clustering framework, guiding cluster seed selection and triangle propagation costs and leading to a convergence of face clusters around perceptually important features. We compare our technique with different fully automatic segmentation algorithms, showing that it provides similar or better segmentation without the need for user input. Since the proposed clustering algorithm is specially suitable for multi-resolution rendering, we illustrate application of our clustering results through a saliency-guided view-dependent rendering system, achieving significant framerate increases with little loss of visual detail.

Keywords: Mesh saliency, deformable meshes, mesh segmentation, view-dependent rendering.

Direcionamento Perceptual em Processamento de Malhas Utilizando Saliência

RESUMO

Considerações de informação perceptual têm ganhado espaço rapidamente em pesquisas referentes a representação, análise e exibição de malhas. Estudos com usuários, *eye tracking* e outras técnicas são capazes de fornecer informações cada vez mais úteis para sistemas voltados a usuário, que formam a maioria das aplicações em computação gráfica. Neste trabalho nós expandimos sobre o conceito de Saliência de Malhas — uma medida automática de importância visual para malhas de triângulos baseada em modelos de atenção humana em baixo nível — melhorando, extendendo e realizando integração com diferentes aplicações.

Nós extendemos o conceito de Saliência de Malhas para englobar objetos deformáveis, mostrando como um mapa de saliência em nível de vértice pode ser construído capturando corretamente regiões de alta importância perceptual através de um conjunto de poses ou deformações. Nós definimos *saliência multi-pose* como um agregado multi-escala de valores de curvatura sobre uma vizinhança localmente estável, em conjunto com deformações desta vizinhança em múltiplas poses. Nós substituímos distância Euclideana por geodésica, assim fornecendo melhores estimativas de vizinhança local. Resultados mostram que saliência multi-pose gera resultados visualmente mais interessantes em simplificações quando comparado à saliência em uma única pose.

Nós também aplicamos saliência de malhas ao problema de segmentação e *rendering* dependente de ponto de vista, introduzindo uma técnica para segmentação que particiona um objeto em um conjunto de *clusters*, cada um englobando um grupo de características localmente interessantes. Saliência de malhas é incorporada em um *framework* para clustering propagativo, guiando seleção de pontos de partida para clusters e custos de propagação de faces, levando a uma convergência de clusters ao redor de características perceptualmente importantes. Nós comparamos nossa técnica com diferentes métodos automáticos para segmentação, mostrando que ela fornece segmentação melhor ou comparável sem necessidade de intervenção do usuário. Uma vez que o algoritmo de segmentação proposto é especialmente aplicável a rendering multi-resolução, nós ilustramos uma aplicação do mesmo através de um sistema de rendering baseado em ponto de vista guiado por saliência, alcançando melhorias consideráveis em *framerate* com muito pouca perda de qualidade visual.

Palavras-chave: saliência de malhas, malhas deformáveis, segmentação de malhas, rendering dependente de ponto de vista.

1 INTRODUCTION

Perceptual considerations are becoming increasingly important in mesh processing, analysis and display techniques (O'SULLIVAN, C. et al., 2004). User-centric systems form the bulk of computer graphics applications, and displaying results that are visually appealing while maintaining interactivity for datasets that become larger and more detailed is proving to be a difficult task. Extensive research has been — and is being — conducted in order to tackle such a problem, but the need to process and render ever more complex meshes keeps outpacing the capacity of state of the art systems.

Many techniques have been developed to help alleviate this issue; among them, mesh simplification and segmentation proved to be particularly useful in reducing the overall size of datasets to be processed or displayed (LUEBKE, 2001). Even methods such as these have their limitations, tough: greatly simplified meshes may lose detail to a point where features of interest may be lost, and segmentation may group faces into clusters that do not represent interesting regions of an object, especially when perceptual factors must be considered. If regions of high interest in a particular dataset could be reliably identified, these — and other — processes could be greatly improved.

Mesh saliency (LEE; VARSHNEY; JACOBS, 2005) provides exactly such information. Using multi-scale curvature analysis based on a center-surround operator, it reliably separates what most observers consider to be interesting regions from the surrounding context, in an entirely automated process. In this work, we improve and extend the concept of mesh saliency, integrating into it a better method for neighborhood estimation, generalizing the operator for posable or deformable objects, and integrating it into a mesh segmentation framework. Applications in mesh simplification and view-dependent rendering are used to illustrate these contributions. Figure 1.1 gives an overview of our work and contributions.

1.1 Multi-Pose Mesh Saliency

Deformable or articulated meshes are ubiquitous among computer graphics applications such as simulations, games and others. As the need for additional detail has increased, dealing with massive deformable meshes has become a challenge — not only for interactive rendering, but also for other operations such as mesh filtering, mesh simplification, mesh segmentation, shape matching, and lighting design. Incorporating a measure of perceptual importance in mesh processing greatly enhances the relevance and usability of such processing in a wide variety of user-centric visual applications. Mesh Saliency, in-

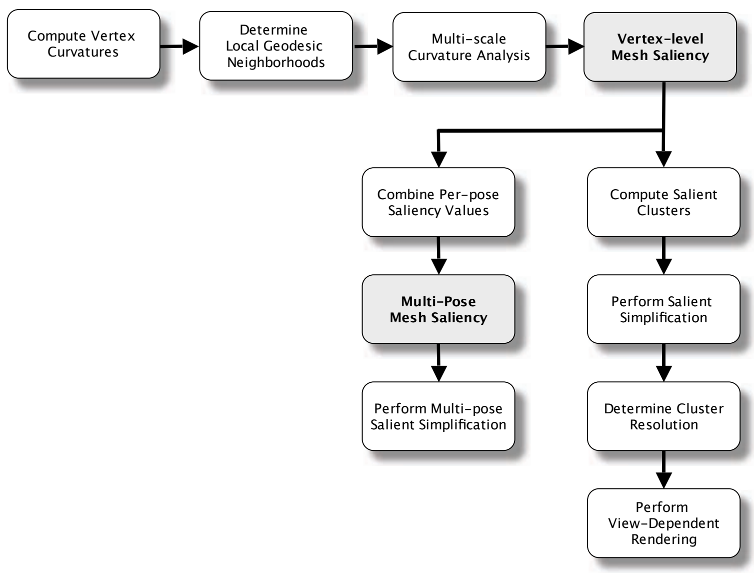

Compute Vertex Curvatures
Determine Local Geodesic Neighborhoods
Multi-scale Curvature Analysis
Vertex-level Mesh Saliency
Combine Per-pose Saliency Values
Compute Salient Clusters
Multi-Pose Mesh Saliency
Perform Salient Simplification
Perform Multi-pose Salient Simplification
Determine Cluster Resolution
Perform View-Dependent Rendering

Figure 1.1: Solution overview: Multi-scale curvature analysis is used to calculate mesh saliency. Analysis of saliency values for multiple poses results in multi-pose mesh saliency, which is combined with QSlim to generate mesh simplifications suitable for all possible poses. Mesh saliency can also guide mesh segmentation and real-time importance estimation, which are in turn used in a view-dependent rendering system.

troduced in (LEE; VARSHNEY; JACOBS, 2005) as a computational model of multi-scale perceptual importance based on a center-surround mechanism, provides a mathematical framework to compute such a perceptual measure on static meshes. We extend the concept of static mesh saliency to cover a broader class of deformable meshes with multiple poses. Our technique takes as input the original mesh and a set of key deformations or poses and outputs a scalar multi-pose saliency map. We show the effectiveness of our new multi-pose saliency measure for the example application of mesh simplification. Figure 1.2 illustrates multi-pose saliency on a simple dataset.

1.2 Saliency-Guided Mesh Segmentation

Mesh segmentation forms the cornerstone of many object manipulation, parameterization and analysis techniques (JI, J. et al., 2006; JULIUS; KRAEVOY; SHEFFER, 2005; NEHAB; BARCZAK; SANDER, 2006; QU; MEYER, 2006; LAI et al., 2006). Most mesh segmentation algorithms work directly on raw, scale-independent, geometric information, such as local curvature and face normals (SANDER, P. V. et al., 2003); even those that attempt to detect prominent features do so without taking into account their perceptual importance. Available techniques that employ visual importance estimation usually require user input to determine areas of high interest to guide the the segmentation process(KHO;

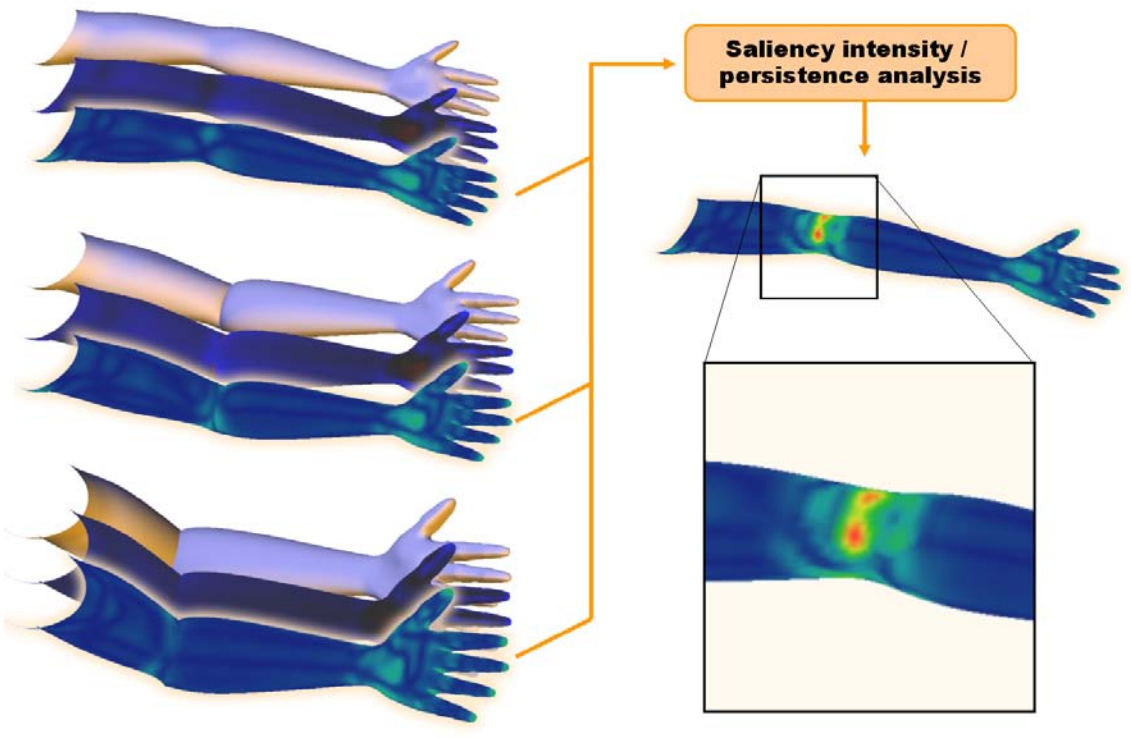


Figure 1.2: Multi-pose mesh saliency takes as input a base mesh and a set of key poses or deformations, computes saliency values for each mesh configuration, and computes a measure of perceptual importance over the entire range of poses. On the left we show the base mesh of the arm and two key poses with their curvature and saliency maps; on the right we show the multi-pose mesh saliency. The detail-box shows the close-up around the elbow.

GARLAND, 2003; CARR, N. A. et al., 2006).

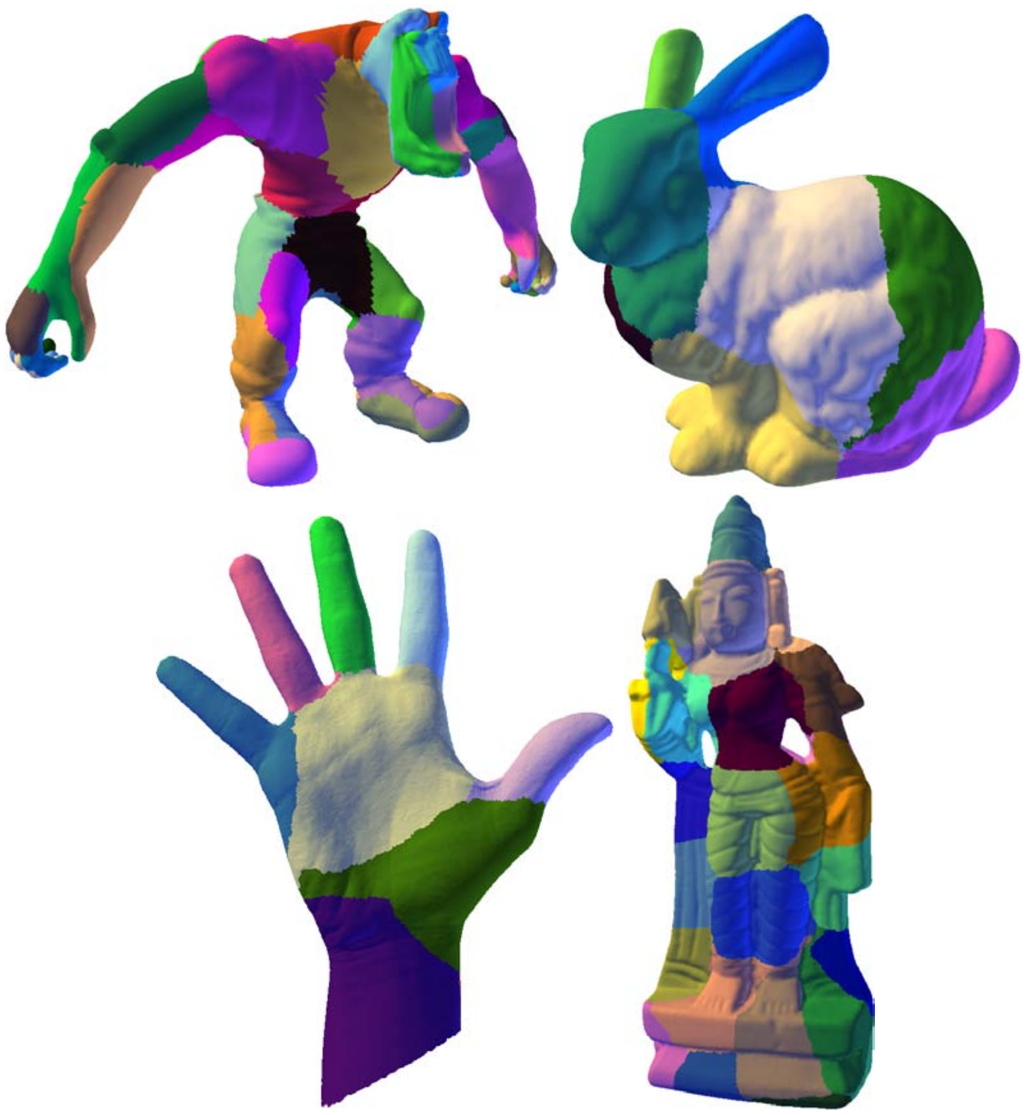


Figure 1.3: Mesh saliency is used to perform salient clustering which is suitable for view-dependent rendering. Segmentation results for the Grunt, Bunny, Hand and Goddess1 datasets.

We integrate mesh saliency (LEE; VARSHNEY; JACOBS, 2005) with mesh segmentation in order to partition a triangle mesh into a set of face clusters representing what most will agree are perceptually interesting features. Our technique builds upon the propagative segmentation algorithm introduced in (SANDER, P. V. et al., 2003), incorporating mesh saliency values for the cluster seed selection and propagation cost computation steps.

As recent surveys (ATTENE, M. et al., 2006) indicate, evaluating the quality of different mesh segmentation algorithms can be a difficult task. We compare our approach against k-means clustering and propagative segmentation, which are fully automatic and considered to be representative techniques. We also validate our results through improved view-dependent rendering of multi-resolution meshes: being driven by a perceptual importance metric, our segmentation technique generates results that are particularly suitable for this application. A multi-resolution representation of the mesh is computed in pre-processing time, guided by cluster-normalized mesh saliency values; later, in render time, cluster importance is estimated through analysis of cluster-level saliency and viewing parameters. We use this perceptual importance estimate to select each cluster's level of detail, while maintaining a given global mesh resolution target. The final mesh for rendering is then constructed in real time through a GPU-based vertex contraction algorithm.

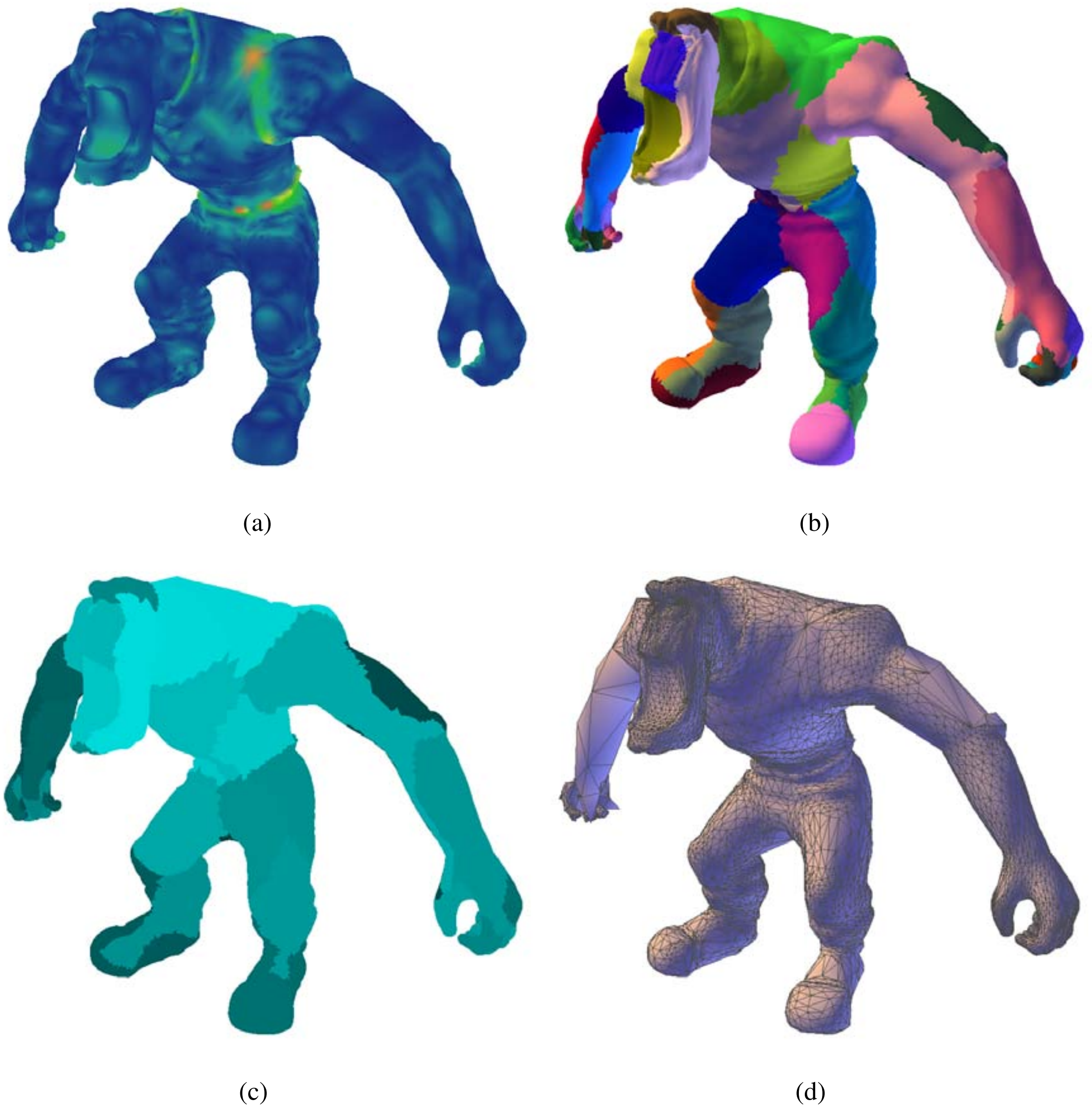


Figure 1.4: Saliency-guided view-dependent rendering steps: saliency computation (a) is used to perform salient clustering (b). View dependent and saliency information are used to estimate cluster importance (c), which is passed to a fragment program that calculates the right resolution for each cluster inside the GPU. Final rendering of the multi-resolution mesh is shown in (d).

1.3 Contributions

Our main contributions in this work can be summarized as follows:

- **Improved single-pose mesh saliency:** We use the geodesic distance, instead of Euclidean, to determine the local neighborhood for computing the single-pose saliency.

This yields superior estimates of visual importance, due to reduced error in neighborhood estimation.

- **Multi-pose mesh saliency:** We introduce the concept of multi-pose saliency, computing a single saliency map over several deformations of a mesh to provide an aggregate measure of importance. Vertices that remain important over multiple poses are assigned higher overall saliency than others. Multi-pose mesh saliency can be integrated into processing of deformable meshes.
- **Multi-pose simplification:** We illustrate the use of multi-pose mesh saliency in deformable mesh processing through the example application of simplification using edge collapses. Our technique is able to better preserve regions of high importance over the range of deformations when compared with simplifications based on just the quadric-error metric or a combination of quadric-error metric with single-pose saliency.
- **Salient clustering**: We use saliency as a measure of perceptual importance to determine cluster starting points and propagation, so that we can join triangles into meaningful groups, each containing a set of locally interesting features. Segmentation results obtained through this technique are as good as or better than comparable techniques, and are particularly suitable for perceptually-guided processes, such as view-dependent rendering.
- **Multi-resolution salient simplification**: We perform mesh simplification over the entire mesh through saliency-guided QSlim (GARLAND; HECKBERT, 1997), weighted by the cluster-localized saliency map to preserve interesting cluster-level features. A multi-resolution representation of the object is created, where regions of high local — e.g. cluster level neighborhood — importance are preserved.
- **Saliency-guided view-dependent rendering**: We analyze each cluster independently based on factors such as saliency and expected screen-space area. Resolution is preserved for clusters with high estimated saliency and visibility. A vertex contraction algorithm running on the GPU is used to compute updated vertex index lists in real time, thus greatly improving framerates while maintaining high visual fidelity.

Figure 1.3 shows an example of our saliency-based segmentation algorithm, while Figure 1.4 showcases the steps of our proposed view-dependent rendering system and their respective results. The rest of this document is organized as follows. Chapter 2 reviews related work in the field. Chapter 3 reviews saliency concepts and shows our improved single-pose saliency computation method, while Chapter 4 discusses the multi-pose mesh saliency algorithm and describes application of multi-pose saliency to mesh simplification. The saliency-guided mesh segmentation technique and multi-resolution rendering system are described in Chapters 5 and 6. Results are discussed in Chapter 7, followed by conclusions and directions for future work.

2 RELATED WORK

A large body of literature exists concerning the individual issues we tackle in this work. Our review shows that while perceptual considerations are rapidly gaining importance in many areas of computer graphics research, little work has been done regarding integration of measures of perceptual importance into mesh analysis and processing applications — even more so for fully automatic measures. In this chapter we review related works regarding multi-resolution meshes, deformable meshes and mesh segmentation under the light of perceptually-guided processing.

2.1 Perceptual Considerations in Computer Graphics

Sullivan *et al.*'s report on perceptually adaptive graphics (O'SULLIVAN, C. et al., 2004) discusses how perceptual information is used to evaluate and improve many visual applications, such as mesh optimization, animation, collision detection, non-photorealistic rendering and global illumination, as well as saliency estimation through gaze-directed user experiments. They provide an extensive survey of recent work regarding perceptual considerations on several fields of computer graphics research, showing that solutions where human perception is taken into account are able to provide greatly improved visual results.

Howlett *et al.* have conducted eye-tracking experiments to identify and predict feature saliency for three-dimensional objects (HOWLETT; HAMILL; O'SULLIVAN, 2005), and show that higher visual fidelity can be attained on low level-of-detail optimizations when using saliency as guiding element. Saliency information is extracted through human gaze data obtained over a set of eye-tracking experiments, and then employed to guide mesh simplification. They observe that while using saliency to guide simplification of natural objects generally provides simplifications with better visual fidelity, for man-made objects the opposite is true - concluding that saliency prediction for artificial objects is more difficult.

The work of (GAL; COHEN-OR, 2006) describes a method for partial matching of triangular meshes, through identification of unique salient features over a set of low level surface descriptors. Such descriptors are independent from the object's underlying triangulation, and can match similar objects or regions with different representations. They define salient features as clusters of local descriptors that form a non-trivial region of the surface. Similar regions are then found through indexing of rotation-invariant features and a voting scheme. Excellent results in shape matching can be obtained through this method, as seen on Figure 2.1; focusing on shape matching, though, this work does not assign perceptual importance to different regions of an object.

The concept of mesh saliency as an automatic perceptual metric was first introduced on (LEE; VARSHNEY; JACOBS, 2005), as a measure of regional importance based on multi-scale curvature estimations. This technique is based on computational models of human visual perception, building upon methods already well established in the field of image processing, and is able to reliably identify in triangular meshes regions that most observers would find of high visual importance. Their paper also presents techniques for viewpoint selection and saliency-guided optimization. The latter, which serves as starting point for our work, greatly improves preservation of perceptually interesting features during optimization, without the need for any kind of user input.

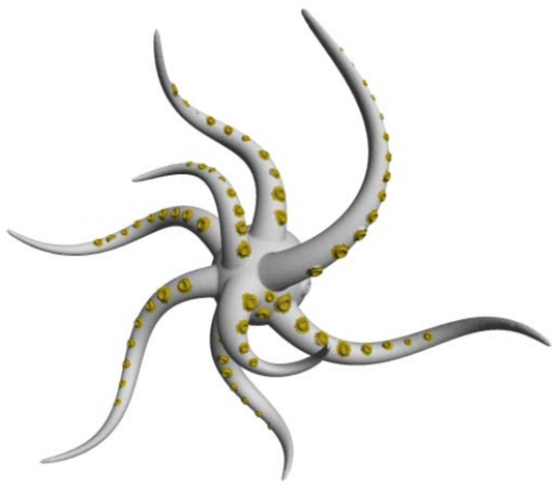


Figure 2.1: Partial shape matching through salient feature detection (GAL; COHEN-OR, 2006).

2.2 Multi-Resolution Meshes

Progressive meshes were introduced in (HOPPE, 1996) as an efficient representation of triangular meshes for optimization, transmission and level-of-detail approximation, among other applications. Their representation is based on edge collapses — where edges are progressively collapsed into a single vertex during simplification and expanded during reconstruction — allows rapid retrieval of arbitrary levels of detail from optimized triangular meshes, a key factor for continuous level-of-detail techniques.

(GARLAND; HECKBERT, 1997) introduced QSlim, a method which uses quadric error metrics (Figure 2.2) for fast and high-quality optimization of triangle meshes through iterative contraction of vertex pairs. Any arbitrary vertex pair within a user-defined distance limit can be considered for contraction, instead of only pairs belonging to the same edge. This method supports non-manifold surfaces, allows error bounding, and is directly applicable to the creation of progressive meshes.

The work of (KHO; GARLAND, 2003) introduces a user-guided simplification method, allowing preservation of key perceptual features on very low level-of-detail optimizations through user selection of interesting areas. They extend the QSlim simplifica-

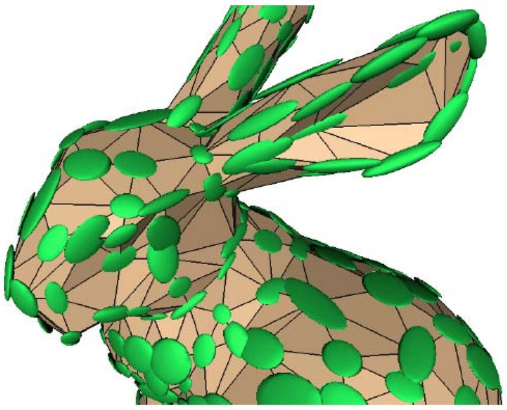


Figure 2.2: Visualization of the error quadrics employed by the QSlim method to guide vertex pair contraction. Bunny dataset simplified to 1.4% of original resolution (GAL; COHEN-OR, 2006).

tion technique, modifying the quadric error values associated with each vertex through user-provided importance estimation and introducing additional constraint quadrics, effectively guiding the order of vertex pair contractions. While good results can be obtained when proper importance values are provided, as seen on Figure 2.3, no automatic measure of perceptual importance is applied.

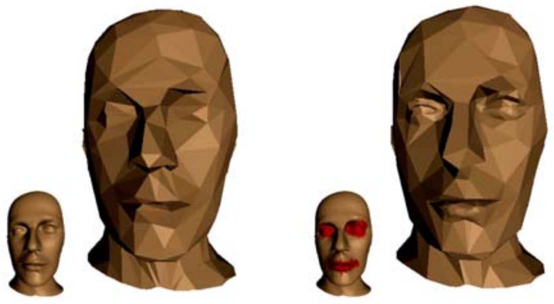


Figure 2.3: User-guided simplification. On the left, fully automatic QSlim. On the right, guided simplification. Results from (KHO; GARLAND, 2003).

2.3 Simplification of Deformable Meshes

Deformable meshes are common across a significant range of computer graphics applications, and a large body of literature exists on the subject of multi-pose analysis or pose-independent operations for deformable meshes. While work exists which takes into account perceptual issues, such as (LEE; VARSHNEY; JACOBS, 2005), few tackle the issue of perceptual importance estimation for deformable or posable meshes.

In (SCHMALSTIEG; FUHRMANN, 1999), Schmalstieg and Fuhrmann introduce a technique to simplify deformable objects that requires the user to specify areas that may undergo deformation as part of ASLOD (Animated Smooth Level of Detail), a comprehensive method for multi-resolution representation of scenes. Areas specified by the user as deformable are then preserved to a greater extent than non-deformable ones. Only the mesh representation on the original pose is maintained — when displaying simplified visualizations of the deformable areas, the simplified skin is modified through linear interpolation, following the underlying skeleton.

Poulin and Houle, on (HOULE; POULIN, 2001), propose treating the base mesh of an articulated model as a static mesh, simplifying it maintaining a progressive mesh representation, and then applying the vertex split operations on the deformed vertex to construct the proper level of detail for any specific pose. A morphing technique is introduced to adapt the vertex split operations to the deformed bone space. No perceptual information is taken into account — in fact, the proposed technique is independent of simplification method, any algorithm that constructs a progressive mesh representation can be employed.

In (MOHR; GLEICHER, 2003), Mohr and Gleicher introduce a deformation-sensitive decimation technique, analyzing vertex contraction error on several input poses when determining contraction costs for deformable meshes. Based on QSlim, their method takes as input a representative sample of poses or deformations, and considers the contraction cost of any vertex pair to be the sum of the contraction costs for that pair over the entire range of representative poses. Simplification proceeds as described in (GARLAND; HECKBERT, 1997). Here, also, perceptual metrics are not considered.

DeCoro and Rusinkiewicz present a similar technique (DECORO; RUSINKIEWICZ, 2005), but specifically for skeletally articulated objects: only the base mesh, skeleton and skinning information are required as input. They also extend the QSlim method, combining error quadrics from the range of possible poses — now obtained through Monte Carlo integration over a pose probability function that dictates the skeleton constraints — in order to minimize the average error of the deforming mesh. All quadrics are transformed to a common coordinate system prior to consideration, avoiding possible error magnification problems. Meshes that undergo other types of deformation, such as morphing or free-form deformations, are not supported.

2.4 Mesh Segmentation

Mesh segmentation — or triangle clustering — techniques have been applied to the solution of many different mesh-related problems, from render-time optimization to control skeleton extraction.

Multi-chart geometry images, introduced in (SANDER, P. V. et al., 2003), employ clustering algorithms to break a triangle mesh into a set of manageable subdivisions used to build a 2D mesh atlas representing the whole object, which can then be used to reconstruct the original mesh. A propagative clustering framework is used — clusters are constructed through a Dijkstra-like search on the dual graph of the mesh — where spatial proximity and normal are used to determine face propagation costs. This provides good results for mesh planarization, generating clusters that are mostly flat and with compact boundaries.

The technique presented in (NEHAB; BARCZAK; SANDER, 2006) divides a mesh into a set of planar clusters, sorted in a view-independent manner that greatly reduces overdraw, therefore reducing rendering time. Here, also, a propagative clustering technique is used; since their main concern during segmentation is cluster planarity, the only metric considered in cost computation during cluster propagation is based on the cluster average normal and the candidate face normal.

Though methodologies vary — from adaptations of vector quantization algorithms (LINDE; BUZO; GRAY, 1980) to Dijkstra-like propagation tuned for best rectangular fit (CARR, N. A. et al., 2006) (Figure 2.4) — most clustering algorithms are guided either by simple geometric information, such as distance and normal deviation (SANDER, P. V. et al., 2003), ignoring perceptual characteristics, or require user input to define interesting areas(CARR, N. A. et al., 2006).

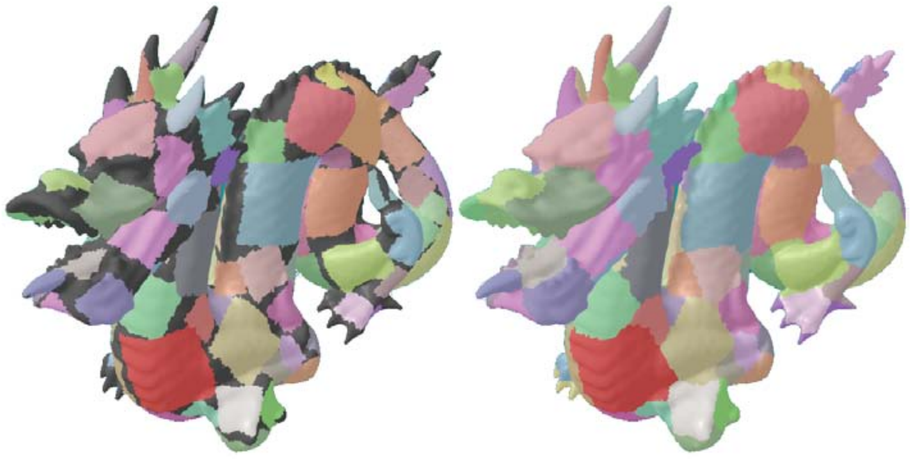


Figure 2.4: User-guided propagative mesh segmentation. Propagation is performed in order to provide best rectangular fit. Results from (CARR, N. A. et al., 2006).

Exceptions exist, though: (KATZ; TAL, 2003) introduces a hierarchical segmentation technique based on fuzzy clustering and spectral cuts, divided in two stages which are applied recursively. The first stage determines the probability of a face belonging to a given cluster, while the second stage computes the actual boundary between clusters. Each iteration treats a binary case, dividing the mesh into two clusters; recursive application can generate the desired number of subdivisions. A set of stopping conditions is used to automatically guide the recursive process. It segments a mesh into meaningful pieces, resulting, generally, in cuts at regions of deep concavities; jagged boundaries are avoided. Effectiveness of the technique is demonstrated through a control skeleton extraction application. Only single-scale geometric information is employed, though; no perceptual information is considered.

A technique for mesh decomposition based on expanding spheres around each vertex is presented in (MORTARA et al., 2003). A spherical bubble is expanded from each mesh vertex, and the intersection of the bubble with the surrounding mesh is studied, taking into account not only the curvature of the surrounding surface but its topology as well. The surrounding surface is then characterized according to a set of criteria, based on the intersection curves, at each vertex, and this information is then used to segment the mesh. While shape analysis is performed at multiple scales, this is not translated into a measure of perceptual importance.

The segmentation technique introduced in (ZHANG; LIU, 2005) uses visually-salient spectral cuts, but saliency is defined only for entire mesh regions and not on the vertex level. Subdivisions are always binary, and performed recursively; cuts are performed at negative minima of surface principal curvatures — shape context is analysed and salient parts are identified in 1D embedding space for cut placement. Intuitively meaningful results can be obtained through this method, but no relative perceptual importance is assigned to regions.

(KATZ; LEIFMAN; TAL, 2005) introduces a method for segmentation through feature point and core extraction, which is invariant regarding object pose and component proportion. In this technique, the input mesh undergoes a coarsening process beforehand in order to reduce computational time and decrease sensitivity to noise. A canonical, pose-invariant representation of the mesh is then constructed, where Euclidean distance between two points is similar to the geodesic distance between the same points in any pose. Feature points are detected, followed by core component extraction — other segments are then obtained by subtracting the core component from the rest of the mesh. Feature points are defined as those resting on the tips of prominent components of a model, and not through perceptual importance estimation.

In (LAI et al., 2006) a feature-sensitive segmentation technique based on k-means clustering is introduced that divides a mesh into meaningful pieces. Feature-sensitive remeshing is employed to build a hierarchy of meshes, then used in hierarchical segmentation. Geodesic distance, curvature and texture information are used during face cost computation for cluster propagation. While segmentation results are visually interesting, no perceptual information is employed or extracted during the process.

3 SINGLE-POSE MESH SALIENCY

Determining regional importance over a mesh surface is a critical step in many applications such as simplification and segmentation. Until recently, most techniques employed purely geometric measures — such as local curvature — or required user input to indicate important areas. Geometric measures such as curvature maxima or minima do not always correlate with perceptual importance. For example, as seen on Figure 3.1, regions with dense patterns of repeated high-curvature bumps may be fairly ordinary from a perceptual point of view, even though single-scale curvature analysis would indicate high importance. Relying on user input reduces automation and offers additional problems, i.e. when processing a large number of different meshes. Mesh saliency builds upon computational models of human attention to automatically extract a multi-scale measure of vertex-level importance.

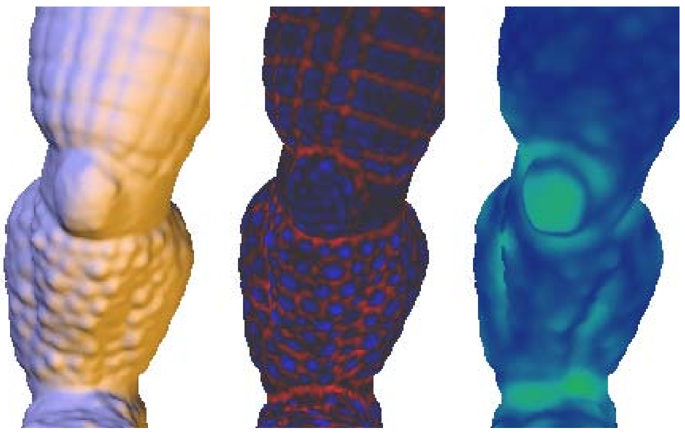


Figure 3.1: Importance estimation comparison: original mesh (a), local curvature (b) and mesh saliency (c).

3.1 Visual Attention Models

Understanding how humans interpret complex scenes is a key factor in perceptually-guided applications. Recent models of human visual attention (ITTI; KOCH, 2001) point to a two-component framework, where scenes are analysed for interesting objects in both a bottom-up, image-based manner and in a top-down, task-oriented manner. Most computational approaches to model visual attention focus on the first component, since it is

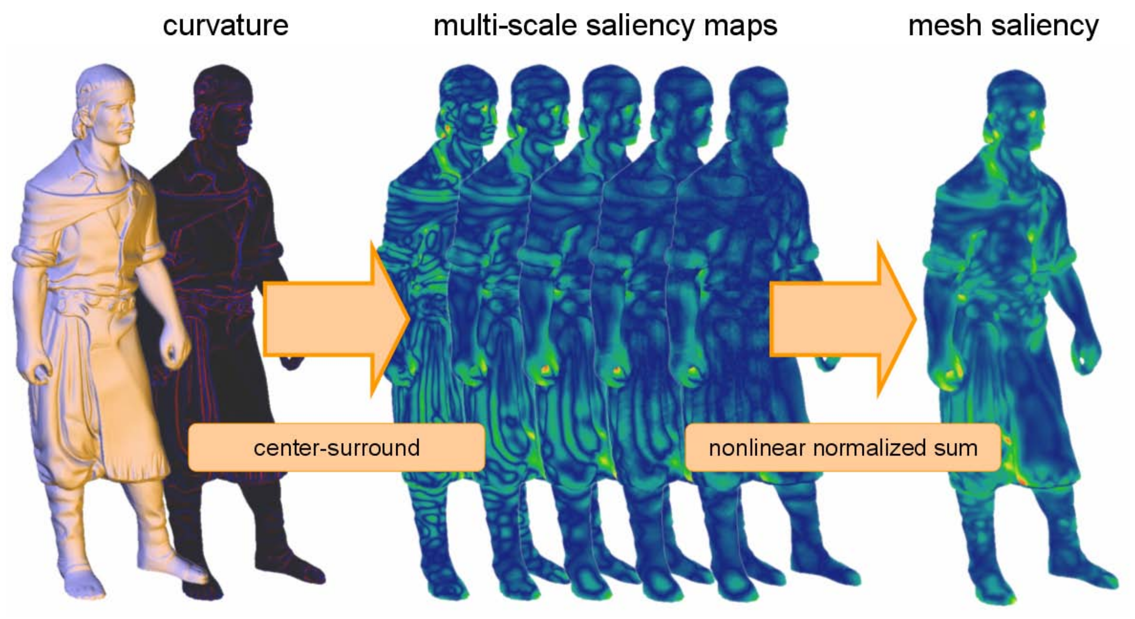


Figure 3.2: Saliency computation uses a center-surround mechanism to combine curvature information obtained at different scales.

able to indicate regions of high interest independently of the task being performed, and is thus applicable to a wide range of problems.

Such models rely heavily on saliency — a measure of visual conspicuousness — to quickly select areas likely to be of high importance. Koch and Ullman were the first to propose this (KOCH; ULLMAN, 1985); they introduced the notion of a "‘saliency map"’ that combined all available visual information and indicated areas of relative importance. The visual focus then simply scanned the saliency map in order of decreasing saliency to identify interesting objects. Studies indicate that what matters most in the early stages of bottom-up attention is feature contrast, and not feature strength. Due to this observation, most models agree that center-surround analysis is essential in separating salient regions from their surrounding context (ITTI; KOCH; NIEBUR, 1998), even though different strategies for building a saliency map may exist for different input types — i.e., when analysing an image several factors must be accounted for, such as color, intensity, orientation and contrast. Additionally, important features may exist in varying relative sizes, leading to multi-scale considerations (ITTI; KOCH; NIEBUR, 1998; LEE; VARSHNEY; JACOBS, 2005).

Several works modeled saliency as a hybrid process combining bottom-up and top-down analysis (WOLFE, 1994; TSOTSOS, J. K. et al., 1995), where high-level guidance is necessary in the earlier stages of attention; Itti et. al, in (ITTI; KOCH; NIEBUR, 1998), proposed a purely bottom-up strategy, where saliency is modeled directly after surround modulation effects using gaussian filters at multiple scales. This latter proved to be extremely effective at many applications, including traffic sign detection, recognition of pedestrians in natural scenes and detection of military vehicles, serving as basis for several other works, including mesh saliency (LEE; VARSHNEY; JACOBS, 2005).

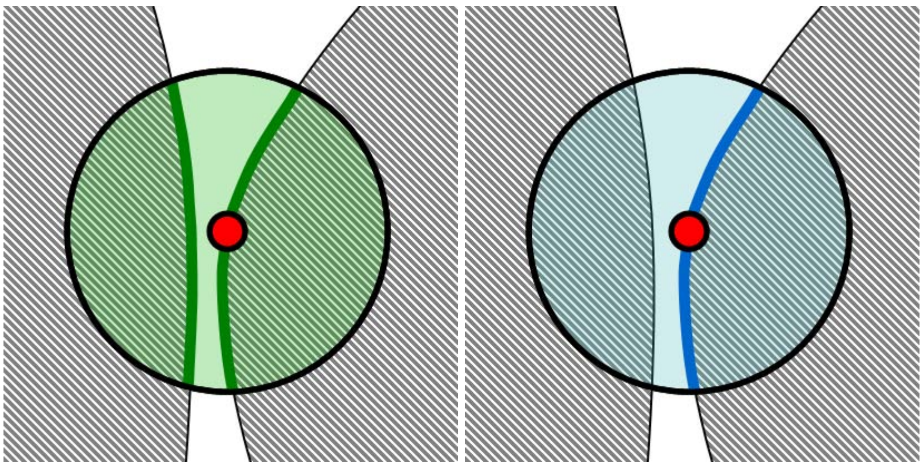


Figure 3.3: Example of a situation where Euclidean distance may yield wrong saliency results. (a) shows, in green, the local neighborhood as determined by the Euclidean distance, encompassing areas from a different section of the mesh. (b) shows, in blue, neighborhood calculated through geodesic distance, correctly capturing the vertex neighbors over the mesh surface. The source vertex is shown in red in both cases.

3.2 Saliency Overview

Inspired by the concept of saliency for two-dimensional images presented in (ITTI; KOCH; NIEBUR, 1998), mesh saliency was introduced in (LEE; VARSHNEY; JACOBS, 2005) as a vertex-level measure of multi-scale regional importance. This method effectively separates important regions from the surrounding context, providing a reliable automatic measure of perceptual importance for triangle meshes.

Mesh saliency applies the center-surround operation presented in (ITTI; KOCH; NIEBUR, 1998) to triangle meshes, substituting local mesh curvature for image color. Vertex-level mesh saliency $\mathscr{S}_i(v)$ for a given vertex v is given by:

$$\mathscr{S}_i(v) \;=\; |G(\mathscr{C}(v),\sigma_i) - G(\mathscr{C}(v),2\sigma_i)| \tag{3.1}$$

where $G(\mathscr{C}(v),\sigma)$ corresponds to a Gaussian-weighted average of the mean curvature $\mathscr{C}(v)$, and σ_i corresponds to the standard deviation of the Gaussian filter at scale i. For all results presented in this work, we have used five scales $\sigma_i \in \{2\varepsilon,\, 3\varepsilon,\, 4\varepsilon,\, 5\varepsilon,\, 6\varepsilon\}$, where ε is defined as 0.3% of the length of the diagonal of the object's bounding box. Note that σ_i only determines the size of the neighborhood being considered — for a given saliency computation, all measurements are extracted from the same mesh.

For mesh saliency as introduced in (LEE; VARSHNEY; JACOBS, 2005), $G(\mathscr{C}(v),\sigma)$ is calculated over a local neighborhood $N(v,\sigma)$ defined by Euclidean distance; $N(v,\sigma) = \{x \,||\, ||x-v|| < \sigma,\ x \text{ is a mesh point}\}$. This is a good measure for most cases, but may generate false results, as exemplified by Figure 3.3. While this might not be critical for static meshes, this is specially important for deformable or articulated meshes. To avoid such issues, we have replaced the Euclidean distance by the geodesic distance for neighborhood determination and Gaussian computation, which gives us an accurate and reliable two-dimensional neighborhood over the surface of the triangle mesh.

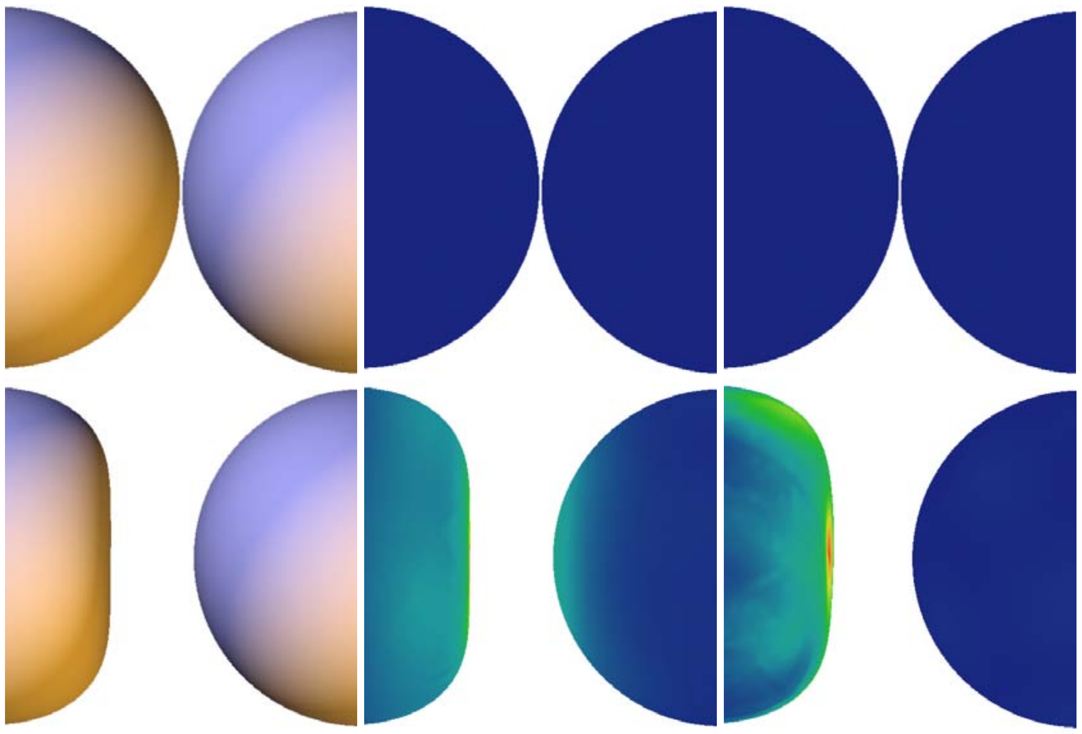


Figure 3.4: Comparison of saliency computation using different distance metrics. The leftmost column shows the model used, formed by two spheres; in the original configuration (top row) and after suffering deformation (bottom row). The middle column shows saliency results using the Euclidean distance; notice how saliency on the rightmost sphere is affected by deformation on its neighbor. The rightmost column shows results for geodesic distance: only the deformed section suffers variation in saliency values.

3.3 Geodesic Saliency

We calculate the geodesic distance using the method presented in (SURAZHSKY, V. et al., 2005), which is itself a modification of the "single source, all destination" Mitchell-Mount-Papadimitriou (MMP) algorithm. The distance function is parameterized over mesh edges; each edge is partitioned into a set of intervals, or windows, representing a group of shortest geodesic paths to the source vertex v. Windows are propagated from the source vertex in a Dijkstra-like manner; the queue is ordered by geodesic distance to the source.

Propagation stops when distance exceeds σ, the limit of the desired local neighborhood. Computation is performed per vertex; neighborhoods from the smallest to the largest scale are determined in order, and distance values from the previous scale are maintained. Effectively, a single geodesic distance computation step is executed per vertex, with propagation stopping at the largest value of σ required for saliency determination.

Figure 3.4 compares results of saliency computation using Euclidean distance to saliency computation using geodesic distance, for two distinct configurations of a deformable mesh.

4 MULTI-POSE MESH SALIENCY

Although mesh saliency provides a reliable measure of perceptual importance for static meshes, it may generate false or incomplete results for objects that undergo deformations or are skeletally animated. Local mesh curvature is an essential component of mesh saliency computation and it can vary wildly on deformable objects, drastically changing importance estimation from one pose to another. Our objective is to compute a single mesh saliency map, indicating potential perceptual importance for any possible pose.

We must, then, evaluate saliency across the entire range of deformations and extract a single importance value per vertex. This process requires, as input, the base mesh M_b plus the set of all key poses (or deformations) P. Each pose $p \in P$ must preserve vertex correspondence, as well as topology information, in relation to the base mesh. The process for calculating multi-pose saliency is as follows.

Per-pose saliency computation is first performed for every pose. Mesh saliency maps — using geodesic distance for neighborhood determination, as detailed on Section 3 — are computed for the base mesh and all input poses. This gives us the range of possible saliency values for each mesh vertex, across the range of possible poses or deformations.

Multi-pose mesh saliency can then be obtained by combining all the per-pose saliency values. Defining the *base saliency* $\mathscr{S}_{bi}$ for a vertex v_i as the lowest saliency value obtained during per-pose saliency computation, the *multi-pose mesh saliency* value $\mathscr{S}_{oi}$ for vertex v_i is given by:

$$\mathscr{S}_{oi} = \mathscr{S}_{bi} + \sum_{j=0}^{|P|} (\mathscr{S}_{i,j} - \mathscr{S}_{bi}) \tag{4.1}$$

where $\mathscr{S}_{i,j}$ is the mesh saliency value for vertex v_i in pose p_j. Intuitively, we are defining *multi-pose saliency* as a multi-scale aggregate of curvature values over a locally stable vertex neighborhood together with the deformations of that neighborhood over multiple poses. The sum of saliency difference across all poses, in relation to the base saliency value, captures saliency persistence and variation for each vertex. Figures 4.1 and 4.2 exemplify this process for the Grunt, Cylinder, Sphere and Arm datasets. On the Arm dataset, high importance is assigned to the inner and outer sections of the elbow, even though both regions have relatively low saliency in the base mesh; this is due to the high curvature variation experienced during deformation, which significantly changes local perceptual importance. The sides of the elbow largely maintain the original saliency value since little variation is encountered. Similar results can be observed on other datasets.

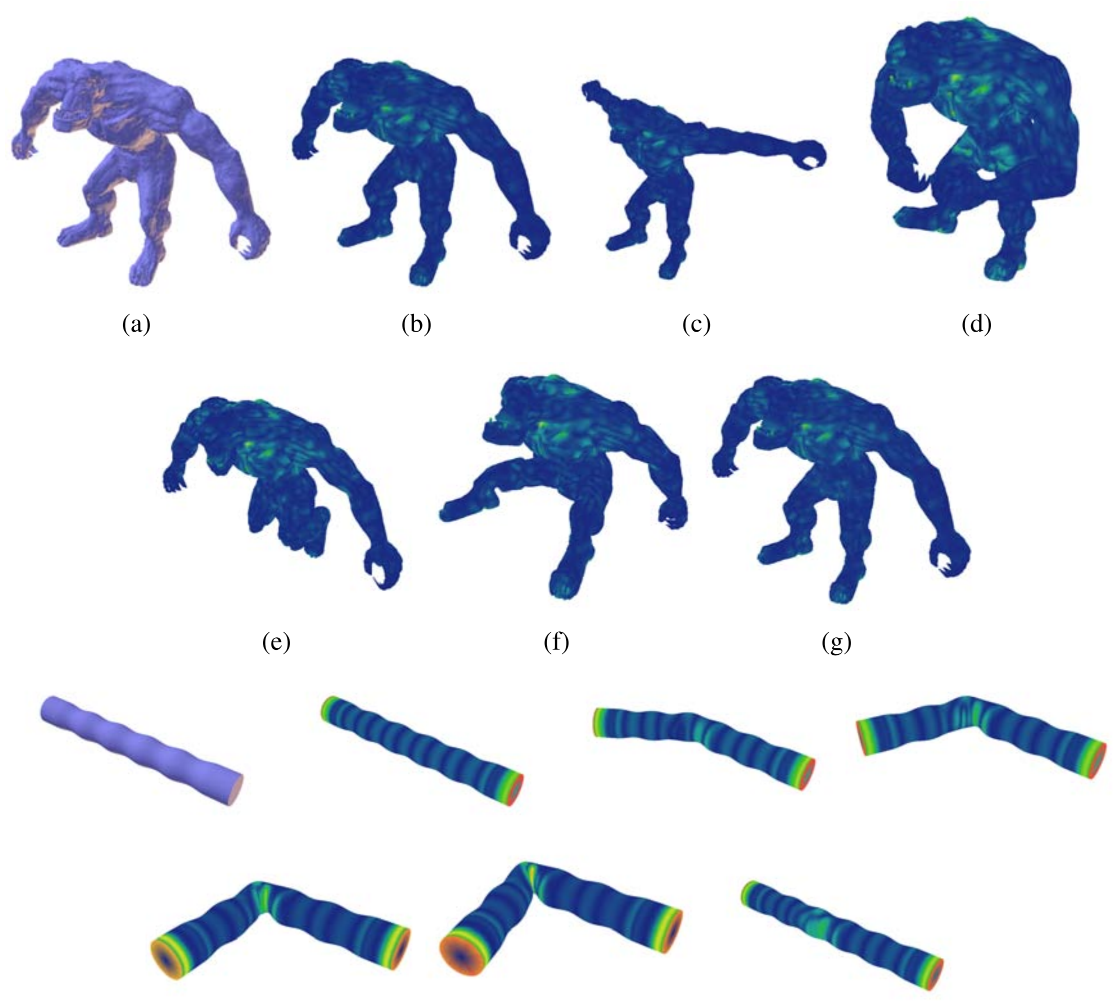


Figure 4.1: Mesh saliency comparison. From left to right: (a) base mesh, (b) single-pose saliency values for the base mesh,(c-f) four additional poses, (g) resulting multi-pose mesh saliency. Results for the Grunt and Cylinder datasets.

4.1 Multi-pose Salient Simplification

Mesh simplification methods are plentiful and varied (LUEBKE et al., 2002). Most take into account purely geometric information — such as local curvature and measured error — to guide the simplification process (HECKBERT; GARLAND, 1999). Several methods take into account possible mesh deformations, be it in the form of key poses or skeletal articulation (MOHR; GLEICHER, 2003; DECORO; RUSINKIEWICZ, 2005). Very few have considered perceptual information (LUEBKE; HALLEN, 2001; WATSON; WALKER; HODGES, 2004; REDDY., 2001). To demonstrate the effectiveness of our multi-pose mesh saliency metric, we have extended the QSlim method (GARLAND; HECKBERT, 1997), by weighting the error quadrics used by the algorithm by the multi-pose mesh saliency. We follow the salient simplification algorithm presented in (LEE; VARSHNEY; JACOBS, 2005), substituting the mesh saliency map for our multi-pose mesh saliency map.

QSlim works through successive vertex pair contractions, ordered by increasing quadric error. Considering P to be the set of planes associated with triangles adjacent to a vertex v.

Each plane $p \in P$ defined by the equation $ax+by+cz+d=0$, represented by $(a,b,c,d)^T$ and with associated quadric $Q_p = pp^T$ has an associated error — defined by the squared distance of v to p — given by $v^T Q_p v$. The quadric Q associated with v is given by the sum of all quadrics of the neighboring planes, $Q = \sum_{p \in P} Q_p$. For a potential contraction pair (v_i, v_j), an optimal contraction target $\bar{v}$ that minimizes error is calculated; contraction error for $\bar{v}$ is given by $\bar{v}^T(Q_i+Q_j)\bar{v}$, where Q_i and Q_j are quadrics for v_i and v_j, respectively. Pair contractions are then performed iteratively, starting from the pair with lowest contraction error. The quadric for a new point $\bar{v}$ is obtained simply by adding both of its parents' quadrics, $Q_i + Q_j$.

Following (LEE; VARSHNEY; JACOBS, 2005), we guide simplification using a weight map $\mathscr{W}_o$, based on the multi-pose mesh saliency $\mathscr{S}_o$. High saliency vertices are amplified, thus ensuring better preservation of visually interesting detail. The amplification operator A introduced in (LEE; VARSHNEY; JACOBS, 2005) is used, based on a threshold α and amplifying parameter λ; saliency values greater than α are amplified by a factor of λ. $\mathscr{W}_o$ is then computed as follows:

$$\mathscr{W}_o(v) = A(\mathscr{S}_o(v), \alpha, \lambda) = \begin{cases} \lambda \mathscr{S}_o(v) & if \mathscr{S}_o(v) \geq \alpha \\ \mathscr{S}_o(v) & if \mathscr{S}_o(v) < \alpha \end{cases} \tag{4.2}$$

Vertex quadrics computed in the initialization phase of the QSlim algorithm are multiplied by their respective weight map value; for a given vertex v, with associated quadric Q, $Q \leftarrow \mathscr{W}_o(v)Q$. Analogously to (LEE; VARSHNEY; JACOBS, 2005), weight map value for a contracted vertex $\bar{v}$, obtained by the contraction of a vertex pair (v_i,v_j), is given by $\mathscr{W}_o(v_i) + \mathscr{W}_o(v_j)$. For all our experiments, λ was set to 100, and α to 70% of the maximum measured multi-pose saliency value. All vertices were defined by four-component homogeneous coordinates.

This ensures that interesting features — for all relevant poses or deformations — are preserved longer. Figure 7.2 compares the results of multi-pose salient simplification to single-pose salient simplification and the standard QSlim method.

We note that it is not our goal to preserve deformation weights for articulated meshes or similar parameters for other deformable objects. Several works, such as (DECORO; RUSINKIEWICZ, 2005) and (HOULE; POULIN, 2001), have proposed different techniques to solve this problem, and their approaches could be adapted to our solution if so desired.

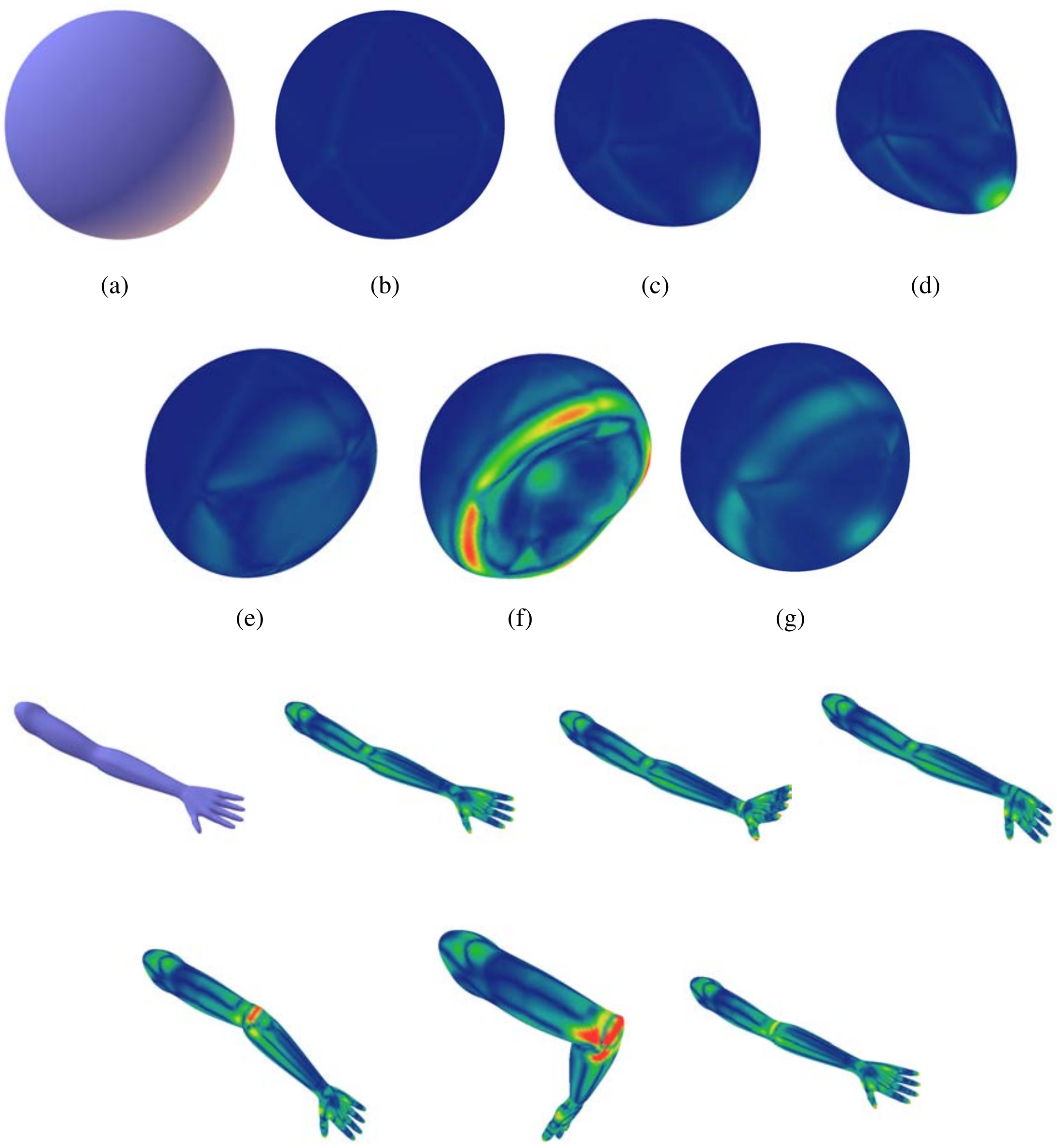


Figure 4.2: Mesh saliency comparison. From left to right: (a) base mesh, (b) single-pose saliency values for the base mesh,(c-f) four additional poses, (g) resulting multi-pose mesh saliency. Results for the Sphere and Arm datasets.

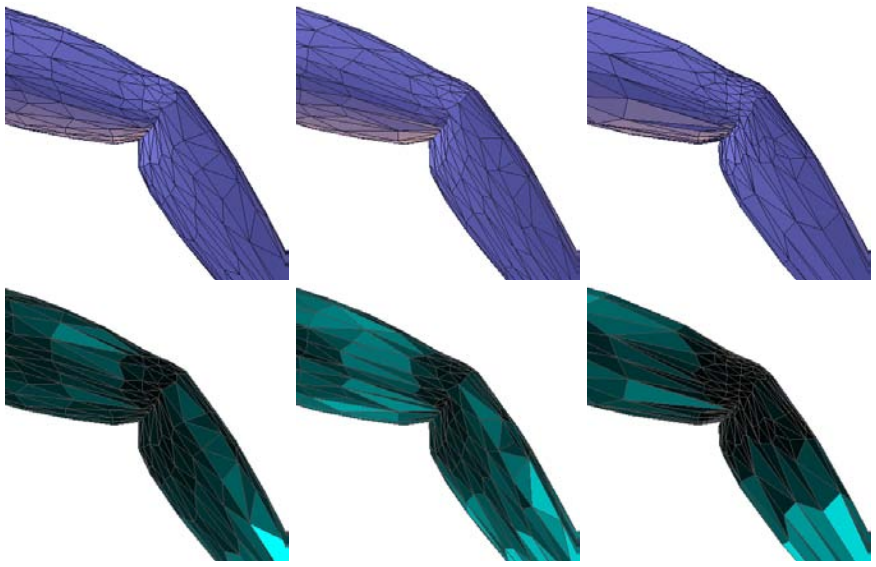


Figure 4.3: Comparison of simplification results, after deformation. Top row: simplification results using standard QSlim, QSlim plus single-pose saliency from the base mesh, and QSlim using multi-pose saliency. Bottom row: relative face sizes for each simplification; lighter faces are larger.

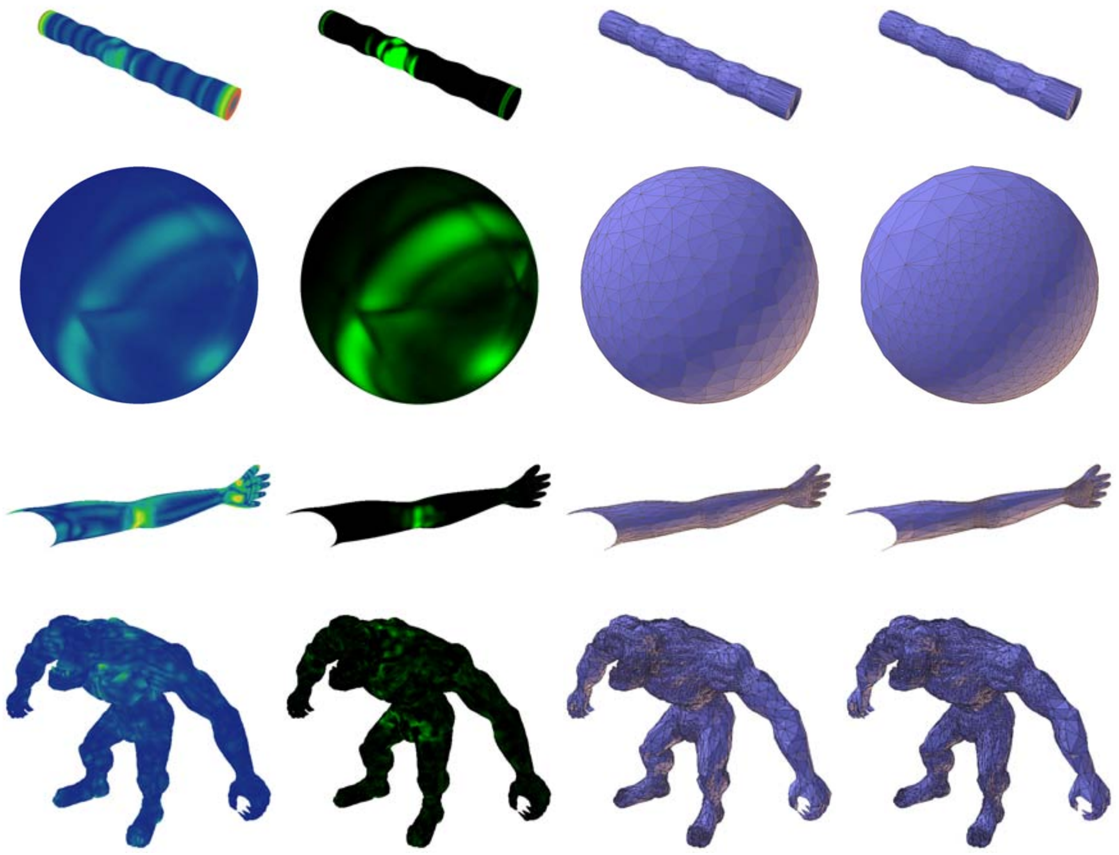


Figure 4.4: Comparison of simplification results using QSlim weighted by mesh saliency values. From left to right: the first column shows multi-pose saliency values, the second column shows the difference between multi-pose mesh saliency and single-pose saliency as computed for the base mesh, the third column shows simplification results using QSlim weighted by single-pose saliency and the fourth column shows simplification results using QSlim weighted by multi-pose mesh saliency. From top to bottom, results for the Cylinder, Sphere, Arm and Grunt datasets, respectively.

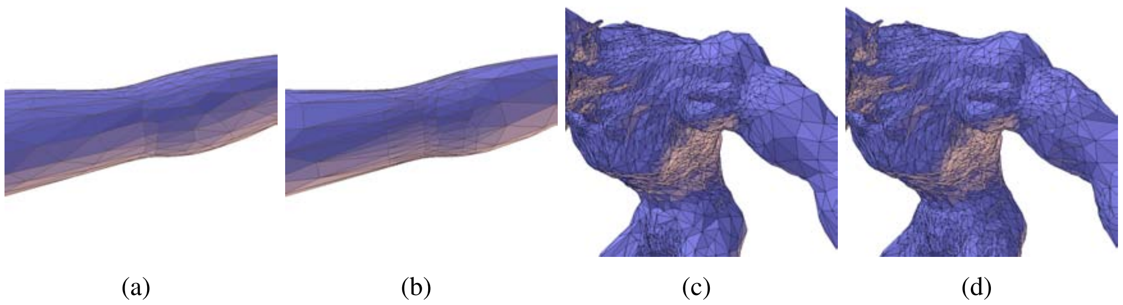


Figure 4.5: Detail of simplification results. For the Arm dataset, using (a) single-pose saliency and (b) multi-pose saliency. For the Grunt dataset, using (c) single-pose saliency and (d) multi-pose saliency. Notice the higher concentration of detail on regions where visually interesting features are likely to undergo deformation.

5 SALIENT CLUSTERING

Mesh segmentation consists of partitioning a triangle mesh into a set of face clusters, each comprised of a subset of the original object's triangles. Faces assigned to each cluster share a few common properties, such as spatial proximity and orientation. We define salient clustering as the process of segmenting a triangle mesh into a group of clusters, where each cluster encompasses a local set of interesting features as defined by mesh saliency. This section describes a segmentation approach that reliably detects and groups triangles belonging to interesting features by integrating vertex-level mesh saliency information into a propagative clustering framework. Note that both static and deformable meshes can benefit from this method, simply by substituting standard, single-pose mesh saliency for multi-pose mesh saliency when applying it to posable or deformable objects.

5.1 Cluster Determination

Our clustering technique is a variation of the chartification algorithm based on Lloyd-Max Quantization presented in (SANDER, P. V. et al., 2003), modified to consider mesh saliency information during cluster computation. As in (SANDER, P. V. et al., 2003), it is an iterative process, alternatingly propagating and re-centering face clusters starting from an initial set of seed elements. Another key difference of our method is that it starts from an already pre-selected set of seed elements, instead of gradually building the cluster list as iterations are performed. Our method is composed of a seed initialization phase, as well as iterative cluster growth and re-seeding phases, which are executed until convergence is reached.

Seed initialization is done by sampling from the regions of highest importance in the mesh. A vertex pool is created, containing the $\lambda_c * C$ highest-saliency vertices, where λ_c is a user-defined multiplier and C is the desired cluster count; C seeds are then randomly chosen from this pool. Each cluster is initialized with saliency, normal and position values. All faces that share a seed vertex are placed on the cluster propagation queue. For all our experiments λ_c was set to 5.

Cluster growth works through Dijkstra-like searches, performed simultaneously for all clusters. The dual graph of the mesh, where pairs of faces are connected by edges, is used as propagation medium. Each search starts on the highest-saliency face already in the queue; subsequent faces are evaluated through a cost function that considers normal deviation, distance and saliency variation. This promotes fairly planar clusters over areas of similar saliency, thus tending to encompass single prominent salient features or sets of similar and closely-packed features. More accurately, the edge cost between a face f on a cluster c and a candidate face f' adjacent to it is defined by:

$$\begin{aligned} cost(f,f') = &(\lambda_n - (\mu(\overrightarrow{n_c}) \cdot \overrightarrow{n_{f'}})) * \\ &(|\mu(\mathscr{S}_c) - \mathscr{S}_{f'}| * |\mathscr{S}_{c_0} - \mathscr{S}_{f'}| * |\mathscr{S}_f - \mathscr{S}_{f'}|) * \\ &(|\overline{x_{f'}} - \overline{x_f}|) \end{aligned} \tag{5.1}$$

where $\mu(\overrightarrow{n_c})$ is the mean cluster normal, $\overrightarrow{n_{f'}}$ is the normal vector for f', $\overline{x_f}$, $\overline{x_{f'}}$, $\mathscr{S}_f$ and $\mathscr{S}_{f'}$ are, respectively, the centroids and saliency values for f and f', $\mu(\mathscr{S}_c)$ is the cluster mean saliency and $\mathscr{S}_{c_0}$ is the cluster starting saliency value. Relative weight for normal contribution is adjusted by λ_n. Note that the scale of each component is not important — since each component multiplies the others, only relative magnitude needs to be considered. Scale is assumed to be constant for any given component in any given mesh. Growth proceeds until all mesh faces are assigned to a cluster. For all our experiments λ_n was set to 1.

Cluster re-seeding is performed next. Each cluster has its seed and initial data updated; the new seed should be the most interior high-saliency vertex possible. New seeds are also selected through Dijkstra search, now starting from all faces located at the cluster border and advancing inwards. Edge cost is now defined by a combination of distance and saliency variation:

$$cost(f,f') = (\overline{x_{f'}} - \overline{x_f}) * (\mathscr{S}_{f'}/max(\mathscr{S}_c)) \tag{5.2}$$

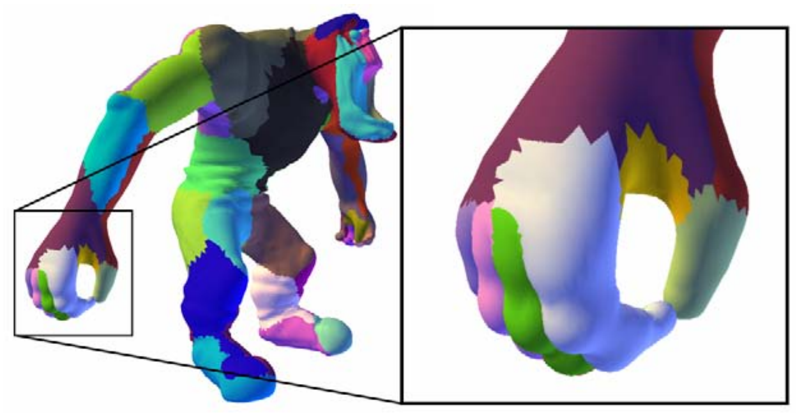
where *max*($\mathscr{S}_c$) is the cluster maximum saliency value detected during the growth phase. The new seed will be the highest-saliency vertex belonging to the last face reached during propagation.

Convergence happens through successive growth and re-seeding phases: both are repeated in order until the new seeds are identical to those encountered on the previous iteration. As in (SANDER, P. V. et al., 2003), we check for the existence of possible cycles in the convergence process; in such a case, any point on the cycle is acceptable.

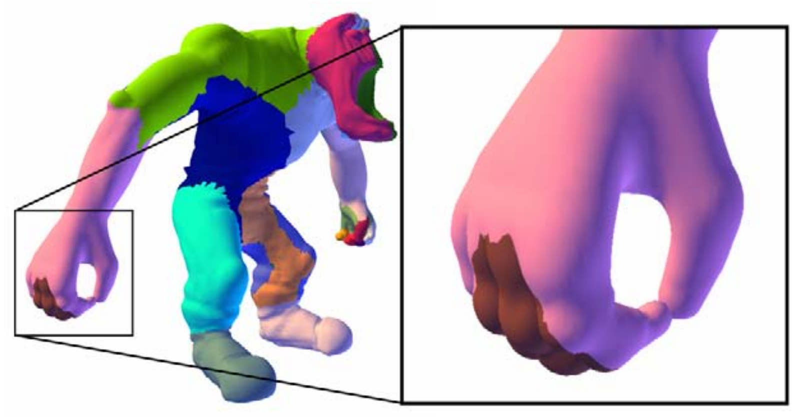
Figure 5.1 compares examples of triangle meshes partitioned using salient clustering and non-salient propagative clustering. Notice how in Figure 5.1.a each finger is effectively isolated in a single cluster, while features are much less distinct in 5.1.b. This comparison is extended in Figure 7.7 to include the clustering algorithm using the K-means algorithm based on the LBG algorithm (LINDE; BUZO; GRAY, 1980).

5.2 Cluster-level Properties

Once all clusters have been determined, each one is analyzed and a set of cluster level properties is determined (Table 5.1),including saliency-based information. Since our clustering method generates segmentations suitable for multi-resolution rendering, we further exploit saliency information at render-time for estimating the screen-space area and perceptual importance of each cluster.



(a)



(b)

Figure 5.1: Comparison of clustering results: Salient clustering (a) and non-salient propagative clustering (b).

5.3 Multi-resolution Salient Simplification

Again, as in Section 4.1, we extend the salient simplification method presented in (LEE; VARSHNEY; JACOBS, 2005) to generate multi-resolution representations of a clustered triangular mesh. This can be seen as an extension of the quadrics-based simplification method (QSlim) where mesh saliency is incorporated as a quadric weighting factor.

Our extension uses a cluster-localized saliency map to guide simplification contractions, better preserving detail that is significant inside a specific cluster. This map is computed by normalizing saliency values inside each cluster, on a linear scale that ranges from zero (at the cluster base saliency) to one (at the cluster peak saliency), and then adding the normalized saliency values to the cluster mean saliency:

$$\mathscr{S}_{ci} = ((\mathscr{S}_i - min(\mathscr{S}_c))/(max(\mathscr{S}_c) - min(\mathscr{S}_c))) + \mu(\mathscr{S}_c) \tag{5.3}$$

This localized map is smoothed and amplified as in (LEE; VARSHNEY; JACOBS, 2005), and its values are then used to weight the quadrics computed by the QSlim method. Optimization is performed over the entire mesh, so connectivity at cluster boundaries can be maintained. Figure 5.2 compares original saliency values to smoothed and amplified saliency and to cluster-normalized saliency.

Table 5.1: Saliency Properties per cluster.

Property	Description
Total saliency	Sum of the saliency contributions from each vertex belonging to the cluster
Mean saliency	Average of the saliency contributions from each vertex belonging to the cluster
Peak saliency	Maximum saliency value encountered among elements of the cluster
Base saliency	Minimum saliency value encountered among elements of the cluster
Total area	Sum of the area of all cluster faces
Mean normal	Average of all cluster face normals
Peak normal	Maximum deviation from the mean normal

We observe that globally important details are better preserved when the original saliency is used, while regions with high local saliency are preserved when we use cluster-localized saliency values. Since our goal is to adaptively preserve detail in a view-dependent manner during render-time, the second option is preferred: using global saliency can lead to vast regions of the model suffering from heavy optimization while detail is carefully preserved on relatively small areas. This would jeopardize our ability to select an adequate level-of-detail in render-time; i.e. when regions that originally had high saliency values are selected for extensive simplification.

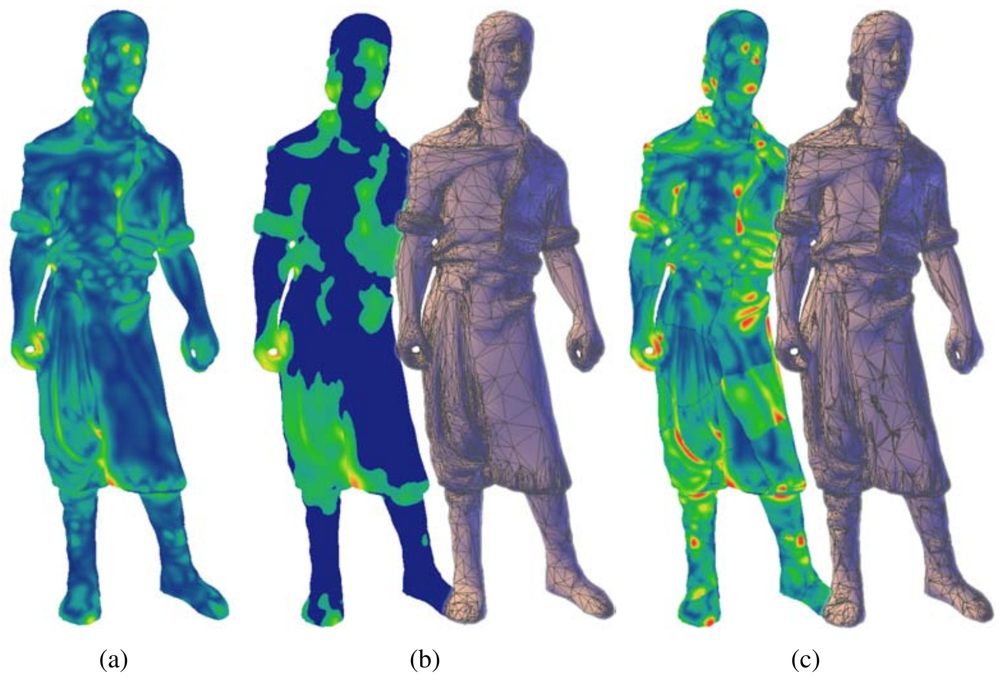


Figure 5.2: Saliency Comparison: original saliency (a), smoothed and amplified saliency and resulting simplified mesh (b) and cluster-localized saliency and resulting simplified mesh (c).

6 VIEW-DEPENDENT RENDERING

Multi-resolution rendering involves selecting the proper level-of-detail for each rendered object based on application requirements, which typically emphasize shorter rendering times while maintaining the best possible visual result. We propose a saliency-guided system, taking advantage of results generated by saliency-guided mesh segmentation to effectively estimate visual importance in real time, feeding a GPU-based vertex contraction system. Figure 6.1 illustrates the interconnection between system components.

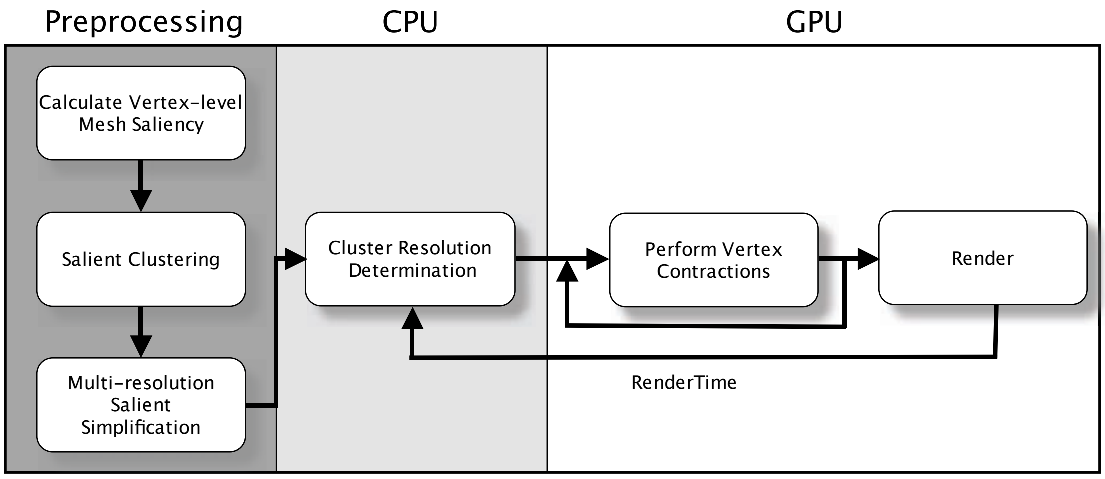

Preprocessing
CPU
GPU
Calculate Vertex-level Mesh Saliency
Salient Clustering
Multi-resolution Salient Simplification
Cluster Resolution Determination
Perform Vertex Contractions
Render
RenderTime

Figure 6.1: System Overview: In a preprocessing stage a salient-based clustering algorithm generates mesh clusters that are simplified based on saliency information. CLOD selection through vertex contractions is performed on the GPU using view-dependent information, generating an array of vertex indices that define the multi-resolution mesh to be rendered.

Our technique works by estimating cluster screen-space visibility and perceptual importance by comparing its position and orientation relative to the camera against a set of previously calculated attributes (detailed in section 5.2). This data is then used to determine the cluster rendering resolution, which serves as input to a GPU-based level-of-detail selector that builds the proper vertex index arrays to be used for rendering. While resolutions are selected independently for each cluster and vertex indices reference localized cluster vertex arrays, a global resolution-matching step is executed to guarantee hole-free cluster boundaries, removing the need for additional zippering and minimizing face overlap.

Salient clustering is an essential component of our solution because it fulfills the following requirements:

1. **Identify sets of triangles that comprise distinct salient features of the original object:** Different features can have different levels of visual importance to an observer. Using mesh saliency, a measure of global importance can be extracted for each cluster, and each can be displayed at a resolution appropriate to the estimated importance. Having similar and spatially-close features within a single cluster allows us to quickly select the level of detail at run-time without the need for extensive computation.

2. **Reduce processing and bandwidth requirements for level-of-detail selection during render-time:** Processing a vertex list for multi-resolution rendering is a time-consuming task. Performing it on the CPU is relatively slow, and occupies a processor that would otherwise be employed for other tasks. A GPU solution can be much faster, but data-transfer bandwidth issues must be carefully considered; dividing an object into clusters can reduce the number of bits required to store a vertex index, thus reducing the required bandwidth.

We evaluated different clustering techniques, and while most of them were able to generate results that satisfied the second requirement, all failed on properly detecting and grouping perceptually interesting features without user input, excepting saliency-guided mesh segmentation.

6.1 Cluster Resolution Determination

Once a global resolution target (R_{global}) for an object to be rendered is chosen, we must determine how each cluster will be optimized to generate the best possible visual results. Two factors are taken into account when level-of-detail for a cluster is being calculated: estimated screen-space visibility and estimated visual importance. Both can be obtained when the cluster position and orientation relative to the camera are compared to the set of cluster attributes obtained in the pre-processing stage.

Estimated screen-space visibility (A_{est}) is basically a factor of the cluster distance from the camera, total area, mean normal and peak normal deviation. We assume, conservatively, that when the vector pointing from the cluster centroid to the camera is perfectly aligned with the cluster mean normal, the entire cluster is visible. As the angle between these vectors increases, visible area decreases proportionally; when it is greater than the cluster peak normal deviation visibility can be expected to be very low. More specifically, we define the visualization angle factor as:

$$\lambda_{angle} = \angle(\overrightarrow{v_{cam}}, \mu(\overrightarrow{n_c})) - peak(\overrightarrow{n_c}) \tag{6.1}$$

where $\overrightarrow{v_{cam}}$ is the vector from the cluster centroid to the camera, $\mu(\overrightarrow{n_c})$ is the cluster mean normal, and $peak(\overrightarrow{n_c})$ is the cluster peak normal deviation (maximum angle between the normal of a face belonging to the cluster and the cluster's mean normal). While this is not a precise measure of visibility — to have such a measure visibility for every face in the cluster would have to be evaluated — it is a suitable approximation for our purposes. Additionally, distance from the camera must be taken into account: the farther away from the camera an object is, the smaller its associated area will be. Distance factor is defined as:

$$\lambda_{dist} = |\overline{x_c} - p_{cam}| - r_c \tag{6.2}$$

where $\overline{x_c}$ is the cluster centroid, p_{cam} is the camera position, and r_c is the radius of the cluster bounding sphere (Figure 6.2). Multiplying the visualization angle by the inverse of the distance factor and by the cluster area gives us the combined screen-space area estimate:

$$A_{est} = \lambda_{angle} * 1/\lambda_{dist} * A_{cluster} \tag{6.3}$$

Note that the case where the distance factor (λ_{dist}) is zero must be considered. Treatment is application-specific; for our tests, when a value of zero was found, the last non-zero value previously encountered was applied.

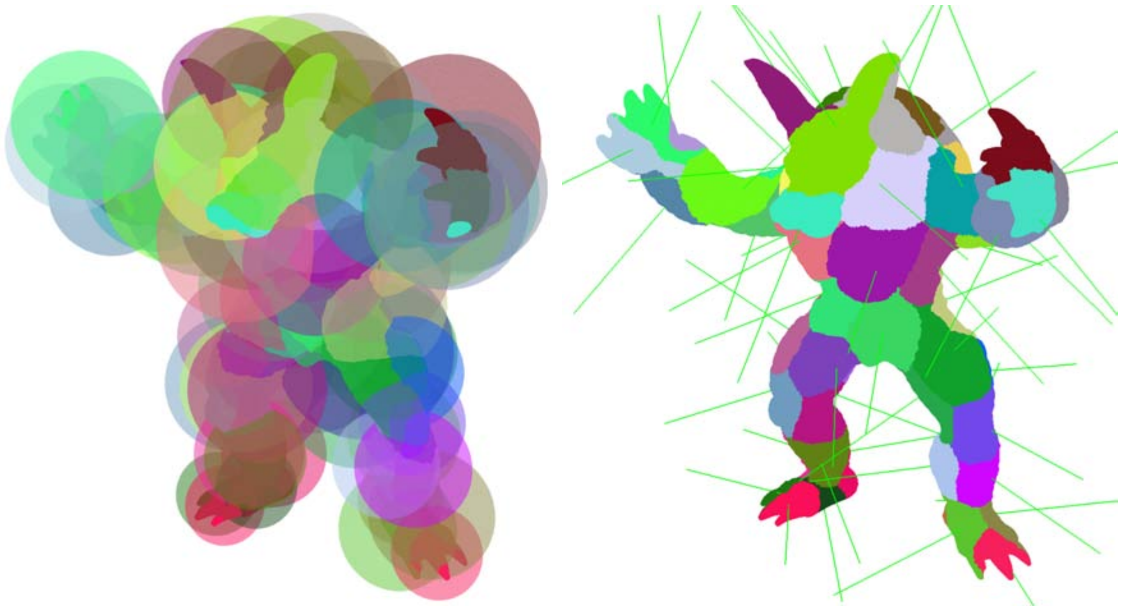


Figure 6.2: Cluster bounding spheres and mean normals.

Visual importance estimation (I_{est}) takes into account a cluster's total, mean and peak saliency values. Highest contribution is obtained when high values exist for the three saliency metrics, giving us the visual importance estimate:

$$I_{est} = (\mathscr{S}_c * peak(\mathscr{S}_c) * \mu(\mathscr{S}_c))/\mathscr{S}_{mesh} \tag{6.4}$$

where $\mathscr{S}_{mesh}$ denotes accumulated saliency for the entire mesh. Examples of importance estimation from different points of view are shown in Figure 6.3.

Contraction distribution sorts clusters by $A_{est} * I_{est}$. Those with lower area and importance values are rendered with lower resolution, while higher values have more detail preserved. For practical purposes, we define a cluster's resolution by the number of vertex

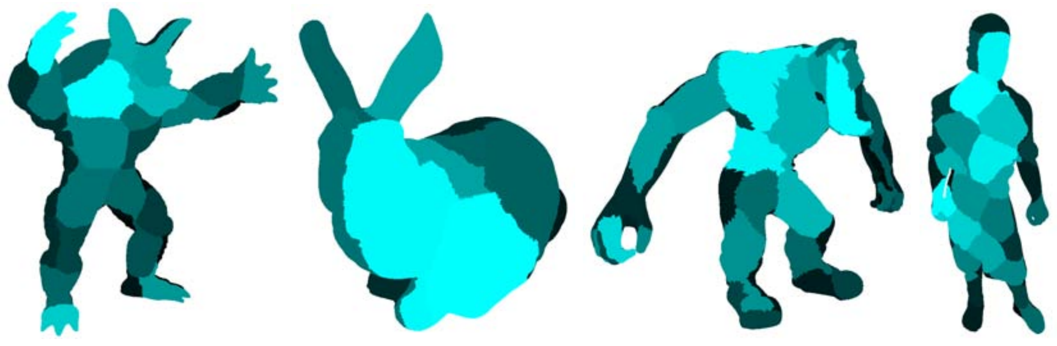


Figure 6.3: Visual Importance Estimation: Lighter regions have greater importance values.

pair contractions to be performed over its multi-resolution representation; contractions are then distributed across the cluster list in order. The lowest importance cluster is assigned:

$$R_{local} = \lambda_{perc} * R_{global} \tag{6.5}$$

contractions, where λ_{perc} is a user-defined percentage value. R_{local} is clamped to the maximum number of contractions allowed on the cluster. This number is then subtracted from the global contraction target and the next cluster is evaluated, until the global contraction target has been reached. If R_{global} is not reached after the last cluster on the list is evaluated, the process continues from the beginning of the list. For all our experiments λ_{perc} was set to 0.02.

Neighbor matching is executed after all clusters have received a contraction target: a vertex pair contraction can only be performed if all contractions leading to it have already been performed. To guarantee this, we keep the global order of the last required pair contraction associated with each vertex contraction. For each cluster, the largest parent-contraction order is determined, and neighboring clusters are evaluated; if the neighbor contraction stop point is smaller than this largest contraction order, the neighbor stop point is replaced by this value. This ensures gap-free boundaries, and minimizes face overlap problems. Some overlap may still occur, especially over regions where significant difference in resolution targets for a local group of clusters is encountered, but such artifacts are rare.

6.2 Level of Detail Selection

Once a resolution target is determined, we must perform vertex pair contractions over each cluster vertex list until its target is reached. While this is a straightforward process, it is considerably time consuming: index lists must be processed several times until all contractions are performed. Updated lists must be uploaded to GPU memory for rendering, thus increasing CPU-GPU bandwidth usage. To speed up this process, we introduce a GPU-based level-of-detail selector, which performs the entire process on GPU memory, and reduces both execution and index list update times. Our GPU-based level-of-detail selector is composed of a single fragment shader, two framebuffer objects and a set of data textures.

Five textures contain all the information required to initialize and run the vertex pair contraction process. The first two textures encode, for each vertex of a given cluster, its initial and subsequent contractions:

- Base Vertex Index (*BVI*) texture (2D, RGBA32F): initial contraction: vertex index (v_i), contracted vertex index ($VC(v_i)$), 1D coordinates for the CVI texture (i_{cvi}) and for both the SCI and CSP textures (i_{sci}).
- Contracted Vertex Index (*CVI*) texture (2D, RGBA32F): subsequent contractions, same format as the BVI texture.

Since a vertex is processed separately for each cluster, we need to ensure that a vertex that appears on multiple clusters undergoes the same number of contractions, thus avoiding cracks in the final mesh. The remaining three textures encode information used in this process:

- Shared Cluster Indices (*SCI*) texture (2D, RGBA32F): cluster IDs for all clusters associated with a given vertex.
- Vertex Contraction Order (*VCO*) texture (2D, R32F): global pair contraction order of a given vertex.
- Cluster Stop Point (*CSP*) texture (1D, R32F): cluster stop points selected during resolution estimation.

Framebuffer objects (*FBOs*) are used for both processing (*FBOP*) and data format conversion (*FBOC*). The first (*FBOP*) is used during the iterative vertex pair contraction process and matches the information stored on the BVI and CVI textures, having two surfaces (*FBOPr* and *FBOPw*) for ping-pong rendering. The second (*FBOC*) is used for data type conversion and copy. Both framebuffer objects share the size defined for *BVI*. The vertex contraction fragment shader takes *CVI*, *SCI*, *VCO* and *CSP* as input textures, as well as the texture associated with *FBOPr* and the widths and heights for the *CVI*, *SCI* and *VCO* textures. It is executed over the domain defined by *FBOP*, where each pixel maps directly to one vertex index in the clustered index buffer. Level-of-detail selection works as follows:

Algorithm 1 VERTEX CONTRACTION ALGORITHM

```
1: Read surface on FBOPr with initial BVI values
2: for $i = 1$ to $maxContractionLevel$ do
3:     Execute shader over FBOPr, writing on FBOPw
4:     Swap FBOPr and FBOPw
5: end for
6: Write contents of FBOPr to FBOC
7: Copy FBOC to vertex index buffer for rendering
```

The value *maxContractionLevel* represents the highest level of any pair contraction target that must be reached among all clusters, and can be calculated with no additional complexity during the cluster resolution determination step. Figure 6.4 further details the contraction shader.

After the vertex contraction shader has executed *maxContractionLevel* times, *FBOPr* contains the updated index array. We copy these values, through a single rendering pass, to

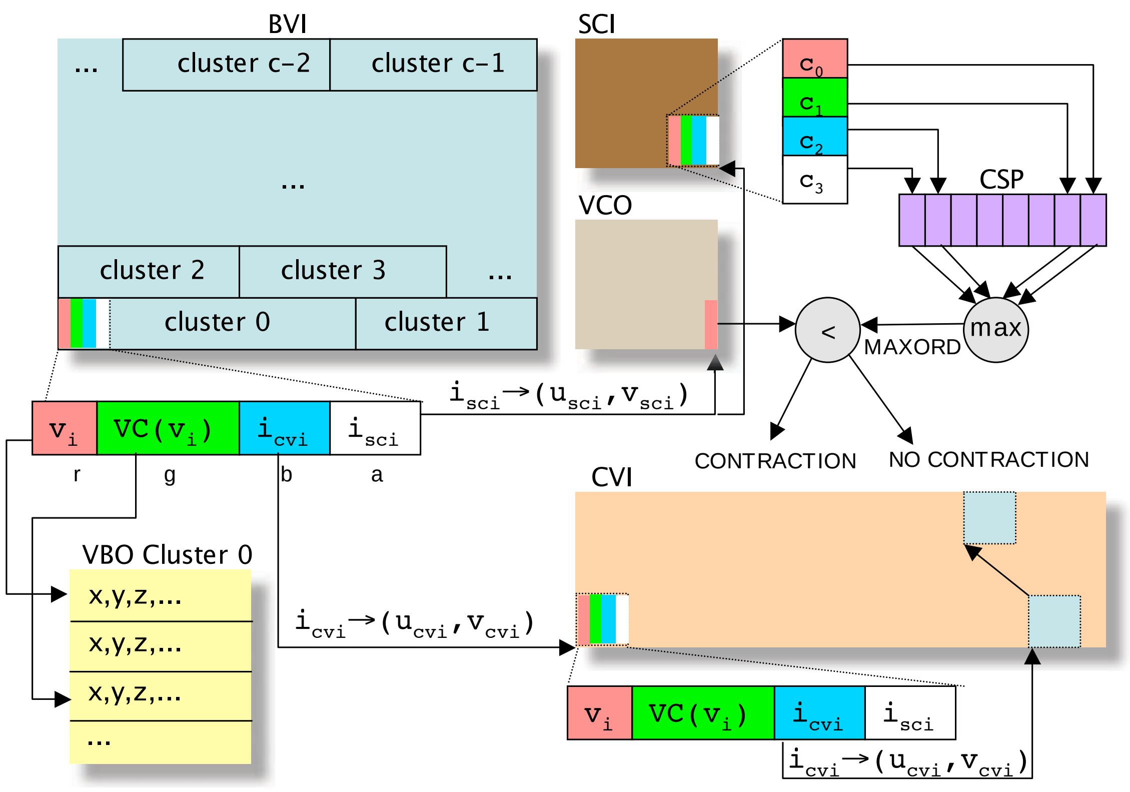


Figure 6.4: Vertex Contraction Shader: First it checks if a vertex has a contraction target (channel *G*: vertices with no target receive value -1). If no target exists, the value is maintained; otherwise, coordinates stored on channels *B* and *A* are decoded. The first is used to load contraction data from *CVI*, while the second is used to access data from both *SVI* and *VCO*. Each channel on *SCI* is used for additional texture indirection, now reading from CSP; the four values read from *CSP* are compared, and the highest order value *MAXORD* is selected. If the global vertex order, read from *VCO*, is smaller than *MAXORD* the vertex must be contracted, and the output value is updated with the one read from *CVI*.

FBOC - which shares the same internal format as the index buffer - from whence values are copied to the index buffer to be used for rendering. For some graphics cards, we observed rounding problems when *unsigned integer* values are manipulated by fragment shaders. *Unsigned short* did not present such errors, with the added benefit of lower data transfer times, so this format was adopted instead. However, it limits clusters to a maximum of 2^{16} vertices, a restriction which can be easily enforced during pre-processing.

6.3 Rendering

Rendering is relatively straightforward. We have a single vertex index buffer, where cluster-localized vertex indices are stored; this buffer is bound at all times. For each cluster, its associated vertex data buffer (where vertex positions, normals, color, etc. are stored) is bound, and the portion of the index buffer relative to the cluster is rendered. It is interesting to note that other cluster-level optimization algorithms, such as partial ordering (NEHAB; BARCZAK; SANDER, 2006), can be used in conjunction with our method without the need for adaptations.

7 RESULTS

7.1 Multi-Pose Mesh Saliency

Previous sections described a method to extract multi-pose saliency information from a base mesh and a set of key poses, using an improved single-pose saliency computation technique as the starting step. To evaluate the effectiveness of multi-pose saliency, we have modified the QSlim method, weighting the vertex error quadrics by a weight map derived from multi-pose mesh saliency.

Figure 3.4 compares perceptual importance as estimated by standard mesh saliency to mesh saliency using geodesic distance. Notice how deforming one of the spheres causes variations in saliency values on its undeformed neighbor when the Euclidean distance is used — the same does not happen when geodesic distance is used; mesh saliency values are correctly preserved. Comparison of saliency values for the Cylinder dataset can be seen in Figure 7.1, single-pose saliency values are given using our modified saliency metric. Observe the differences in the region near the joint; single-pose saliency fails to report any importance for such regions whereas our multi-pose saliency approach correctly assigns high saliency values to them.

For our experiments, all datasets were simplified from the base mesh, deformations were re-applied over the simplified representations using the original skeleton and bone influence fields. An example of simplification results for the base mesh is shown on Figure 7.2. Notice how detail is better preserved around the visually interesting portions of the deformed regions when multi-pose saliency is used as the weighting factor. For the Sphere dataset detail is better preserved on the side that suffers deformation with multi-pose salient simplification whereas single-pose salient simplification results in a roughly uniform tessellation.

Figure 4.3 shows results for the Arm dataset, with deformation re-applied after simplification. Regions around the joint are better preserved with multi-pose salient simplification, resulting in an improved visualization when deformation is applied; low-saliency, static regions undergo greater simplification. Single-pose saliency better preserves static features present on the base mesh, at the cost of lower detail on joints and deformed regions, while using standard QSlim we achieve more uniform simplification along the surface of the object, at the cost of ignoring possibly salient regions. Figure 4.4 shows simplification results for all datasets, while figure 4.5 has a close-up look to illustrate how detail is kept on features that are likely to undergo deformation.

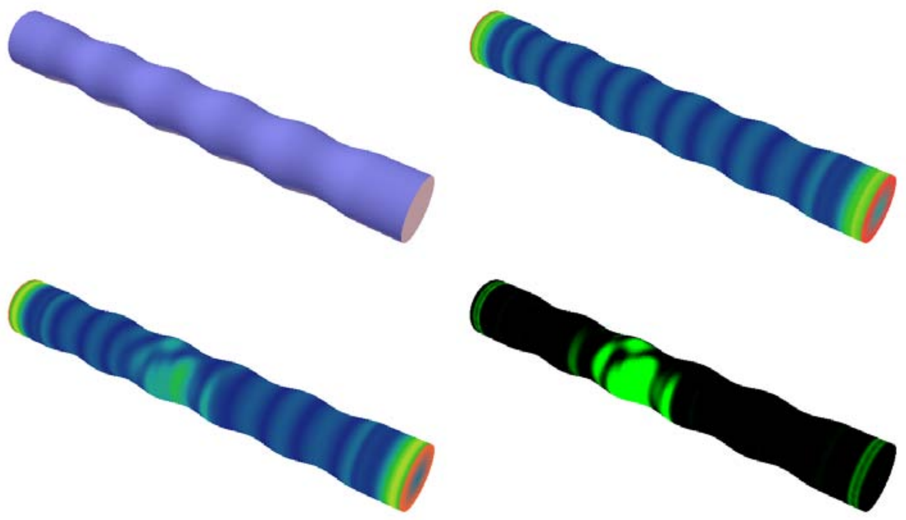


Figure 7.1: Difference between single-pose and multi-pose saliency for the multi-pose cylinder of Figure 4.1. Top row: base mesh, and single-pose saliency for base mesh. Bottom row: multi-pose saliency and difference between multi-pose and single-pose saliency for the base mesh. Lighter colors indicate greater difference.

7.2 Saliency-Guided Mesh Segmentation

We have tested our segmentation algorithm with several datasets, covering a broad range of geometric and topological complexity. All experiments were conducted on an Athlon 64 3500+, with 2GB of RAM and an NVIDIA GeForce 8800 GTX 768 graphics card. Table 7.1 gives more specific information on each dataset used. Execution times for segmentation are given in Table 7.2; while k-means clustering takes considerably less time than our technique, its results are relatively poor. Times for propagative segmentation are similar to those of our technique, but it fails to detect salient features, generally targeting flat regions. Figures 7.7 to 7.11 illustrate the segmentation results for several datasets.

Table 7.1: Datasets

Dataset	**Vertices**	**Faces**	**Edges**	**Clusters**
Grunt	26143	52282	78423	45
Bunny	35947	69451	104288	10
Hand	136663	273060	409724	10
Isis	187644	375284	562926	30
Ganesh	206618	413236	619827	30
Dragon	240057	480076	720114	45
Goddess1	137406	274822	412231	30
Goddess2	523578	1047330	1570923	30
Armadillo	172974	345944	518916	45
Laçador	653891	1307794	1961691	45

Significantly better frame-rates - when compared both to full resolution representations and optimization through CPU-based pair contraction - were observed for all models when using our method, for meshes rendered at 5% of the original resolution. Computation time for each mesh update was amortized over 10 frames for both the GPU and CPU implementations. Table 7.3 summarize these results.

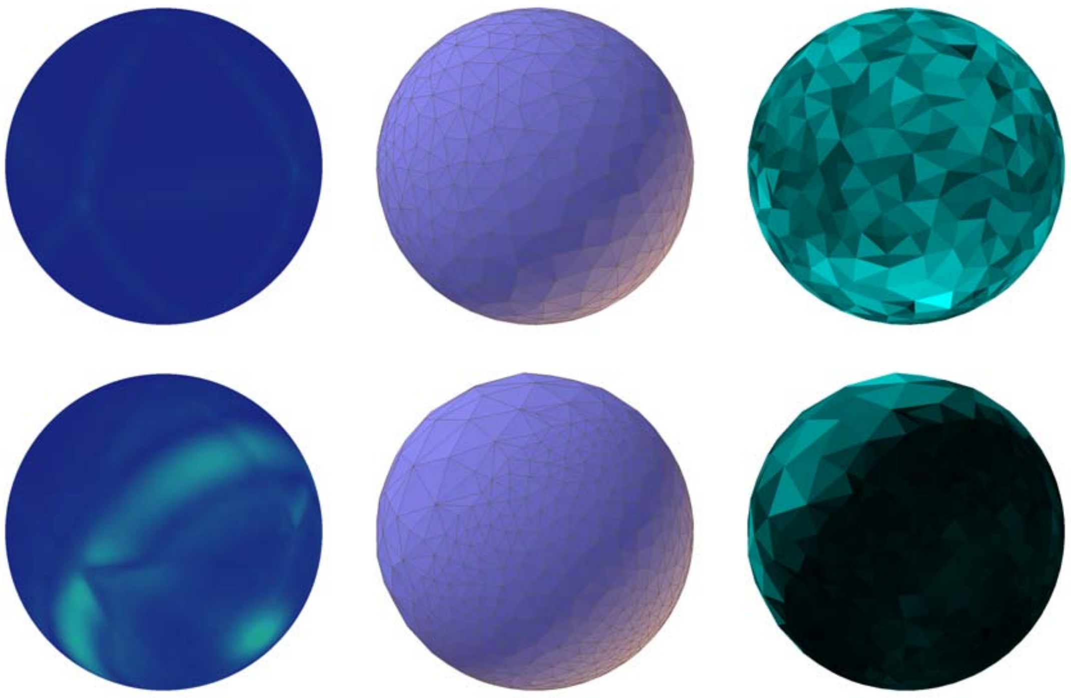


Figure 7.2: Comparison of simplification results for the multi-pose sphere of Figure 4.1. From left to right: mesh saliency, simplified mesh using QSlim weighted by saliency values, relative triangle sizes for simplified mesh. The top row shows results for single-pose saliency, computed from the base mesh; the bottom row displays results for multi-pose mesh saliency. The different deformations undergone by the Sphere model can be seen on Figure 4.1.

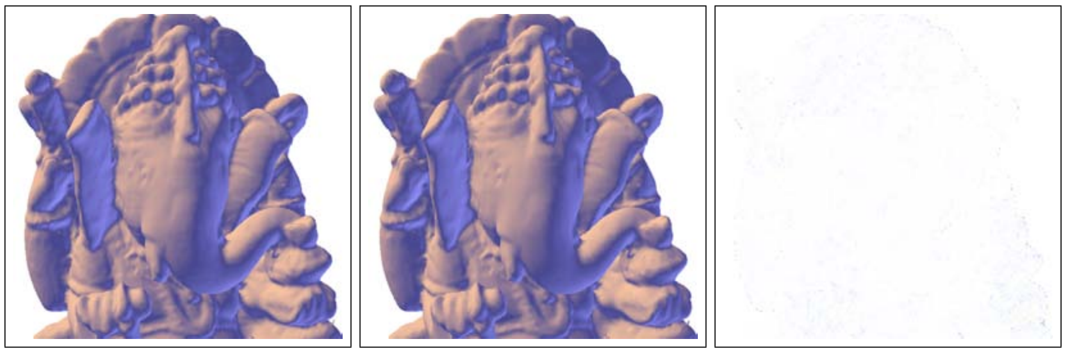
Advantage gained in relation to rendering the full-resolution mesh decreases when less optimization is performed, but even at as much as 60% of the original resolution improvements in framerate are significative when amortization is performed. Still, this loss could be minimized through greater amortization, setting a cutoff point (optimizing meshes only when significant simplification is required) or other render-time resolution selection techniques. Rendering multiple instances of the same optimized mesh, such as characters or repeating scenery objects in massively populated environments, will also decrease relative costs. As an example, rendering nine instances of the Armadillo dataset simplified to 60% of its original resolution more than doubles the framerate: from 20 FPS when rendering the full resolution mesh to 42 FPS when our method is enabled, with minimal visual loss. Results practically indistinguishable from the original mesh can be obtained at up to 35% simplification, as shown on Figure 7.3, and with minimal difference at 5% (Figures 7.4 and 7.6).

Table 7.2: Saliency computation and clustering times, in seconds.

Dataset	**Saliency Comp.**	**Salient Seg.**	**Prop. Seg.**	**K-Mean Seg.**
Grunt	8.516	13.859	21.203	0.14
Bunny	5.187	30.359	64.844	0.172
Hand	78.219	247.39	218.687	0.625
Isis	136.781	861.547	1104.55	2.002
Ganesh	80.516	545.328	925.672	2.531
Dragon	187.906	865.437	602.187	3.563
Goddess1	56.531	578.61	833.843	0.906
Goddess2	1049.27	7534.73	4058.34	3.438
Armadillo	91.047	417.265	623.688	1.609
Laçador	1747.53	6945.98	7141.34	8.063

Table 7.3: FPS for full mesh, CPU and GPU optimization.

Dataset	**No**	**CPU**	**GPU**		
	Opt.	**(5%)**	**(5%)**	**(35%)**	**(60%)**
Grunt	1087	685	2220	1759	1419
Bunny	817	475	2147	1507	1126
Hand	241	116	823	518	357
Isis	154	68	505	266	190
Ganesh	160	64	482	283	222
Dragon	127	55	398	336	261
Goddess1	216	127	839	423	328
Goddess2	64	26	272	134	84
Armadillo	184	93	783	341	262
Laçador	50	21	234	105	75



(a) Original (b) 35% (c) Difference

Figure 7.3: View-dependent rendering results for the Ganesh dataset, at 35% of original resolution, with framerate increasing from 64 to 283 FPS. Original mesh (a), simplified mesh (b) and difference image — nearly no difference (c).

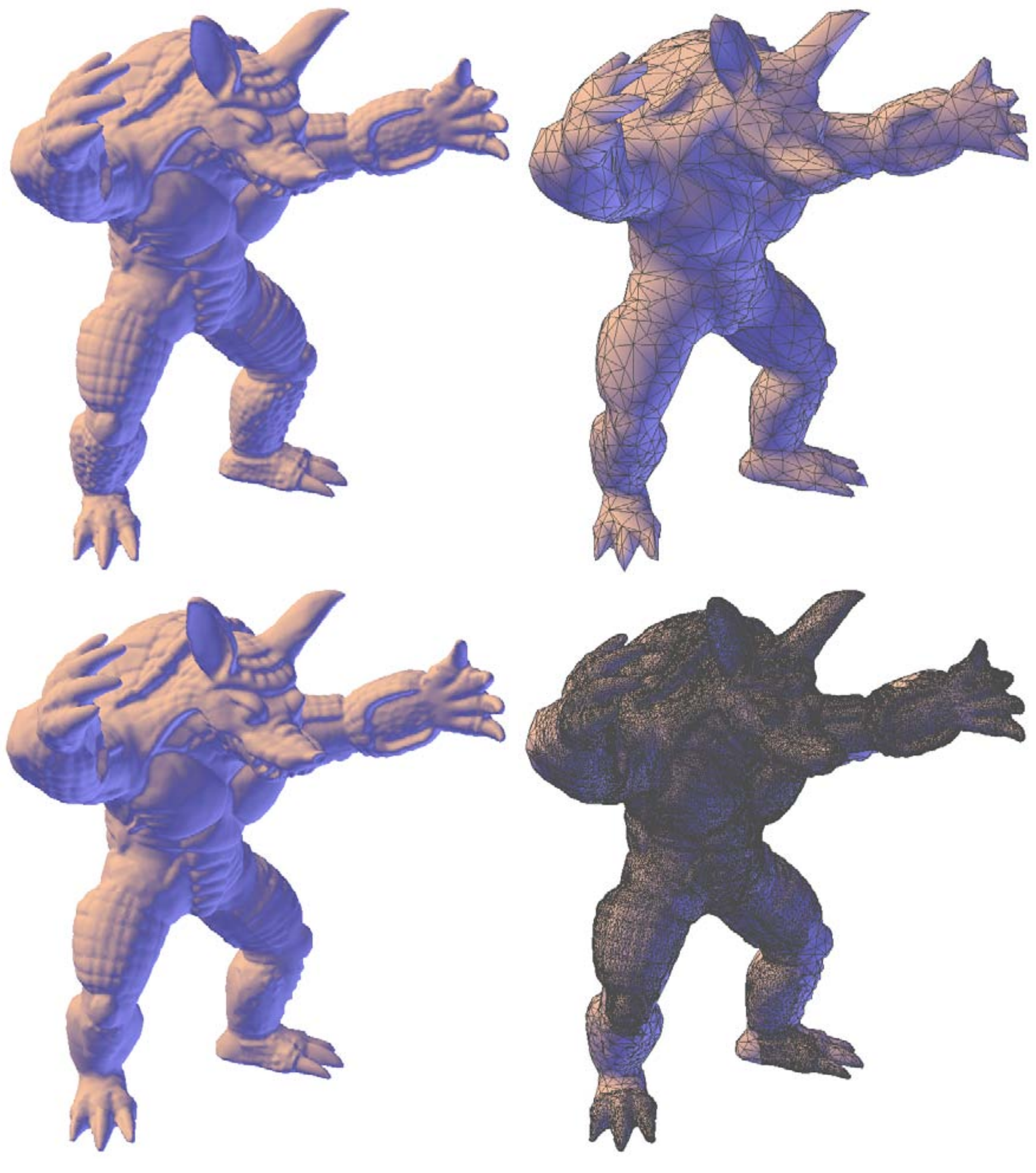


Figure 7.4: Armadillo results. Original Mesh, QSlim simplification to 1% (progressive mesh backstop point), Saliency-guided Simplification using clustering to 35% of original resolution (with and without wireframe). Notice how most relevant details were preserved on the mesh simplified to 35% when compared to the full resolution.

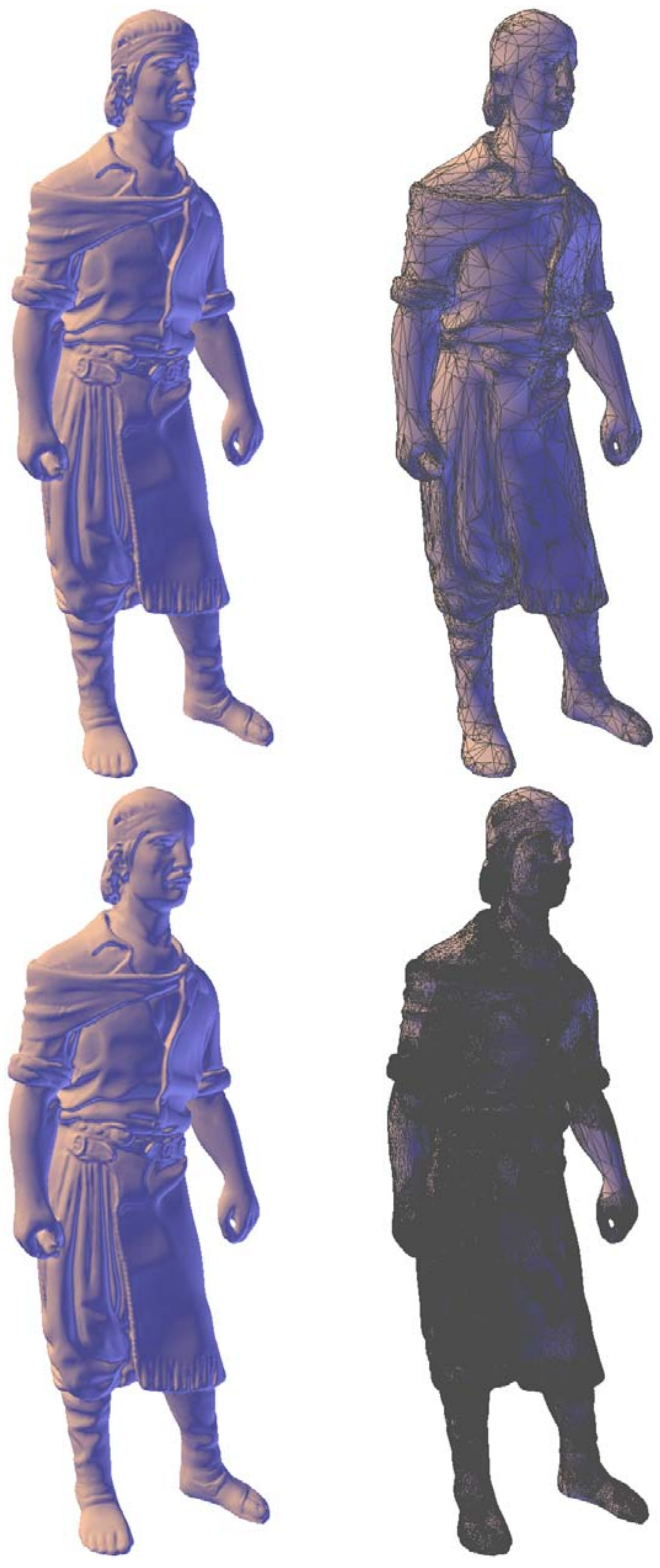


Figure 7.5: Laçador results. Original Mesh, QSlim simplification to 1% (progressive mesh backstop point), Saliency-guided Simplification using clustering to 35% of original resolution (with and without wireframe). Notice how most relevant details were preserved on the mesh simplified to 35% when compared to the full resolution.

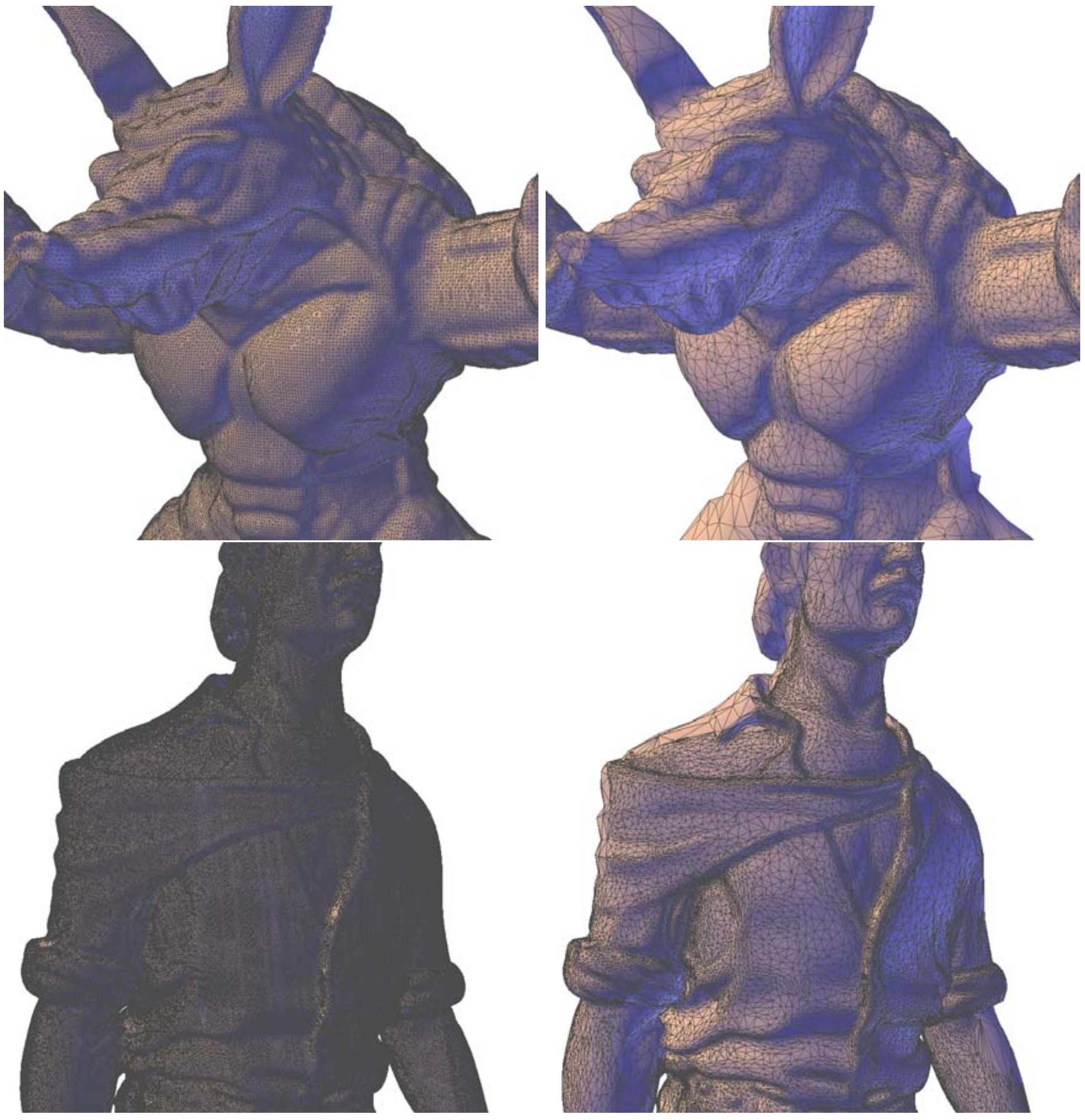
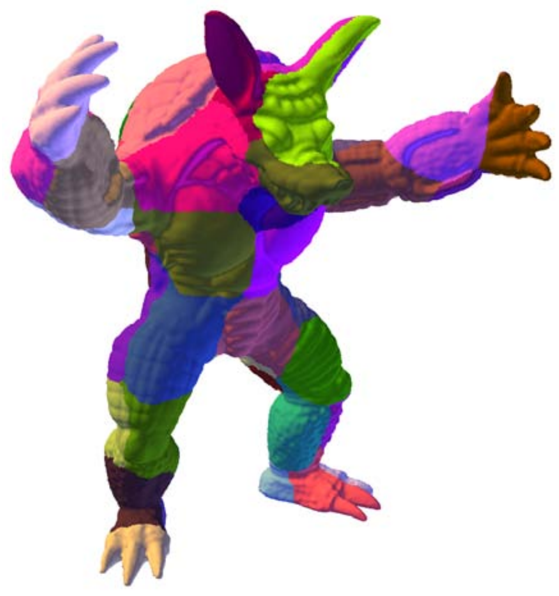
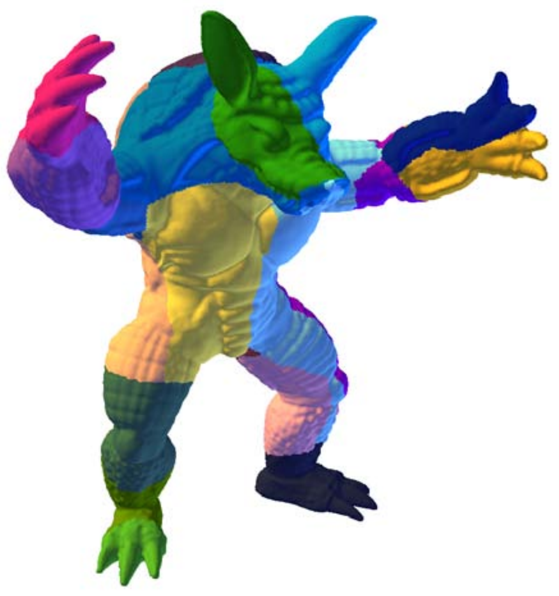


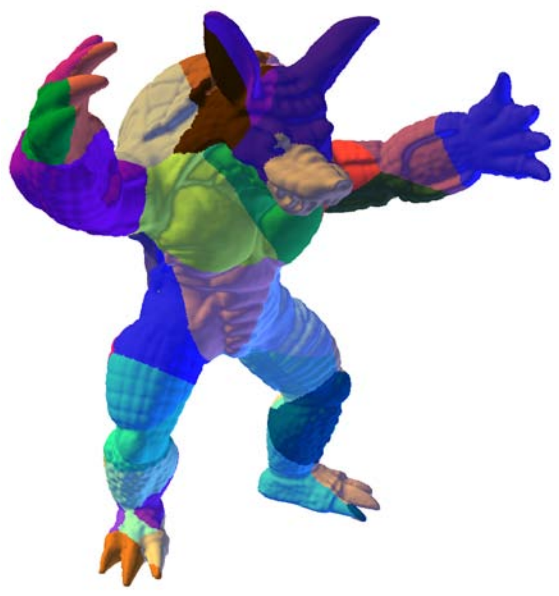
Figure 7.6: Armadillo and Laçador Results. Detail from Original Mesh and Saliency-guided Simplification to 5% of original resolution.



(a)

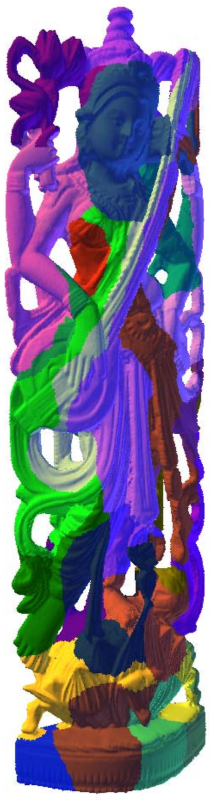


(b)

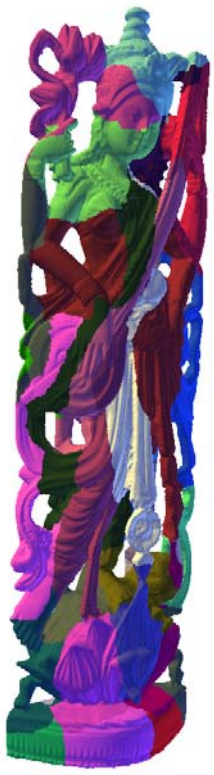


(c)

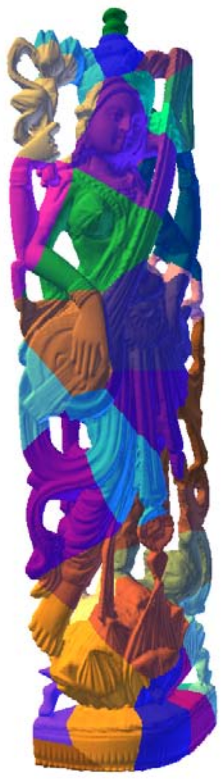
Figure 7.7: Comparison of Clustering Techniques for the Armadillo: Salient Clustering (a), Distance+normal Propagative Clustering (b) and K-Means Clustering (c). Notice the improved cluster convergence around interesting features, such as faces and hands when salient clustering is used.



(a)

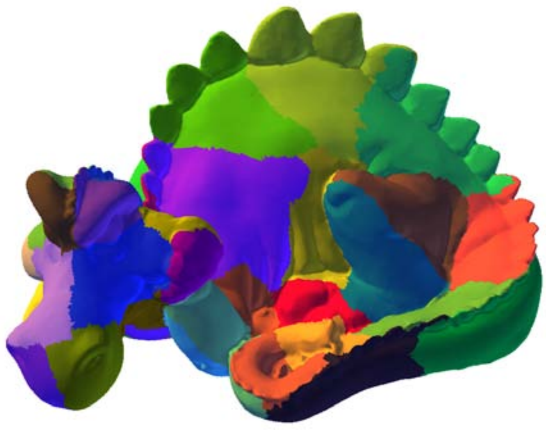


(b)

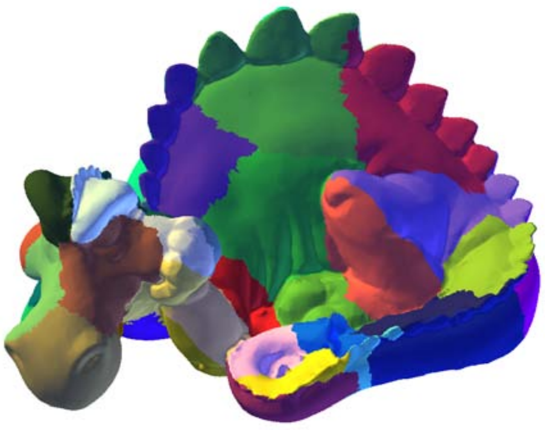


(c)

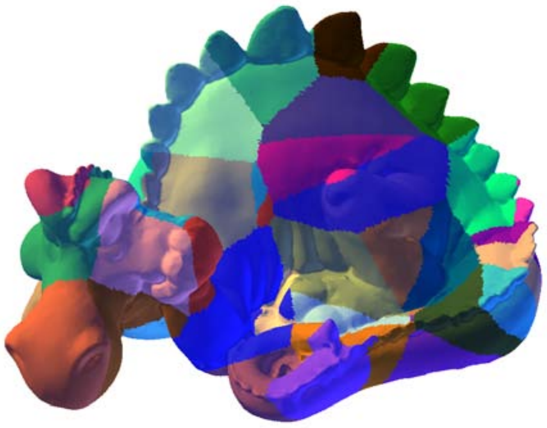
Figure 7.8: Comparison of Clustering Techniques for the Goddess2 Dataset: Salient Clustering (a), Distance+normal Propagative Clustering (b) and K-Means Clustering (c).



(a)

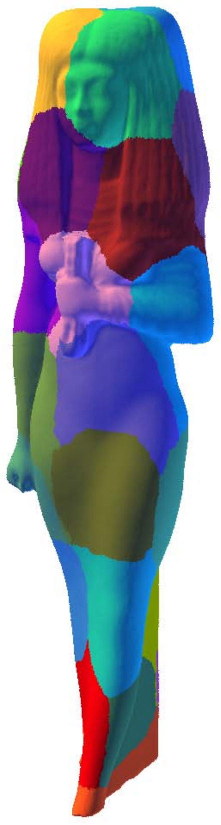


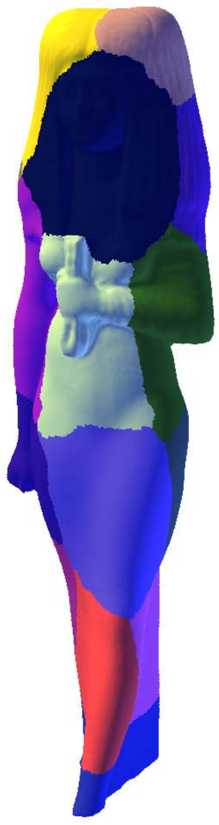
(b)



(c)

Figure 7.9: Comparison of Clustering Techniques for the Dragon Dataset: Salient Clustering (a), Distance+normal Propagative Clustering (b) and K-Means Clustering (c).


(a)


(b)

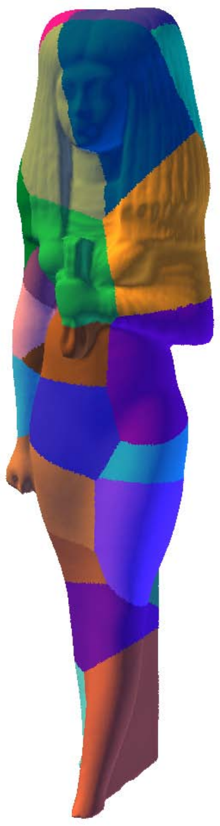
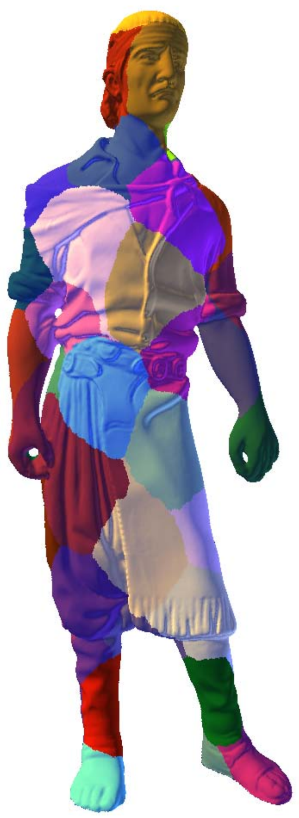
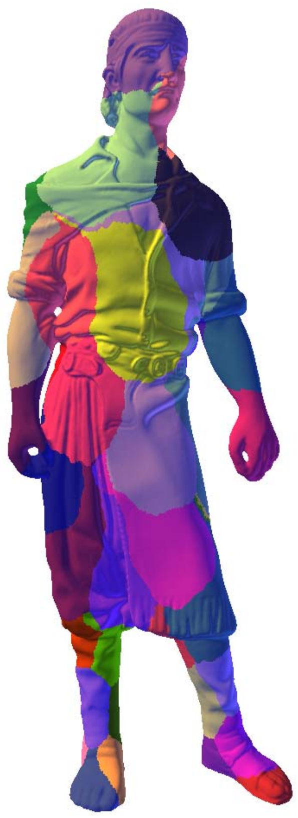

(c)

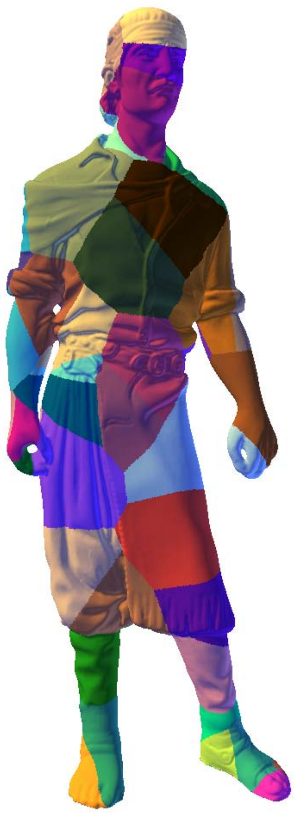
Figure 7.10: Comparison of Clustering Techniques for the Isis Dataset: Salient Clustering (a), Distance+normal Propagative Clustering (b) and K-Means Clustering (c).



(a)



(b)



(c)

Figure 7.11: Comparison of Clustering Techniques for the Laçador Dataset: Salient Clustering (a), Distance+normal Propagative Clustering (b) and K-Means Clustering (c).

8 CONCLUSIONS AND FUTURE WORK

In this work we have extended the concept of mesh saliency and integrated it into several applications. This chapter summarizes our contributions and improvements when compared to competing techniques and indicates possible courses for future research.

8.1 Multi-Pose Mesh Saliency

We have extended the concept of mesh saliency to take into account variations on perceptual importance caused by mesh deformations. An improved approach to computation of single-pose mesh saliency was presented, using geodesic distance instead of the Euclidean distance for local neighborhood determination, generating more robust perceptual importance estimations. A method to calculate a single multi-pose mesh saliency value from a base mesh and a set of key poses or deformations was also introduced, using our modified single-pose saliency metric to analyze each pose. Saliency values are then combined, taking into account saliency persistency and variation across the set of key poses; regions with high saliency across all poses receive higher multi-pose saliency values than those with high saliency only on few poses.

Effectiveness of our multi-pose saliency metric is demonstrated through application on a mesh simplification problem. We show that weighting the simplification process with multi-pose saliency results in simplifications that better fit the key poses while maintaining detail in the regions of interest across the pose set when compared to simple error-guided simplification, or simplification weighted by single-pose mesh saliency.

Multi-pose mesh saliency provides a reasonable metric of visual importance for deformable objects. Currently our method takes into account only mesh curvature information across a set of key poses. It will be interesting to generalize it to not only take into account other visual attributes, such as color and texture, but also deformation velocity and other spectral attributes of the deformation process. Extending our technique to work from a base mesh with associated skeleton and skinning information, as on (DECORO; RUSINKIEWICZ, 2005), can prove to be useful for already skinned objects, requiring no further work from the artist. Integrating multi-pose saliency with a view-dependent rendering framework is another promising research prospect.

8.2 Saliency-Guided Mesh Segmentation

We described a method to reliably segment a triangle mesh into a set of clusters containing visually interesting features. The method uses mesh saliency as a computational model of perceptual importance to iteratively determine face clusters through propagation from a starting point selected from a pool of high-saliency vertices. No user input is required, and salient regions are correctly captured. We show how segmentation of equal or better quality than comparable methods is achieved, in similar computational times.

Additional validation was performed through a view-dependent rendering system, where salient clusters were assigned different mesh resolutions at render-time based on their visual importance, calculated from viewpoint information and combined cluster saliency data; high simplification rates were achieved with little visual loss, resulting in greatly improved framerates.

Further refinements may be possible, though. Saliency computation can benefit from better metrics for neighborhood determination, such as geodesic distance, consequently improving segmentation results; salient feature detection — recognition of individual interesting features by local saliency peak analysis — may improve both the results of the segmentation process and of the cluster importance estimation step. Validation of our visual importance metric through user studies can provide a more solid basis for employing mesh saliency as an automatic metric for perceptual importance. We intend to investigate these directions as a part of our future work.

REFERENCES

ATTENE, M. et al. Mesh Segmentation - A Comparative Study. In: IEEE INTERNATIONAL CONFERENCE ON SHAPE MODELING AND APPLICATIONS, SMI, 2006. **Proceedings...** [S.l.]: IEEE Computer Society, 2006.

CARR, N. A. et al. Rectangular Multi-Chart Geometry Images. In: EUROGRAPHICS SYMPOSIUM ON GEOMETRY PROCESSING, 4., 2006. **Proceedings...** [S.l.: s.n.], 2006.

DECORO, C.; RUSINKIEWICZ, S. Pose-independent Simplification of Articulated Meshes. In: ACMSIGGRAPH SYMPOSIUM ON INTERACTIVE 3D GRAPHICS AND GAMES, SI3D, 2005. **Proceedings...** New York, NY, USA: ACM Press, 2005.

GAL, R.; COHEN-OR, D. Salient geometric features for partial shape matching and similarity. **ACM Trans. Graph.**, New York, NY, USA, v.25, n.1, p.130–150, 2006.

GARLAND, M.; HECKBERT, P. S. Surface simplification using quadric error metrics. **Computer Graphics**, New York, v.31, n.4, p.209-216, Nov. 1997. Work presented on Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), 1997.

HECKBERT, P. S.; GARLAND, M. Optimal Triangulation and Quadric-Based Surface Simplification. **Computational Geometry**, [S.l.], v.14, p.49–65, 1999.

HOPPE, H. Progressive meshes. In: COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH, 23., 1996. **Proceedings...** New York, NY, USA: ACM Press, 1996. p.99–108.

HOULE, J.; POULIN, P. Simplification and real-time smooth transitions of articulated meshes. In: GRAPHICS INTERFACE, GRIN, 2001. **Proceedings...** Toronto, Ont., Canada: Canadian Information Processing Society, 2001. p.55–60.

HOWLETT, S.; HAMILL, J.; O'SULLIVAN, C. Predicting and Evaluating Saliency for Simplified Polygonal Models. **ACM Trans. Appl. Percept.**, New York, NY, USA, v.2, n.3, p.286–308, 2005.

ITTI, L.; KOCH, C. Computational Modeling of Visual Attention. **Nature Reviews Neuroscience**, [S.l.], v.2, n.3, p.194–203, Mar.2001.

ITTI, L.; KOCH, C.; NIEBUR, E. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, Los Alamitos, CA, USA, v.20, n.11, p.1254–1259, 1998.

JI, J. et al. View-dependent refinement of multiresolution meshes using programmable graphics hardware. **Vis. Comput.**, Secaucus, NJ, USA, v.22, n.6, p.424–433, 2006.

JULIUS, D.; KRAEVOY, V.; SHEFFER, A. D-Charts: quasi-developable mesh segmentation. **Computer Graphics Forum**, Amsterdam, v.24, n.3, p.581-590, 2005.

KATZ, S.; LEIFMAN, G.; TAL, A. Mesh segmentation using feature point and core extraction. **The Visual Computer**, [S.l.], v.21, n.8-10, p.649–658, 2005.

KATZ, S.; TAL, A. Hierarchical mesh decomposition using fuzzy clustering and cuts. **ACM Transactions on Graphics**, New York, v.22, n.3, p.954-961, July 2003.

KHO, Y.; GARLAND, M. User-guided simplification. In: SYMPOSIUM ON INTERACTIVE 3D GRAPHICS AND GAMES, SI3D, 2003. **Proceedings...** New York, NY, USA: ACM Press, 2003. p.123–126.

KOCH, C.; ULLMAN, S. Shifts in selective visual attention: towards the underlying neural circuitry. **Human Neurobiology**, [S.l.], v.4, n.4, p.219–227, 1985.

LAI, Y.-K. et al. Feature sensitive mesh segmentation. In: ACM SYMPOSIUM ON SOLID AND PHYSICAL MODELING, SPM, 2006. **Proceedings...** New York, NY, USA: ACM Press, 2006. p.17–25.

LEE, C. H.; VARSHNEY, A.; JACOBS, D. W. Mesh saliency. **ACM Trans. Graph.**, New York, NY, USA, v.24, n.3, p.659–666, 2005.

LINDE, Y.; BUZO, A.; GRAY, R. M. An Algorithm for Vector Quantizer Design. **IEEE Transactions on Communications**, [S.l.], v.28, p.84–95, 1980.

LUEBKE, D. A developer's survey of polygonal simplification algorithms. **IEEE Computer Graphics and Applications**, [S.l.], v.21, n.3, p.24–35, 2001.

LUEBKE, D.; HALLEN, B. Perceptually-Driven Simplification for Interactive Rendering. In: EUROGRAPHICS WORKSHOP ON RENDERING TECHNIQUES, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.223–234.

LUEBKE, D. et al. **Level of Detail for 3D Graphics**. New York, NY, USA: Elsevier Science, 2002.

MOHR, A.; GLEICHER, M. **Deformation Sensitive Decimation**. Madison: University of Winsconsin, 2003. Technical Report.

MORTARA, M. et al. Blowing Bubbles for Multi-Scale Analysis and Decomposition of Triangle Meshes. **Algorithmica**, Secaucus, NJ, USA, v.38, n.1, p.227–248, 2003.

NEHAB, D.; BARCZAK, J.; SANDER, P. V. Triangle order optimization for graphics hardware computation culling. In: SYMPOSIUM ON INTERACTIVE 3D GRAPHICS AND GAMES, SI3D, 2006. **Proceedings...** New York, NY, USA: ACM Press, 2006. p.207–211.

O'SULLIVAN, C. et al. Perceptually Adaptive Graphics. In: EUROGRAPHICS, 2004. **Proceedings...** [S.l.]: INRIA and the Eurographics Association, 2004. p.141–164. (State of the Art Reports, n.STAR-6)

QU, L.; MEYER, G. W. Perceptually driven interactive geometry remeshing. In: SYMPOSIUM ON INTERACTIVE 3D GRAPHICS AND GAMES, SI3D, 2006. **Proceedings...** New York, NY, USA: ACM Press, 2006. p.199–206.

REDDY., M. Perceptually Optimized 3D Graphics. **IEEE Computer Graphics and Applications**, [S.l.], v.21, n.5, p.68–75, 2001.

SANDER, P. V. et al. Multi-chart geometry images. In: EUROGRAPHICS/ACM SIGGRAPH SYMPOSIUM ON GEOMETRY PROCESSING, SGP, 2003. **Proceedings...** Aire-la-Ville, Switzerland: Eurographics Association, 2003. p.146–155.

SCHMALSTIEG, D.; FUHRMANN, A. **Coarse viewdependent levels of detail for hierarchical and deformable models**. [S.l.]: University of Vienna, 1999. Technical Report.

SURAZHSKY, V. et al. Fast exact and approximate geodesics on meshes. **ACM Transactions on Graphics**, New York, v.24, n.3, p.553-560, July 2005.

TSOTSOS, J. K. et al. Modeling visual attention via selective tuning. **Artif. Intell.**, Essex, UK, v.78, n.1-2, p.507–545, 1995.

WATSON, B.; WALKER, N.; HODGES, L. F. Supra-threshold control of peripheral LOD. **ACM Transactions on Graphics**, [S.l.], v.23, n.3, p.750–759, 2004.

WOLFE, J. M. Visual search in continuous, naturalistic stimuli. **Vision research**, [S.l.], v.34, n.9, p.1187–1195, 1994.

ZHANG, H.; LIU, R. Mesh Segmentation via Recursive and Visually Salient Spectral Cuts. In: VISION, MODELING, AND VISUALIZATION, 2005, Berlin. **Proceedings...** [S.l.]: Akademische Verlagsgesellschaft Aka GmbH, 2005. p.429–436.