Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

Saliência em Malhas 3D e Aplicações em Renderização Não Fotorealística

Felipe Soares Queiroga

Trabalho de Graduação

Recife
1 de Dezembro de 2009

Universidade Federal de Pernambuco
Centro de Informática

Felipe Soares Queiroga

Saliência em Malhas 3D e Aplicações em Renderização Não Fotorealística

Trabalho apresentado ao Curso de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: *Marcelo Walter*
Co-orientador: *Silvio de Barros Melo*

Recife
1 de Dezembro de 2009

Agradecimentos

Depois de tanto trabalho para a escrita das páginas deste documento, veio o momento mais difícil: agradecer àqueles que muito auxiliaram desde o início. Pelo fato de estar bastante cansado com a escrita e detalhamento de todo o documento, nesta parte é sempre mais difícil entrar em detalhes e se prolongar demais. Porém, é o momento mais gratificante em que posso dizer a todos aqueles que tanto me ajudaram, durante estes longos meses de escrita, muito obrigado. Por essa razão, espero não esquecer de citar nenhuma das pessoas que tanto me ajudaram neste momento. Mas, caso ocorra, gostaria de me desculpar e culpar todo o trabalho que tive durante a escrita e desenvolvimento deste projeto.

Em primeiro lugar, gostaria de agradecer à minha mãe, Flávia por ter lutado e sofrido bastante para que hoje, eu pudesse estar na universidade, mais precisamente fora da graduação, com este trabalho. Em segundo, mas não em menor importância, à toda a minha família. À minha irmã Juliana, minhas avós Yêda e Jandira, minha secretária Marta, minhas cachorras que foram verdadeiras companheiras de madrugadas e também aos meus tios, tias, primos, etc. Todos que entenderam os motivos de minha ausência durante alguns momentos em que tive que trabalhar duramente para obter este resultado. Além de momentos, não tão bem-humorados, devido à falta de horas no dia para comer, dormir, tomar banho e trabalhar na monografia.

Como não podia faltar, gostaria de agradecer aos leilões realizados pela Gol, pois graças a eles, tornaram-se financeiramente viável as minhas viagens para Florianópolis, onde pude encontrar-me com a pessoa que mais me dá paz nesse mundo tão cheio de loucuras. Apesar de toda a insanidade presente em Natália, ela é a pessoa mais capaz de me ajudar a viver com menos peso nas costas e foi a responsável pelo empurrão inicial para a escrita deste documento, além de sempre me incentivar e motivar durante todo o processo da realização deste trabalho.

Aos companheiros de situação que trabalhavam para realizarem os seus trabalhos e estavam sempre juntos nas madrugadas de trablho. Aos meus amigos, que sofreram e acompanharam os momentos sempre por perto. Em momentos que precisei trabalhar, tive o respeito deles, mesmo abdicando de eventos onde poderíamos encontrar-nos. Nos momentos que mais precisei me distrair, pude contar com eles! Quantos potes de sorvete foram contabilizados durante este processo? Difícil dizer...

Sem seguir uma ordem de importância, gostaria de agradecer especialmente ao professor Marcelo Walter, que foi um verdadeiro mentor durante todo o processo, e se mostrou bastante disponível para momentos de reuniões e auxílio com os processos da monografia e da aplicação. Apesar de ter a possibilidade de ter que realizar uma parte deste trabalho à distância, mostrou-se disponível para me orientar durante todo o processo, e o fez muito bem.

Por último, gostaria de agradecer às pessoas que leram esta seção até o final, pois isso é quase impossível!

"Do trabalho penoso da alma para a glória sem limites, do sofrimento para a plenitude de graça!"

—ISAÍAS (53:11)

Resumo

Recentes avanços, ocorridos na última década, nas áreas da Computação Gráfica, multimídia e de telecomunicações, influenciaram a evolução ocorrida na manipulação, transmissão e no uso de dados digitais na internet. Dados tridimensionais constituem, atualmente, a mídia emergente e neste contexto, objetos 3D, ou malhas que representam os mesmos, são utilizados em diversas aplicações centradas visualmente nos usuários, o que sugere a necessidade de incorporar maiores detalhes da percepção visual humana no processamento destas malhas. Apesar de já existirem excelentes trabalhos realizados que incorporam princípios de percepção relacionados a imagens, houve pouca atenção dedicada ao uso de métricas, inspiradas por percepção visual, para o processamento de malhas que compõem objetos 3D, utilizando apenas informações existentes na própria estrutura da mesma. As informações geométricas a respeito da malha, são capazes de demonstrar a repetição de padrões que, por sua vez, quando existente, inclusive em regiões de alta curvatura são visualmente desinteressantes, pois o que desperta o interesse visual é o incomum e a partir disso, foi definido o conceito de saliência de uma malha, que representa a medida de importância visual de determinadas regiões em um objeto 3D.

Este trabalho de Graduação está contextualizado na área de Computação Gráfica, o que faz com que as métricas de percepção visual sejam classificadas como essenciais. A partir da proposta de gerar imagens com um maior conteúdo visual, foi implementado neste trabalho o algoritmo de saliências em malhas 3D, criado por Lee em 2005, utilizando o seguinte sistema de processamento de malhas 3D: *Meshlab*. A partir dos resultados obtidos com o algoritmo, utilizamos conceitos básicos de renderização não foto-realística, como silhuetas, objetivando gerar imagens com um maior conteúdo perceptual e artístico. Desta forma, foi desenvolvido um algoritmo em NPR visando destacar informações que representem um maior conteúdo artístico a partir das regiões consideradas visualmente atraentes.

Palavras-chave: saliência em malhas 3D, percepção visual, NPR

Abstract

Recent advances that occurred in the last decade in the areas of computer graphics, multimedia and telecommunication, have influenced the evolution of the manipulation, the use and the transmission of digital data on the Internet. Three-dimensional data are currently the emerging media and in this context, 3D objects, or 3D meshes are used in various user-centered applications, which suggests the need to incorporate details of human visual perception in the processing of these meshes. Although excellent work of incorporating principles of perception related to images have been published, there has been little attention paid to the use of metrics, inspired by visual perception, for processing 3D meshes using only the information in it. The mesh, on it's own, contains geometric information capable of showing repeated patterns that even on high curvature regions are visually unattractive, because it's the unusual that delights and interests. Based on this description, the concept of mesh saliency was introduced as a measure of regional importance for three-dimensional meshes.

This graduate work is contextualized in the computer graphics, which classifies the metrics of visual perception as essential. We propose the generation of images containning more visual content, implementing the algorithm of mesh saliency, introduced by Lee in 2005, using the Meshlab system for processing 3D meshes. From the results obtained with the algorithm, we review some basic concepts of non-photorealistic rendering features, like silhouettes, aiming to create images with greater perceptual and artistic content. We developed an algorithm on NPR to represent images with a greater artistic content considering visually appealing regions.

Keywords: mesh saliency, visual perception, NPR

Sumário

Lista de Figuras

CAPÍTULO 1

Introdução

Avanços tecnológicos nos meios de telecomunicação, Computação Gráfica e multimídia durante a última década, contribuíram para a evolução dos dados digitais manipulados, visualizados e transmitidos através da internet. Nos dias de hoje, dados tridimensionais constituem o conteúdo multimídia emergente e, neste contexto, objetos 3D, ou malhas que representam os mesmos, estão sujeitos às diversas aplicações centradas visualmente nos usuários, o que sugere a necessidade de incorporar maiores detalhes da percepção humana no processamento destas malhas [Lee et al., 2005].

Apesar de já existirem excelentes trabalhos realizados que incorporam princípios de percepção em ambientes bidimensionais, houve pouca atenção dedicada ao uso de métricas, inspiradas por percepção visual, para o processamento de malhas que compõem objetos 3D, utilizando apenas informações existentes na própria estrutura da mesma. Existe também uma grande quantidade de estudos que utilizaram ferramentas para mapear os movimentos dos olhos do usuário (*Eye Tracker*)[Chung et al., 2004; Howlett et al., 2004] com intuito de melhor detectar as áreas de maior importância de uma malha 3D, mas os mesmos não permitem tanta liberdade ao usuário, impossibilitando-o de se movimentar durante o uso do aparelho. Lee et al. [2005] afirmam que a própria malha 3D, a partir de suas informações geométricas, é capaz de demonstrar a repetição de padrões. Os mesmos, quando encontrados, inclusive em regiões de alta curvatura são visualmente desinteressantes, pois o que desperta o interesse visual é o incomum e a partir disso, foi definido o conceito de saliência de uma malha, que representa a medida de importância visual de determinadas regiões em um objeto 3D.

1.1 Objetivo

A proposta deste trabalho é desenvolver e adequar o método de cálculo de saliências em malhas 3D desenvolvido por Lee et al. [2005] que utiliza um determinado mecanismo baseado nos vértices, que compõem a malha 3D do objeto, tratando-os como centros e considerando a sua vizinhança com o intuito de identificar regiões diferentes de suas respectivas redondezas, no contexto dos valores de curvatura média (por exemplo, uma região rugosa no meio de uma área plana). A partir do conteúdo destas informações, é proposta a exploração do mesmo no contexto de renderizações estilizadas, ou expressivas, que também são conhecidas por renderizações não foto-realísticas (*Non-Photorealistic Rendering* - NPR). O objetivo é unir estes dois conceitos dentro da Computação Gráfica, para explorar o potencial da informação contida na saliência, como elemento visual de importância local, no processo de renderização em NPR.

Apesar de inúmeros trabalhos na área utilizarem este conceito de saliências, como medida

de importância de uma região, para obter resultados com maior conteúdo perceptível [Liu et al., 2007; Barni et al., 2009; Feixas et al., 2009; Lavoué, 2009], nenhum deles explorou esta possibilidade. Desta forma, visamos explorar estas informações contidas na própria geometria da malha, que compõe o objeto 3D, com o intuito de melhor representar a imagem final, focando atenção especial a detalhes que possam caracterizar regiões com um maior interesse visual. Por esta razão, serão utilizadas técnicas de NPR que possuam características mais focadas em melhor representar áreas de percepção, visando assim, obter resultados mais artísticos e com maior capacidade de transmitir informações consideradas mais relevantes para o usuário final.

1.2 Estrutura do Documento

Este documento está estruturado da seguinte maneira: no Capítulo 2 serão descritos os avanços realizados na área de percepção visual e será realizada uma maior explicação dos conceitos de saliências visuais, saliências em malhas de objetos 3D (e o algoritmo de Lee et al. [2005]) e a importância da renderização não foto-realística (NPR) neste contexto. O Capítulo 3 terá o seu foco em descrever o desenvolvimento da nossa aplicação, detalhando cada parte do sistema. No Capítulo 4 serão demonstrados os resultados obtidos e realizadas comparações de informação perceptual das mesmas. Por fim, o Capítulo 5 apresentará algumas considerações finais, melhorias futuras para o algoritmo e para a aplicação, além de alguns passos a serem realizados para o futuro, mas que não fazem parte do escopo deste trabalho.

CAPÍTULO 2

Revisão Bibliográfica

No ramo da computação visual, extrair o máximo de características que descrevam uma imagem é considerado um dos problemas principais e, na maioria das vezes, essas imagens, sejam elas renderizações de uma cena 3D ou não, compõem aplicações centradas em usuários. Por esta razão, a incorporação de detalhes focados na percepção humana, ou nas falhas da mesma, no processamento dessas representações, pode ser uma ferramenta fundamental para a geração de imagens que passem o máximo de informação importante ao seu observador final [Kadir e Brady, 2001; Lee et al., 2005]. Porém, definir o que é significativo em uma imagem depende muito do contexto da mesma. Para estudar e compreender melhor este contexto, é necessário aprofundar-se um pouco mais na área de Computação Gráfica baseada em Percepção.

Na seção 2.1, damos maiores explicações sobre o contexto de como são utilizados critérios da percepção humana associados com Computação Gráfica. Na seção 2.2, explicamos os conceitos de saliência visual e da sua importância em imagens bidimensionais, estendendo esse conceito para o mundo tridimensional na seção 2.3. Já na seção 2.4 será explicitado o ramo artístico da renderização não foto-realística e a contextualização e importância da mesma neste projeto.

2.1 Computação Gráfica baseada em Percepção

Antes de falar deste ramo da Computação Gráfica, é preciso conhecer um pouco mais o funcionamento do Sistema de Visualização Humana, referido de agora em diante por SVH. O mesmo é capaz de resolver uma grande variedade de tarefas visuais com uma certa facilidade e confiabilidade, e esses encargos são realizados na parte frontal comum do sistema visual, composto pela retina e pela chamada visão de baixo nível [Kadir e Brady, 2001].

Um dos primeiros estudos realizados sobre a visão de baixo nível é atribuído a Neisser [Neisser, 1964 apud Kadir e Brady 2001, p. 83]. Nele, Neisser afirma que essa parte do SVH é composta por dois estágios: o pré-atento e o atento. O estágio pré-atento é aquele por qual possuímos maior interesse, pois nele somente as informações que se destacam (chamadas de *pop-out features*) são detectadas. Ou seja, regiões da imagem que contenham um certo tipo de descontinuidade serão descobertas neste estágio. Esse modelo visual definido por Neisser influencia bastante, até a atualidade, as comunidades de computação visual e reflete no modelo clássico de visão computacional - detecção de característica e agrupamento baseado em percepção, seguido por um modelo de correspondência.

De acordo com Chung et al. [2004], há muito tempo já se sabe que o SVH não consegue processar informação visual de uma maneira uniforme. O sistema de visualização dos prima-

tas, através do nível intermediário e alto de processamentos visuais, só permite a seleção de um subconjunto das possíveis informações sensoriais disponíveis antes de conseguir processar qualquer detalhamento mais específico na imagem [Tsotsos et al., 1995 apud Chung et al. 2004, p. 49]. A seleção desse subconjunto transforma-se em foco de atenção que realiza uma varredura em uma imagem, já que o SVH utiliza uma combinação de dados direcionados à imagem e modelos principais no seu processamento de dados visuais, de uma forma direcionada à saliência [Kadir e Brady, 2001; Itti et al., 1998]. Chung et al. afirmam que apesar desse foco de atenção poder ser realizado de forma consciente, ele também ocorre de forma inconsciente, atraindo a atenção para locais visualmente notáveis ou salientes. É amplamente difundido que a visão de baixo nível humana é, praticamente, independente de contexto o que a leva a ser atraída por partes da cena que destacam-se de alguma forma [Kadir e Brady, 2001].

A partir das explicações acima, podemos definir o principal foco do ramo da Computação Gráfica baseada em Percepção, pois a mesma visa explorar estas características do sistema de visão humana (SVH). O objetivo é determinar quais partes de uma imagem chamam maior atenção do usuário e, a partir disso, tentar melhorar ou destacar essas partes objetivando uma melhora de qualidade de percepção e, até mesmo, a representação em maiores detalhes [Howlett et al., 2004]. De acordo com O'Sullivan et al. [2004], a Computação Gráfica com base em percepção envolve a investigação de diversos problemas citados. Dentre estes, O'Sullivan et al. citam: a representação da realidade de maneira mais fiel no ramo da renderização foto-realística; uma melhor possibilidade de escolha entre velocidade e precisão de um algoritmo no ramo da renderização em tempo real; os tipos de anomalias que são mais notáveis e quando é possível fingir realidade e conseguir bons resultados; como melhor quantificar todos esses fatores e usá-los de forma metódica para adaptar as renderizações de acordo com a percepção do usuário que visualiza a imagem gerada. Para um melhor conhecimento e aprofundamento no assunto, O'Sullivan et al. incita que os pesquisadores da área gráfica devem estudar a literatura de experimentos realizados nos ramos da psicologia, neurofisiologia, psicofísica e outras áreas relacionadas ao sistema de percepção humano.

Estudos realizados de forma interdisciplinar envolvendo a área de Computação Gráfica e a área de percepção resultaram em novas idéias para ambas. A partir da necessidade dessas pessoas se encontrarem para discutirem o assunto em fóruns, a SIGGRAPH/EUROGRAPHICS realizou um encontro sobre Computação Gráfica baseada em Percepção [Howlett et al., 2004]. Desde então, muitos pesquisadores têm trabalhado ativamente neste campo e várias novas áreas de pesquisa vêm sendo trabalhadas.

Desde 2004, após o encontro realizado pela SIGGRAPH e EUROGRAPHICS, o crescimento do uso de informação de percepção para representar, analisar e renderizar malhas 3D só fez crescer e ganhar ainda mais importância. Ao longo deste capítulo serão discutidos alguns desses estudos realizados e, os mais pertinentes, serão descritos em maiores detalhes. Uma parcela desses trabalhos aqui detalhados, que também tiveram seu trabalho baseados na percepção visual, utilizaram um aparelho para realizar um mapeamento dos movimentos dos olhos, o *eye tracking*, e outras técnicas diferentes que tornaram possível o conhecimento de vários novos sistemas centrados nos usuários, que representam a estrutura de aplicações na área de Computação Gráfica [Barni et al., 2009].

2.2 Saliência Visual

De acordo com Kadir e Brady [2001], saliência visual é um termo que refere-se à idéia de que algumas partes da cena destacam-se de forma a chamar um tipo de atenção sem esforço aparente, determinado de pré-atenção, que causa um estímulo visual impactante que atinge o estágio de visão de baixo nível do SVH. Previamente, em 1995, Julesz [Julesz, 1995 apud Kadir e Brady 2001, p. 85] utilizava o termo *pop-out* para descrever estes processos de destaque visual que ocorrem no estágio pré-atento do SVH como nos exemplos encontrados na Figura 2.1 e que captam a atenção imediata. Dessa forma, é fácil entender porque a idéia de saliência já havia sido utilizada em diversos algoritmos da computação visual, embora de forma implícita [Kadir e Brady, 2001].

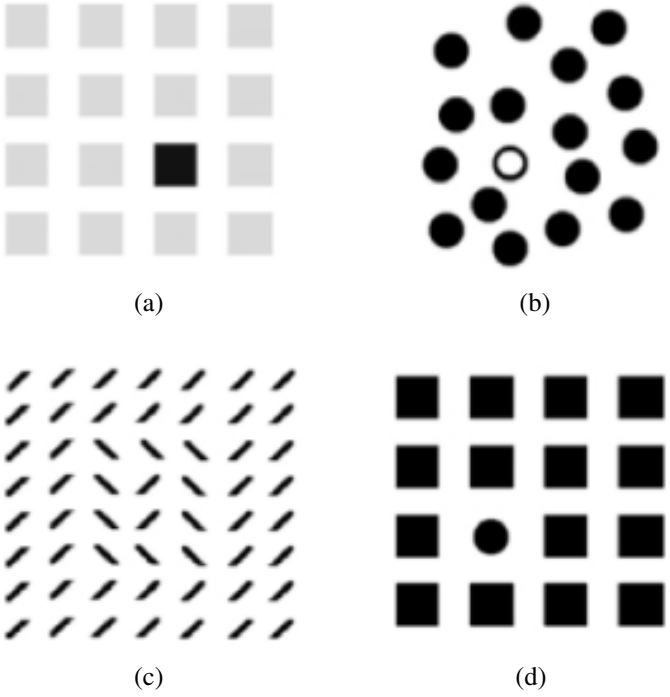


Figura 2.1: Figuras que demonstram algumas informações que captam a atenção imediata do sistema pré-atento - Saliência Visual. Nas figuras (a) e (b), podemos ver exemplo de destaques por coloração enquanto a forma é mantida. Já nas figuras (c) e (d), vemos destaques por forma enquanto a coloração é mantida. Imagem retirada de [Kadir e Brady, 2001].

Desde 1998, já havia trabalhos baseados em saliência. Em seu artigo, Itti et al. [1998] falam sobre um sistema baseado em atenção visual inspirado na arquitetura dos neurônios no Sistema Visual dos Primatas. De acordo com este trabalho, o sistema utiliza a combinação de imagens em diferentes escalas agrupando-as em um único mapa de saliência topográfico resultante. A idéia desse sistema é de quebrar o problema complexo do entendimento de uma cena selecionando, de forma rápida e computacionalmente eficaz, suas partes mais notáveis para futuras análises em um maior nível de detalhes. Antes da publicação deste trabalho, em

1998, utilizava-se bastante a expressão "atenção visual"ao tratar do assunto. Porém, Itti et al. utilizam o termo de atenção visual baseada em saliência dando, assim, início a um novo termo no ramo.

Em 2001, Kadir e Brady [2001] publicaram um artigo falando da importância de saliências e da escala na descrição das informações de uma imagem. Neste estudo, eles falam que naturalmente, saliência implica em raridade, mas que de acordo com Gilles [Gilles, 1998 apud Kadir e Brady 2001, p. 86], o oposto não é necessariamente verdadeiro, pois se todas as coisas fossem raras, então nada deveria ser marcado como saliente. O próprio Gilles aponta um outro problema que é comumente encontrado em sistemas de medição de saliências baseados, exclusivamente, em raridade, pois raridade depende intrinsecamente do método utilizado para a mensuração. Caso sejam usados descritores com um alto índice de discriminação, então tudo tende a ser destacado como raro. Por outro lado, caso os índices de discriminação sejam muito gerais, nada tende a ser marcado como raridade. Encontrar o nível apropriado para discernir os descritores é uma tarefa bastante difícil. A Figura 2.2 mostra o resultado da utilização de um índice de discernimento global. Ou seja, não há diferenciação de conteúdo da imagem, apenas há a diferenciação baseada em cores e nos seus arredores.

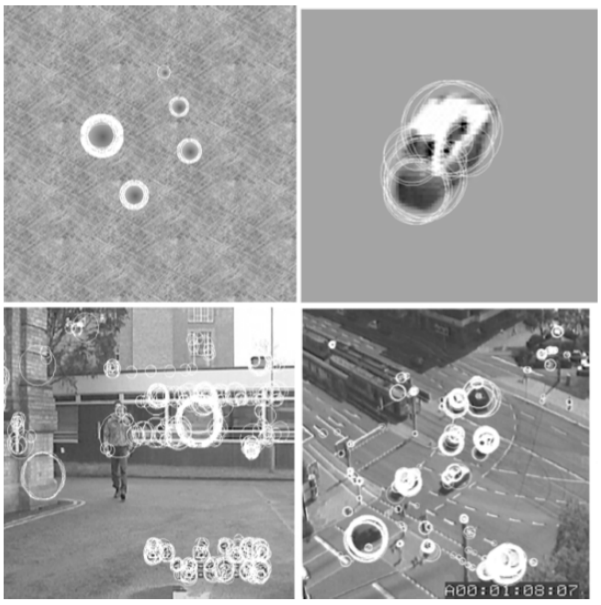


Figura 2.2: Um dos algoritmos de cálculo de pontos salientes em uma imagem. Nesta, podemos ver os pontos de maior saliência e as suas diferentes escalas. Neste caso, foi utilizado um fator de discernimento global. Imagem retirada de [Kadir e Brady, 2001].

No ano de 2004, ano de realização do encontro organizado pela SIGGRAPH e EUROGRAPHICS, surgiram outros trabalhos relacionados à saliência, uma boa parte deles pode ser encontrada nas referências em [O'Sullivan et al., 2004]. Além desses, houveram trabalhos que destacaram-se com a utilização de um *eye tracker* (Figura 2.3), que consiste em detectar movimentos realizados pelo olho, associados à uma busca visual, de uma forma precisa utilizando informações de posição relativa a partir da reflexão de um raio infra-vermelho sobre a pupila

e a córnea. Levando em consideração que o tempo necessário para que o cérebro considere informações visuais, é necessário que o olho permaneça na mesma posição por mais de 100 ms. O uso desse aparelho serve para realizar um mapeamento visual que é então utilizado para identificar características salientes em uma cena [Chung et al., 2004].

Um dos trabalhos que destacou-se pelo uso do aparelho descrito, foi o publicado por Howlett et al. [2004], onde foi abordado o problema de caracterizar a existência de saliência para objetos 3D e a garantia de que, caso essas características existam, elas podem ser encontradas previamente. Neste trabalho, foram realizadas algumas alterações do nível de detalhes de um objeto 3D, através de um algoritmo, baseado na versão original do *QSlim* [Garland e Heckbert, 1997], de simplificação desenvolvido com a utilização de informações destacadas como salientes. A partir dos resultados obtidos, foram realizados experimentos comparativos com usuários para verificar se as áreas detectadas como salientes em um maior nível de detalhes, ainda mantinham-se salientes no objeto após a sua simplificação. Os resultados mostraram que para objetos não manufaturados, as características salientes foram mantidas, facilitando o trabalho de reconhecê-los.

No entanto, esses aparelhos de mapeamento visual ainda são muito caros e não proporcionam conforto para o usuário. Além de necessitarem de um incômodo procedimento de calibragem, impossibilitam que a pessoa possa movimentar-se ao utilizá-lo. Por estas razões, Lee et al. [2009] explicam que o uso dos princípios estudados em Percepção do SVH devem ser aplicados para estimar computacionalmente a região onde os usuários podem estar olhando ao invés de utilizar-se de um sistema tão incômodo.

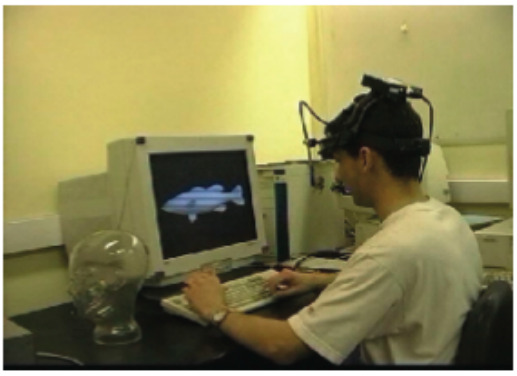
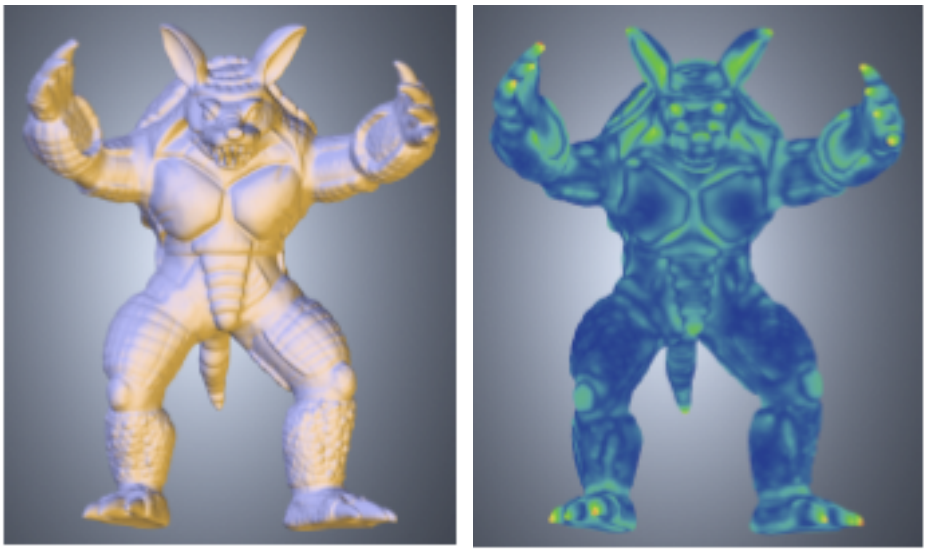


Figura 2.3: O dispositivo de *Eye Tracking* da *SMI EyeLink*. Imagem retirada de [Howlett et al., 2004].

2.3 Saliência em Malhas 3D

Alguns anos antes de Sungkil Lee [2009] falar sobre os empecilhos causados pela necessidade do uso de um aparelho de *eye tracking*, Chang Ha Lee [2005] havia discutido e proposto uma abordagem diferente. No artigo publicado em 2005, foi apresentado um algoritmo que incorpora explicitamente modelos baseados na visão de baixo nível do SVH. Nele, é associada, pela

primeira vez, a idéia de saliência como uma medida de importância de uma determinada região, em uma malha que representa um objeto 3D. Lee et al. [2005] desenvolveram um algoritmo explicitamente baseado no modelo de Itti et al. [1998]. Um algoritmo que realiza o cálculo de forma independente de escala, utilizando-se de um operador de centro-vizinhança sobre a média Gaussiana das curvaturas médias de cada um dos vértices da vizinhança, para calcular a saliência da mesma. Dessa forma, eles conseguiram expandir a idéia de saliência que estava associada a imagens 2D e fizeram uma associação direta para objetos no ambiente 3D, fazendo com que a saliência do objeto só precisasse ser calculada uma única vez para um determinado objeto e não para cada cena a ser redesenhada em uma aplicação. Um exemplo clássico retirado do artigo citado acima está na Figura 2.4.



(a) *Stanford Armadillo* (b) Mapa de Saliência do *Stanford Armadillo*

Figura 2.4: Modelo 3D do *Stanford Armadillo* (a) e o seu Mapa de Saliência à direita (b). Cores mais quentes (vermelhos e amarelos) representam alta saliência e cores mais frias (verdes e azuis) baixa saliência. Retirada de [Lee et al., 2005].

.

2.3.1 Algoritmo de Saliências de Lee

Para desenvolver o seu algoritmo, Lee et al. [2005] partiram do princípio de que é o diferente e inesperado que capta o interesse das pessoas e, em seu trabalho, utilizaram o mesmo conceito de raridade descrito por Gilles [1998], adaptando a idéia para medição de raridade local. Sabe-se que medidas puramente geométricas que representam formas, tal como a curvatura, têm um histórico rico de uso na literatura de processamento de malhas. Porém, uma métrica baseada puramente na curvatura não necessariamente será uma boa métrica de importância relacionada à percepção, pois não necessariamente vai destacar-se dentro do seu contexto. Se considerarmos um pico com alta curvatura no meio de uma esfera, é claro que este espinho deve ser classificado como algo importante quando considerando a sua percepção, como pode ser visto na Figura 2.5.

Da mesma forma, uma região lisa no meio de uma região com uma alta densidade de morros ou picos de alta curvatura, também deve ser classificada como uma área de grande importância de percepção. Por essa razão, antes de classificar se uma região destaca-se ou não em um objeto 3D, precisa-se saber mais informações sobre os seus arredores antes de classificar o seu grau de saliência.

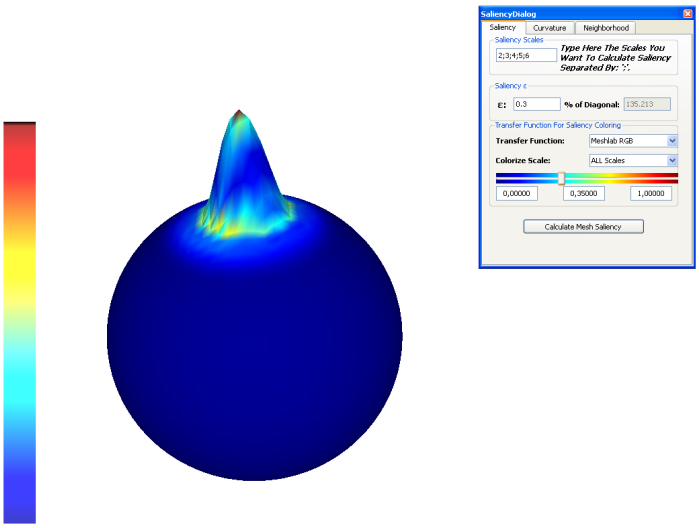
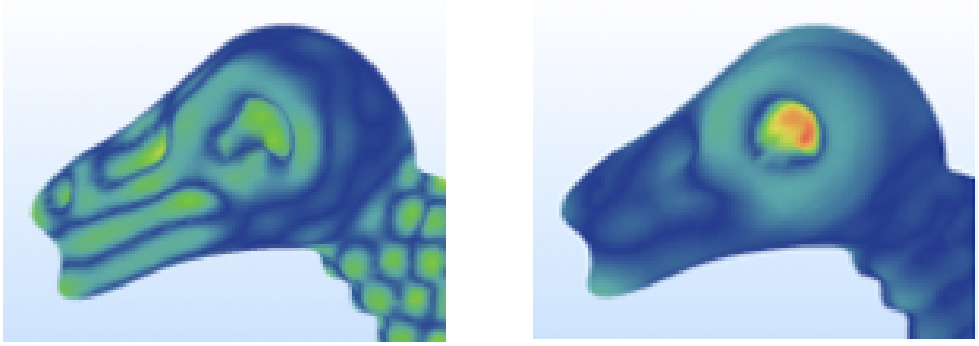


Figura 2.5: Saliência, de um pico, em uma esfera. A escala de cores pode ser vista à esquerda da imagem. Baixa saliência em azul e alta em vermelho. Imagem gerada utilizando a aplicação descrita neste documento.

Ao introduzir o conceito de saliência, espera-se que um bom modelo para calculá-la seja capaz de atuar em diferentes escalas de visibilidade de um objeto, pois nem tudo que é interessante em uma escala, permanece interessante em outra. Uma melhor demonstração disso pode ser vista na Figura 2.6. Tendo em mente também essa percepção da necessidade de múltiplas escalas significativas para o mapa de saliência em uma malha 3D, Lee et al. utilizaram um modelo de cálculo de saliências que leva tais proporções em consideração e a forma como isto será realizado será descrita em mais detalhes a seguir.

Pelo fato do algoritmo de Itti et al. [1998] possuir uma das técnicas mais efetivas para computação de saliência para imagens em 2 dimensões, Lee et al. optaram por utilizar um mecanismo de centro-vizinhança semelhante ao citado em 1998 adaptado para objetos 3D. Diferentemente das imagens, onde a cor é considerada o atributo mais importante, em malhas de objetos 3D a geometria é quem representa este papel. Apesar de não só a geometria ser considerada importante, neste algoritmo só foi incorporado o uso de informações geométricas, mas ele pode ser, facilmente, adaptado para incorporar outros tipos de atributos aparentes em uma superfície.

Em imagens, uma região com intensidade - função entre forma e iluminação - uniforme equivale a uma área de saliência zero, de acordo com o método de Itti et al. que utiliza as



(a) Escala Menor (b) Escala Maior

Figura 2.6: Saliência na imagem (a), em menor escala, destaca informações menores como o nariz, a boca e algumas partes do pescoço e na imagem (b), identifica uma informação maior: o olho, por exemplo. Imagens retiradas de [Lee et al., 2005].

variações como motivação principal no cálculo da mesma. No caso da saliência para objetos 3D, a aferição é realizada baseada apenas na forma, independente de iluminação. Logo, a esfera é um exemplo de zero canônico para saliências em malhas 3D, pois os seus valores de curvatura são invariantes e ao utilizar o mecanismo de centro-vizinhança, todas as regiões possuirão valores de saliência iguais. A Figura 2.7 mostra a saliência calculada, por nossa aplicação, em uma esfera.

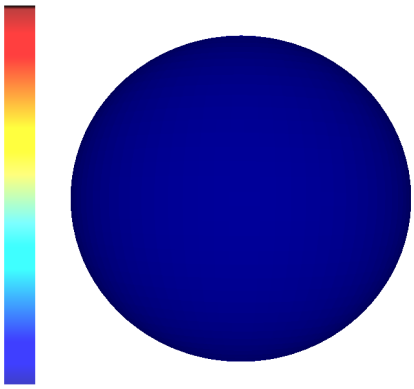


Figura 2.7: Saliência calculada em uma esfera e escala de cores utilizada à esquerda. Azul representa áreas de saliência baixa ou nula.

A idéia principal do algoritmo do cálculo de saliências em malhas de objetos 3D é filtrar as curvaturas dos vértices que a compõem, utilizando um operador de centro-vizinhança sobre a média Gaussiana das curvaturas médias. Dessa forma, é possível obter uma certa noção de importância em uma determinada região. Na Figura 2.8, pode ser visto em imagens, o processo que será descrito a seguir. O algoritmo de Lee et al. [2005] pode ser sumarizado nos cinco

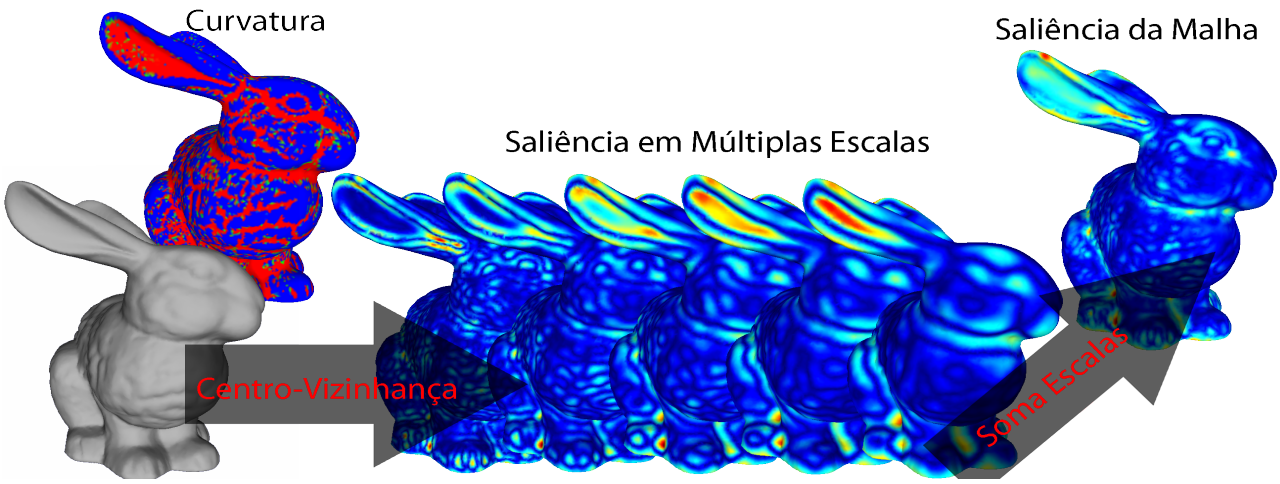


Figura 2.8: Visão geral do cálculo da saliência exemplificada no modelo do *Bunny*, com 50.000 polígonos. Primeiro é calculada a curvatura média nos vértices da malha. Para cada vértice, é computada a saliência como a diferença entre as curvaturas médias com um filtro Gaussiano menor e outro maior e utilizando um mecanismo de centro-vizinhança para combinar as informações encontradas, em diferentes escalas, na curvatura. Imagem gerada por nossa aplicação.

passos a seguir:

1. Inicialmente, é preciso calcular a curvatura média para cada um dos vértices $\mathbf{v} \in \boldsymbol{M}$, onde $\boldsymbol{M}$ representa a malha que compõe o objeto 3D. Usaremos $\mathscr{C}(\mathbf{v})$ para denotar a curvatura média do vértice $\mathbf{v}$. Para esta etapa, foi selecionado o algoritmo para cálculo de curvaturas principais em um vértice, de Taubin [1995] e, após encontrar as duas curvaturas principais, foi realizada a média aritmética dos valores para obter-se o valor da curvatura média no vértice $\mathbf{v}$.

2. Compute a média Gaussiana da curvatura média para cada um dos vértices $\mathbf{v}$ utilizando a equação abaixo:

$$G(\mathscr{C}(\mathbf{v}), \sigma) = \frac{\sum\limits_{\mathbf{x} \in N(\boldsymbol{v},\, 2\sigma)} \mathscr{C}(\mathbf{x}) \exp\left[\dfrac{-||\mathbf{x}-\mathbf{v}||^2}{2\sigma^2}\right]}{\sum\limits_{\mathbf{x} \in N(\boldsymbol{v},\, 2\sigma)} \exp\left[\dfrac{-||\mathbf{x}-\mathbf{v}||^2}{2\sigma^2}\right]}, \tag{2.1}$$

onde a vizinhança de um vértice, indicada por $N(\mathbf{v}, \sigma)$, denota o conjunto de outros vértices que estão dentro dos limites de 2σ. No algoritmo de Lee et al., foi utilizada a distância Euclidiana. É válido ressaltar que na fórmula (2.1) o valor 2σ foi utilizado como um limiar para o filtro Gaussiano.

3. Para calcular a saliência em um vértice **v**, é necessário avaliar a diferença absoluta entre as médias Gaussianas computadas em maior e menor escala utilizando a seguinte fórmula:

$$\mathscr{S}(\mathbf{v}) = |G(\mathscr{C}(\mathbf{v}), \sigma) - G(\mathscr{C}(\mathbf{v}), 2\sigma)| \tag{2.2}$$

   Assim como no algoritmo de Lee et al., foi utilizado um desvio-padrão, para a escala com maior qualidade, de duas vezes a escala de qualidade inferior.

4. Para o cálculo de saliências em múltiplas escalas, é preciso fazer uma pequena modificação na equação (2.2). Para suportar o cálculo para cada uma das diferentes escalas, a equação é alterada para:

$$\mathscr{S}_i(\mathbf{v}) = |G(\mathscr{C}(\mathbf{v}), \sigma_i) - G(\mathscr{C}(\mathbf{v}), 2\sigma_i)|, \tag{2.3}$$

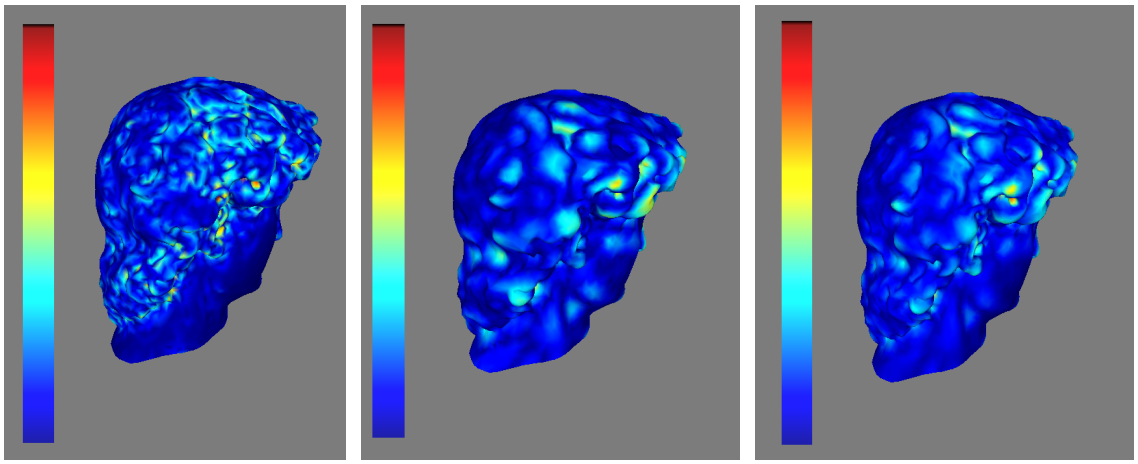
   onde σ_i é o desvio-padrão básico do filtro Gaussiano na escala *i*.

   No artigo de Lee et al. [2005], todos os mapas de saliência foram calculados usando cinco escalas $\sigma_i \in \{2\varepsilon, 3\varepsilon, 4\varepsilon, 5\varepsilon, 6\varepsilon\}$. O valor de ε representa 0.3% do comprimento da diagonal principal da *bounding box* do modelo 3D.

5. A última etapa para o cálculo final da saliência em uma malha 3D é a forma de combinar os $\mathscr{S}_i$ mapas de saliência em diferentes escalas. Para a realização deste cálculo é aplicado um operador S de supressão não-linear, similar ao que foi proposto por Itti et al. [1998], para cada um dos mapas já calculados. Com esse operador de supressão, há uma valorização global dos mapas com um baixo número de altos picos (Figura 2.9 (b)) ao mesmo tempo em que há a suavização global dos mapas com um alto número de picos similares (Figura 2.9 (a)).
   Essa supressão ajuda na redução do número de pontos classificados como salientes (Figura 2.9 (c)), caso não fosse usada teríamos uma quantidade muito alta de pontos salientes e isso não reflete a unicidade que é o que torna uma área saliente. Este operador pode ser descrito da seguinte forma:

   Para cada um dos mapas de saliência $\mathscr{S}_i$, os valores são normalizados para valores no intervalo [0, 1]. Depois, é necessário encontrar, em cada escala, o vértice que possui a saliência máxima, de valor $\mathbf{M}_i$, onde *i* representa a escala. Com o vértice de saliência igual a $\mathbf{M}_i$ encontrado, precisa ser calculado o valor de $\overline{m}_i$ como a média aritmética no domínio da vizinhança do mesmo, excluindo o valor da saliência máxima naquela escala. Para cada uma das escalas, após encontrados os valores citados, é necessário calcular o valor do fator $(M_i - \overline{m}_i)^2$ e multiplicá-lo pelo mapa de Saliência $\mathscr{S}_i$ naquela escala. No final, o mapa de saliência nas múltiplas escalas *i* é calculado através do somatório das mesmas após o filtro de supressão: $\mathscr{S} = \sum_i S(\mathscr{S}_i)$.

Dentro do seu artigo, Lee et al. [2005] utilizam a sua técnica de saliência, descrita acima, como um operador para obter melhores resultados no algoritmo de simplificação de malhas.



(a) Escala 2ε (b) Escala 6ε (c) Múltiplas Escalas

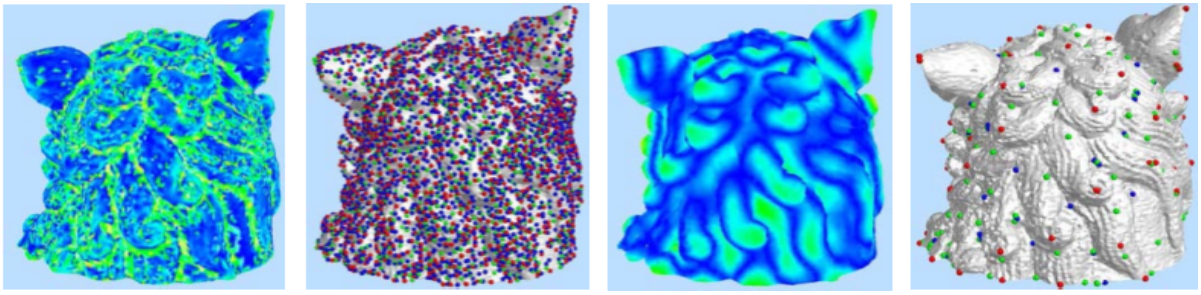
Figura 2.9: Mapa de Saliência na escala 2ε (a), mostra altos picos similares marcados em vermelho. Já na escala 6ε (b), podemos visualizar um baixo número de altos picos. A imagem (c), mostra o resultado do mapa em múltiplas escalas, após a realização do filtro de supressão, beneficiando os baixos picos da saliência em (b) e suprimindo os altos picos repetitivos vistos em (a). Imagem gerada pela nossa aplicação.

Eles fizeram um estudo comparativo do seu método com o método baseado em quádricas - o QSlim desenvolvido por Garland e Heckbert [1997]. No mesmo artigo, uma outra utilização foi dada ao algoritmo com o intuito de melhorar os resultados obtidos na seleção automática de *viewpoints*. Após este passo dado por Lee et al., diversos estudos foram realizados utilizando estes conceitos e alguns deles serão comentados a seguir.

2.3.2 Evolução das Saliências 3D

Após esta nova abordagem sobre saliências realizada por Lee et al. [2005] surgiram diversos outros trabalhos utilizando a saliência em malhas de objetos 3D. No ano de 2007, Liu et al. [2007] publicaram um artigo com um novo método para extrair Pontos Críticos Salientes em uma malha combinando o trabalho de saliências de Lee com a teoria de Morse. De acordo com Liu et al., a utilização de saliência, ao invés de aplicar medidas puramente geométricas de forma, como a curvatura, produz resultados mais satisfatórios com um número menor de pontos críticos. Uma boa comparação dos resultados pode ser vista na Figura 2.10.

Atualmente, o assunto de saliências, seja ela calculada em imagens ou nos próprios objetos 3D, ainda tem bastante força. Neste ano de 2009, houve mais três artigos publicados que utilizaram saliência como um atributo perceptivo com intuito de explorar o SVH. No artigo de Li et al. [2009], são utilizados tanto a saliência de um vértice quanto a sua curvatura principal para desenvolver um novo algoritmo de remoção de ruídos e de suavização de malhas 3D. Já Barni et al. [2009] utilizam a saliência de malhas para auxiliar na resolução do problema de segmentação de malhas e renderização dependente de um *viewpoint*. No seu artigo, a saliência é incorporada de forma propagativa no agrupamento de partes da malha guiando a formação



(a) Curvatura Média (b) Pontos Críticos na Curvatura (c) Saliência na Malha (d) Pontos Críticos na Saliência

Figura 2.10: Pontos Críticos Salientes (pontos azuis, vermelhos e verdes representam pontos de mínimo, máximo e de sela, respectivamente). A imagem (a), mostra a parte de trás da cabeça do leão e sua curvatura média correspondente. Na imagem (b), vemos os 7,629 pontos críticos encontrados devido à sensibilidade a ruídos da curvatura. Na imagem (c), a saliência da malha do objeto 3D correspondente. Na imagem (d), os pontos críticos em menor quantidade. Liu et al. afirmam que a saliência (c) captura a textura do cabelo do leão e nega o ruído da curvatura em (a) e, com isso, seleciona os pontos críticos mais interessantes em uma região de grande importância [Liu et al., 2007].

desses grupos de triângulos. Feixas et al. [2009], por sua vez, focaram o trabalho na seleção de *viewpoint* por considerá-la uma área crescente no ramo da Computação Gráfica. A seleção de *viewpoints* e a simplificação de malhas 3D já haviam sido exploradas por Lee et al. [2005] em seu artigo, mas Feixas utiliza uma abordagem diferente utilizando saliência como um fator de importância para demonstrar como critérios de percepção poderiam ser inseridos no seu método desenvolvido.

Diversas outras abordagens envolvendo saliências em cenas e em objetos 3D já foram realizadas, mas não foram muito detalhadas neste trabalho por não terem forte influência direta sobre o mesmo. Dentre estas abordagens, podemos exemplificar o trabalho de Longhurst et al. [2006] que utilizou o poder das placas gráficas (*Graphics Processing Unit - GPU*) e desenvolveu uma forma de calcular um mapa de saliências agregado com mapas de profundidade e de movimento, em tempo real, para o seu uso em renderização seletiva. Já Lee et al. [2009] utilizaram o seu trabalho realizado em 2007 que realizava o cálculo de mapas de saliência em tempo real, também com o uso de GPU, adaptando-o para gerenciar o nível de detalhes de alguns ambientes virtuais. Zhihong et al. [2009] utilizaram uma abordagem diferente para o cálculo de saliências: as *Crest Lines* que já são bastante utilizadas em análise de imagens, reconhecimento facial, análise e registro de estruturas anatômicas, renderização não foto-realística, etc. Essas abordagens utilizadas são conhecidas e fizeram parte da pesquisa, mas não tem influência direta no que foi desenvolvido neste trabalho.

2.4 Renderização Não Foto-realística

No ramo da Computação Gráfica tradicional, desde os primórdios da mesma, há um grande foco e busca pelo realismo. Esta área é conhecida como Renderização Foto-realística e este termo representa uma área que mostra técnicas e formas artísticas que tentam representar ou criar imagens sintéticas, que possuem tamanha semelhança com fotografias reais, que são comumente confundidas [Green et al., 1999]. Por exemplo, na Figura 2.11 da Ponte de Londres, é possível confundir-se e acreditar que a mesma é uma fotografia tirada no local e não uma imagem gerada por computador.

Figura 2.11: Imagem da Ponte de Londres renderizada utilizando foto-realismo. Imagem Cortesia de Luciano Neves no site da Autodesk [aut, 2009].

Em oposição à técnica de foto-realismo, foi criado o ramo da Renderização Não Foto-realística (*Non-Photorealistic Rendering - NPR*) que tem o seu foco no processamento de imagens ou vídeos que visam **reproduzir** estilos de artistas e ilustradores. Explicita-se reproduzir, pois as técnicas de NPR têm como finalidade renderizar imagens no intuito de simular efeitos já criados e aprimorados durante anos por artistas. Por tratar-se de um ramo que propõe representar diferentes estilos, o ramo da NPR é também conhecido como renderização estilizada ou expressiva. Mesmo assim, pode-se dizer que há mais espaço para o crescimento de NPR do que para o foto-realismo, pois o mesmo já é estritamente bem definido, enquanto o meio não foto-realístico depende da criatividade e de técnicas já utilizadas ou que ainda venham a surgir [Britto Neto e Carvalho, 2008; Halper et al., 2002; Viola et al., 2006].

Uma das vantagens da NPR é a sua capacidade de informação extra-semântica. Por exemplo, caso um arquiteto deseje mostrar um espaço e explicitar a forma na qual as pessoas se encaixam no mesmo, ele pode demonstrar essas informações através de uma imagem em escala de cinza e utilizar o branco para demonstrar as dimensões do ambiente. Caso ele utilize uma imagem mais realista, algumas pessoas podem se prender a detalhes como as cores das paredes, mobílias e objetos e nesse caso, não é o intuito final do artista definir esse tipo de

informação neste estágio. No exemplo visto na Figura 2.12, foi utilizado NPR para representar uma casa de dois andares e um deles foi inclinado para melhor visualização do andar de baixo [Hagedorn e Döllner, 2007].

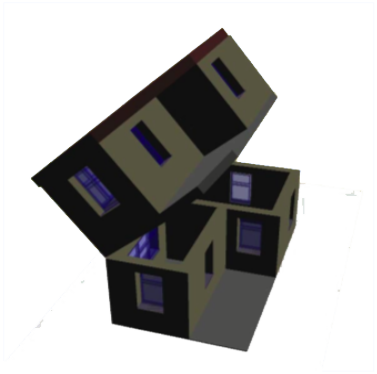


Figura 2.12: Na imagem, um dos andares é inclinado para que seja possível visualizar o interior do andar no térreo. Como a renderização possui uma característica não foto-realística, o observador não está preocupado com a inclinação do andar de cima ser ou não possível. Imagem retirada de Hagedorn e Döllner [2007].

Alguns dos estilos, bastante difundidos, representados por NPR que podemos citar são: pintura que, como o próprio nome já diz, visam reproduzir o efeito das pinceladas que compõem um pintura como a da Figura 2.13; esboço, que visa representar alguns contornos e apenas os traços mais importantes e que definam as informações desejadas em um determinado momento, objetivando uma imagem mais limpa, como a vista na Figura 2.14; *stippling*, que é a representação de uma imagem através de um conjunto de pequenos pontos para representar tons como o sapo visto na Figura 2.15; *cartoon*, diferentes formas de arte e ilustração que tem como finalidade a representação de um desenho de forma característica vista em desenhos, como o mago visto na Figura 2.16 [Haller e Sperl, 2004; Lewis et al., 2005; Cherlin et al., 2005; Kim et al., 2009].

2.4.1 Representação de Malhas Através de Linhas

Dentro do ramo da NPR, a representação por linhas é uma das formas mais simples e eficientes de se renderizar uma malha que compõe um objeto 3D. Essa forma de exibição é completamente dependente do posicionamento da câmera e dos objetos a serem representados na cena. Por essa razão, é necessário ser redesenhada a cada mudança ocorrida em qualquer um dos dois. Porém, a importância gráfica dos resultados obtidos e o poder da informação contida na mesma são bastante utilizados como componentes para esquemas mais complexos de renderização [Pop et al., 2001; Jaimes, 2006; Northrup e Markosian, 2000].

Dentro deste contexto, podemos dividir esta representação por linhas em três tipos: Silhuetas ou contornos que são compostos pelas curvas que separam uma face que está virada para

Figura 2.13: Nas imagens acima, podemos visualizar a obra **Quarto em Arles** do pintor impressionista Vincent van Gogh renderizado em diferentes tipos de pinceladas. Retirado de Haller e Sperl [2004].

Figura 2.14: Podemos visualizar o objeto *Knot* renderizado utilizando o efeito de esboço. A figura foi gerada utilizando quatro diferentes níveis de segmentação. Imagem retirada de Lewis et al. [2005].

frente da câmera (*front-facing*) de uma face que está virada para o lado não visível da cena (*back-facing*); Dobras, rugas ou *creases* que são definidos em uma própria malha como traços acentuados da mesma e podem ser vistos no exemplo mostrado na Figura 2.17; Contornos sugestivos que representam um conceito criado por Rusinkiewicz et al. em 2003 e visa tentar informar lugares que representariam silhuetas em posições próximas à posição atual da câmera. A partir de informações com tamanho poder descritivo, é possível tentar recriar, ou criar, efeitos artísticos que valorizem alguns aspectos visuais em uma imagem tentando aproximar-se um pouco mais dos trabalhos, feitos à mão, por artistas [Rusinkiewicz et al., 2008].

As linhas do contorno de um objeto 3D são características que facilitam muito a sua visão quando o mesmo encontra-se representado de forma bidimensional após a sua renderização. Essa é uma das principais razões que tornam a silhueta uma característica de extrema importância na representação de malhas de objetos. Outra das razões é o potencial que o contorno possui ao representar imagens com aparência de que foram desenhadas à mão. Artistas e ilus-

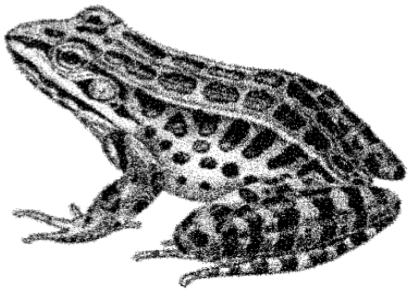


Figura 2.15: Gerada utilizando o algoritmo de *stippling* que representa uma imagem através de um conjunto de pequenos pontos que definem o tom da figura. Imagem retirada de Kim et al. [2009].

Figura 2.16: Imagem que apresenta um desenho em *cartoon* de um mago. Imagem retirada de Cherlin et al. [2005].

tradores utilizam bastante silhuetas e normalmente o fazem desenhando-as explicitamente ou através de um reforço no contraste das bordas da mesma como pode ser visto na Figura 2.18. Além disso, métodos bastante notórios no ramo de NPR, como o *cell shading* ou, *toon shading* como também é conhecida, utilizam o cálculo de bordas apenas como parte do algoritmo da técnica [Brosz et al., 2004; Kalnins et al., 2003].

Para calcular a silhueta de um objeto 3D é preciso primeiro entender o que define a mesma neste contexto. Para definir se um ponto pertence à silhueta de um determinado objeto 3D, é preciso encontrar o vetor $\vec{E}$ que representa o vetor do olho em direção ao ponto e o vetor $\vec{N}$ normal ao ponto. Um ponto é determinado ponto de silhueta se o ângulo formado entre estes dois vetores for de 90^o, ou seja, $\vec{E} \cdot \vec{N} = 0$. A Figura 2.19 representa bem essa descrição [Gooch e Gooch, 1999].

2.5 Concepção

A partir do estudo destes dois importantes conceitos, Saliência em Malhas 3D e NPR, deu-se a concepção deste trabalho de graduação com o intuito de beneficiar a área de NPR com

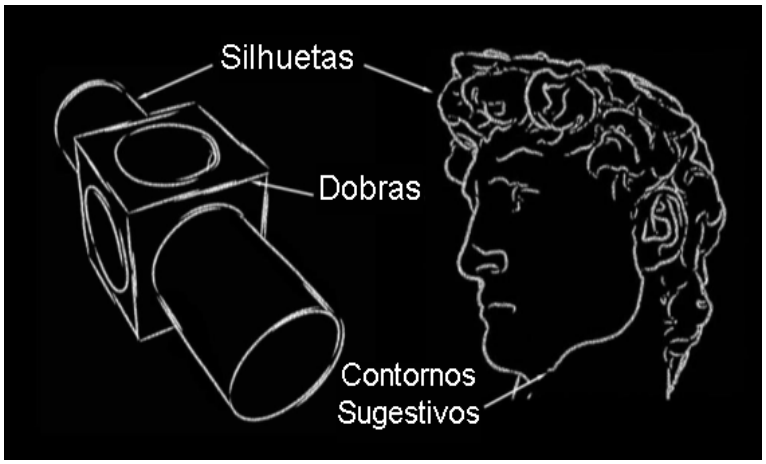


Figura 2.17: Explicita os três tipos de linhas que compõem determinada representação: silhuetas, dobras e contornos sugestivos. Imagem retirada de Rusinkiewicz et al. [2008].

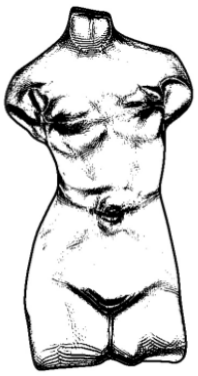


Figura 2.18: Modelo 3D da Venus renderizada com um estilo que simula uma peça de madeira talhada utilizando contorno. Imagem retirada de Hertzmann [1999].

as informações contidas na saliência. Até o presente momento, nada que unisse esses dois temas havia sido realizado. A força e o potencial da saliência em malhas de objetos 3D aliado com o poder descritivo de técnicas de NPR, tal como as silhuetas, tem capacidade de resultar em imagens ainda mais descritivas e com realce mais forte nas características consideradas importantes, pelo usuário final, unido com o destaque das características que o artista considera importante para determinadas tarefas. Além disso, o potencial uso de saliências em objetos 3D ainda pode ser aproveitado, após a conclusão deste estudo, para diversos outros fins, sendo estes relacionados à NPR ou não.

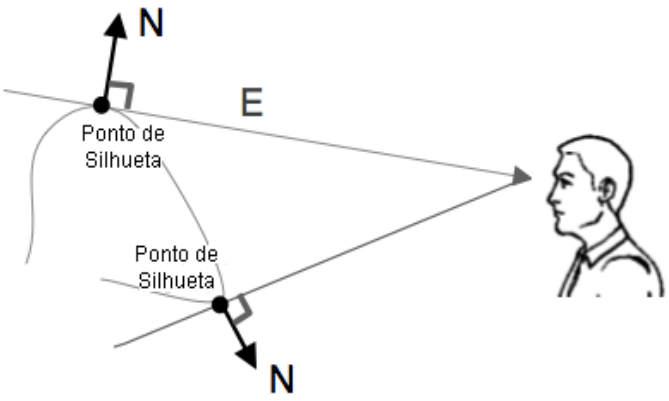


Figura 2.19: Caso o resultado da equação $\vec{E} \cdot \vec{N}$ seja igual a 0, ou seja, o ângulo formado entre os vetores $\vec{E}$ e $\vec{N}$ seja de 90^o, o ponto é considerado um ponto de silhueta. Imagem retirada de Gooch e Gooch [1999].

CAPÍTULO 3

Saliências em Malhas 3D Aplicado em NPR

Para o desenvolvimento do projeto, foram estudadas algumas bibliotecas e *frameworks* já disponíveis e de código aberto, para a realização de tarefas de manipulação em malhas 3D. A que mais se encaixou com o perfil do mesmo foi o *Meshlab* [mes, 2009]. A principal razão para tal escolha se deu pelo fato do grupo de desenvolvedores da mesma ser ativo e pelo grande suporte de formatos de carregamento de arquivos com diferentes extensões. Na seção 3.1, detalharemos melhor o sistema utilizado para o desenvolvimento do projeto e de que forma o mesmo auxiliou no desenvolvimento da aplicação. Na seção 3.2 pode ser vista uma visão geral do sistema e as suas interfaces, seguida de uma explicação mais detalhada do algoritmo implementado e das alterações realizadas no mesmo.

3.1 Meshlab

O ***Meshlab*** é um sistema avançado de processamento de malhas 3D e conta com ferramentas de uso automático e assistidas pelo usuário na edição, limpeza, conversão e renderização de malhas de objetos 3D grandes e desestruturadas. Uma das facilidades do sistema é a sua portabilidade o que permite que funcione em Sistemas Operacionais como: *Windows*, Linux (a partir do código fonte) e Mac OS X (somente Intel). Este sistema possui o código aberto e bastante extensível, e foi criado com intuito de auxiliar o processamento de estruturas, não tão pequenas, de objetos 3D e que, por ventura, podem ter sido decorrentes de digitalização 3D, providenciando um vasto conjunto de ferramentas. Atualmente, o sistema dá suporte aos seguintes formatos de entrada e saída: PLY, STL, OFF, OBJ, 3DS, VRML 2.0, X3D e COLLADA [mes, 2009].

O sistema começou a ser desenvolvido no final de 2005, como um projeto de um sistema de propósito geral, como parte do curso de Ciência da Computação da Universidade de Pisa: curso de Fundamentos de Gráficos Tridimensionais (*Fondamenti di Grafica Tridimensionale - FGT*) e a maioria do código do mesmo foi desenvolvida por estudantes da própria Universidade. Desde então, diversos outros estudantes vêm dando continuidade ao projeto e desenvolvendo cada vez mais recursos. Além deles, o projeto tem suporte ativo da comunidade ***3D-CoForm project*** e ainda é ativamente desenvolvido por um pequeno grupo de pessoas do Laboratório de Computação Visual do Instituto Italiano (ISTI), por um grande grupo de estudantes universitários, e por grandes desenvolvedores mundo afora. Uma amostra do resultado desse trabalho em conjunto, é o lançamento da versão 1.2.2 que ocorreu neste ano, mais precisamente no dia 09 de Setembro [mes, 2009].

3.1.1 *VCG Library*

Ao falar do ***Meshlab***, é importante destacar a base sobre a qual ele foi construído e que foi usada para todas as tarefas de processamento de malhas, que compõem objetos 3D, neste projeto. A ***VCG Library*** é uma biblioteca portável baseada em *templates* desenvolvida em C++, visando a manipulação e o processamento de malhas de objetos 3D compostas tanto por triângulos quanto por tetraedros. Essa é a definição da biblioteca VCG dada pela equipe de desenvolvimento que trabalhou na mesma no Laboratório de Computação Visual (*Visual Computing Lab - VCL*) do Instituto Italiano (ISTI) do Conselho Nacional de Pesquisa Italiana (Italian National Research Council - CNR) [vcl, 2009].

Esta biblioteca, que contém mais de 50.000 linhas de código, inclui diversos algoritmos para a tarefa de processar malhas, dentre eles podem ser vistos: simplificação por alta qualidade de quádricas, limpeza de uma malha 3D, suavização de malhas e remoção de ruídos, cálculo de curvatura, dentre outros diversos algoritmos encontrados no núcleo da biblioteca. O VCL, grupo desenvolvedor da mesma, a utiliza como base para as suas ferramentas desenvolvidas em *software*. Os grandes exemplos citados por eles são: *QuteMoi* que é um visualizador biomolecular, de código aberto e funciona em tempo real, que oferece diversos efeitos visuais; Metro que é uma ferramenta desenvolvida para medir a diferença entre duas malhas de objetos 3D, formadas por triângulos, e que, de acordo com o *google scholar* [goo, 2009] possui acima de 560 citações; e o ***Meshlab*** que foi descrito acima [vcg, 2009].

3.1.2 *Qt - Nokia GUI Toolkit*

Para conseguir a portabilidade que possui, o ***Meshlab*** precisou utilizar bibliotecas e estruturas que mantivessem essa característica multi-, e na parte de Interfaces Gráficas com o Usuário (GUI) utiliza o Qt (lê-se: *"cute"*). O Qt é um sistema para o desenvolvimento de GUI's e também é utilizado para desenvolver programas como ferramentas para consoles e servidores. Atualmente, este sistema foi assumido pela divisão de desenvolvimento de sistemas da Nokia, desde que a mesma adquiriu a *Trolltech*, a original produtora do Qt, em 2008 e tem o seu uso amplamente difundido em aplicações de renome internacional. Por exemplo: *Adobe Photoshop*, *Google Earth*, o navegador Opera e *Skype* [qt:, 2009]. Assim como o *Meshlab*, o Qt também possui o seu código aberto, utiliza a linguagem C++ e é distribuído sobre os termos da GNU (*General Public License*).

Além desse sistema, o Qt apresenta uma ferramenta de desenvolvimento chamada *Qt Designer* que pode ser utilizada no desenvolvimento e implementação de interfaces com o usuário construídas com base no sistema Qt. Com essa ferramenta, o trabalho de criar uma interface pode ser considerado de forma mais alto-nível. A interface da mesma possui alguns componentes prontos, e uma forma fácil de definir a estrutura do seu conteúdo, tornando assim mais fácil a criação e atualização das telas utilizadas no programa, pois possui uma integração com o sistema do Qt e pode ter o seu código gerado automaticamente a partir da extensão do arquivo do *Qt Designer*: ".ui".

3.2 Visão Geral do Sistema

Após a realização dos passos de configuração do Qt, explicados em maiores detalhes no Apêndice A, podem ser seguidos os passos descritos pelo *site* do *Meshlab* [mes, 2009], para conseguir obter uma *solution*, no *Visual Studio*. Após essa fase, pode-se dar início à primeira *build* do projeto. Após a mesma, resultarão 12 erros de compilação. Basta realizá-la novamente, pois os erros são causados por problemas de dependência de projetos que ainda não haviam sido compilados.

Pela complexidade do algoritmo implementado e pela necessidade de uma interface mais bem detalhada, foi necessário desenvolver uma ferramenta de edição para o *Meshlab*. O próprio grupo de desenvolvedores do sistema informa que caso seja necessário ter uma interface com uma maior quantidade de controles, é aconselhável desenvolver um ferramenta de edição, pois a mesma permite maior liberdade para o usuário final. Caso contrário, poderia ter sido desenvolvido apenas um filtro que permite menor liberdade para o controle dos campos [mes, 2009].

A interface gráfica utilizada no desenvolvimento é bastante simples. Explicações mais detalhadas referentes aos valores encontrados na Figura 3.1 podem ser vistos a seguir:

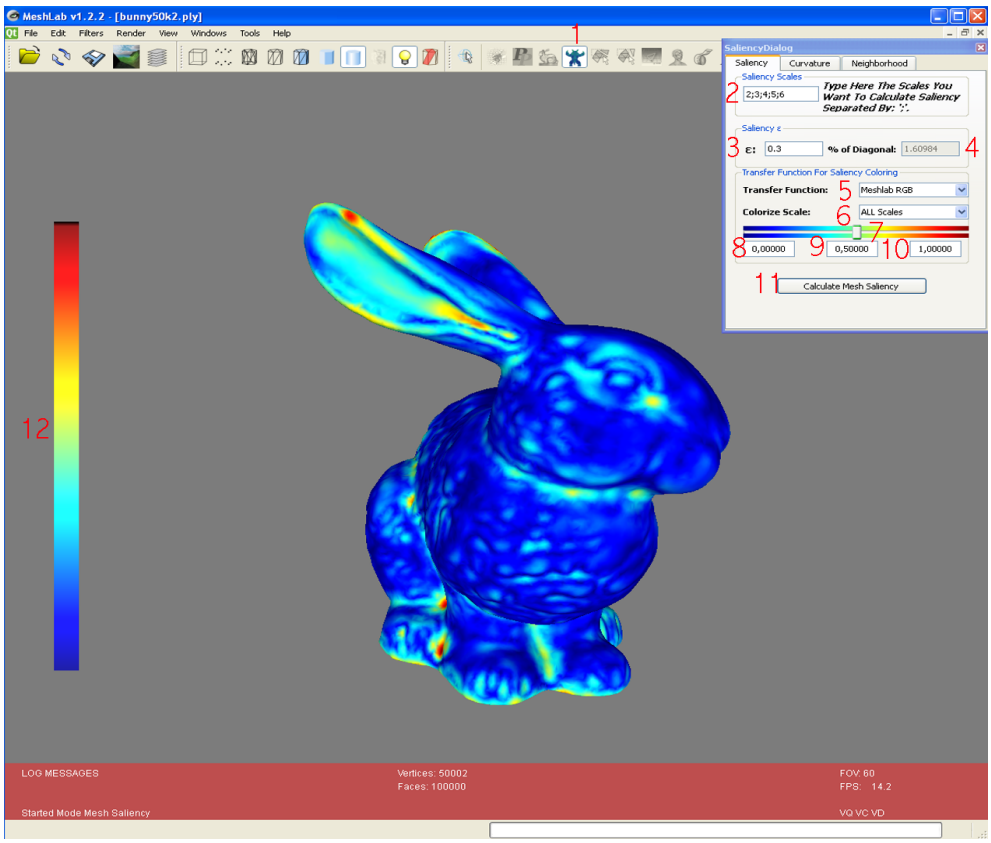


Figura 3.1: Interface utilizada para o cálculo da saliência em um objeto 3D.

1. Ícone que seleciona a ferramenta de cálculo de saliências no *Meshlab*.

2. Neste campo, as diferentes escalas que serão utilizadas no cálculo da saliência podem ser escritas separadas por ';'. Assim como no artigo, inicialmente este campo contém as escalas $2\varepsilon, 3\varepsilon, 4\varepsilon, 5\varepsilon, 6\varepsilon$.

3. Aqui, o usuário possui liberdade para escolher qual a porcentagem, do valor contido em **4.**, ele quer utilizar para definir o valor de ε. Assim como no artigo, este campo tem o seu valor iniciado em 0,3% (Ver Figura 3.2 para melhor compreensão do resultado visual com a mudança deste campo).

4. Explicita o valor da diagonal principal da *bounding box* do modelo 3D. Este campo não pode ter seu valor alterado pelo usuário.

5. Permite que seja alterada a função de transferência de cor para os valores da saliência. Algumas funções encontram-se pré-definidas. As mais utilizadas foram: *Meshlab* RGB, *Gray Scale*, RGB e *Red Scale* por apresentarem melhor resultado visual para testes.

6. Aqui, o usuário pode alterar qual a escala do mapa de saliência que ele deseja visualizar e ter seus valores passados para o *shader* descrito na seção 3.4 deste mesmo capítulo.

7. Essa barra permite que o usuário altere o balanceamento das cores. O algoritmo de coloração não é linear (Ver Figura 3.3 para melhor entendimento do efeito visual causado com a variação nesta barra).

8. Este campo contém o valor mínimo da saliência na escala selecionada.

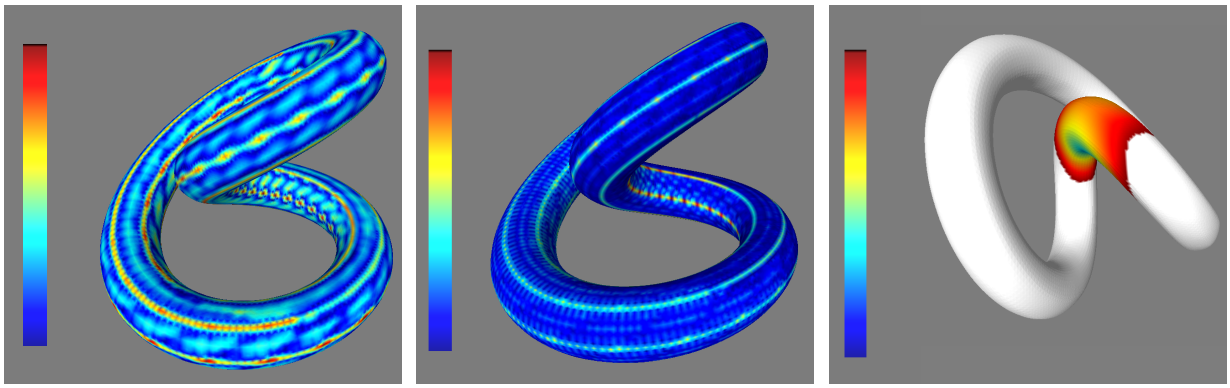
9. O valor aqui representado pode ser alterado com a utilização da barra descrita no item 7.

10. Contém o valor máximo da saliência na escala.

11. Botão que dá início ao cálculo da saliência com os valores dos campos **2** e **3**.

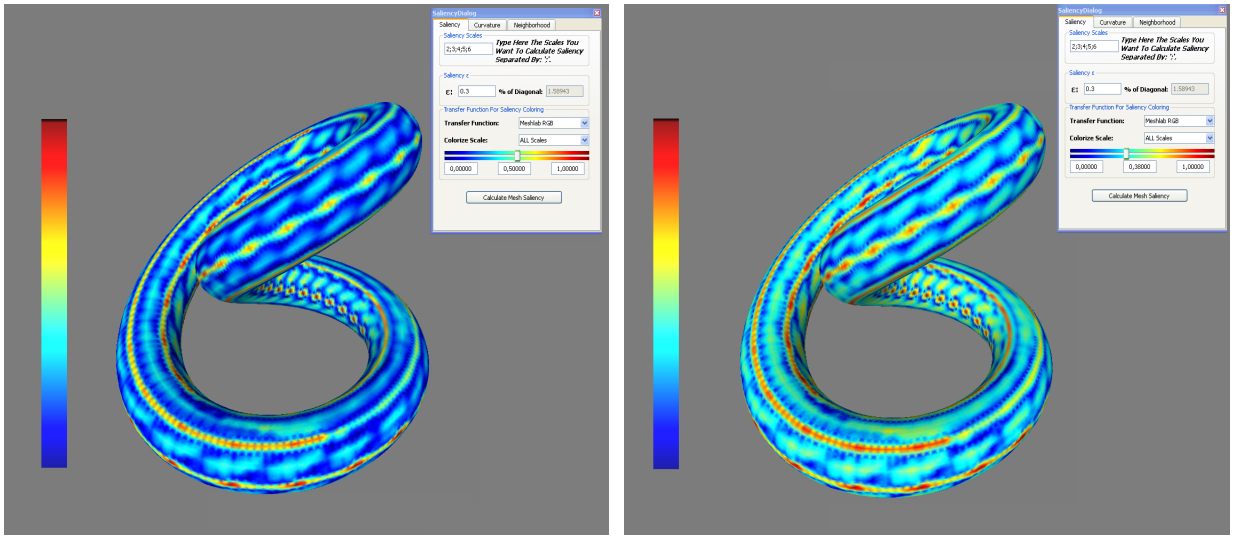
12. Barra que representa a escala de cores, de forma vertical, que está sendo utilizada.

Além desta interface principal, existem mais duas abas que podem ser utilizadas para a realização de testes sobre os algoritmos de distância e curvatura utilizados, além de selecionarem que algoritmos serão utilizados no cálculo da saliência. Na Figura 3.4, pode ser vista a interface que realiza o cálculo da curvatura média e colore cada vértice do objeto de acordo com a função de transferência de cor selecionada. Já na Figura 3.5, o usuário pode inserir o valor de uma distância na qual ele quer encontrar os vizinhos de um vértice aleatório. Esta vizinhança será exibida com a coloração da função de transferência de cor selecionada.



(a) (b) (c)

Figura 3.2: Saliência em múltiplas escalas do modelo *Knot*, visto em (a) e (b). Em (a), foi utilizado $\varepsilon = (0,3\%\ d)$ e em (b), $\varepsilon = (0,09\%\ d)$. Com d representando o valor da diagonal principal da *bounding box* do modelo. Em (c), podemos ver a vizinhança de um vértice aleatório calculada na escala 6ε com $\varepsilon = (0,3\%\ d)$. A razão pela qual a saliência em (b) é mais coerente do que em (a), é porque a distância Euclidiana alcança o outro lado da malha com $\varepsilon = (0,3\%\ d)$, e por isso, o resultado da saliência em (b) é mais coerente.



(a) (b)

Figura 3.3: Em (a) e (b), foram calculadas as saliências em múltiplas escalas utilizando os mesmos valores. A única diferença entre as duas figuras está no valor do balanceamento das cores. Em (a), o valor está na posição mais comum, com valor de 0,5. Em (b), o valor do balanceamento de cores está em 0,3. Vale ressaltar que a função de coloração não é linear.

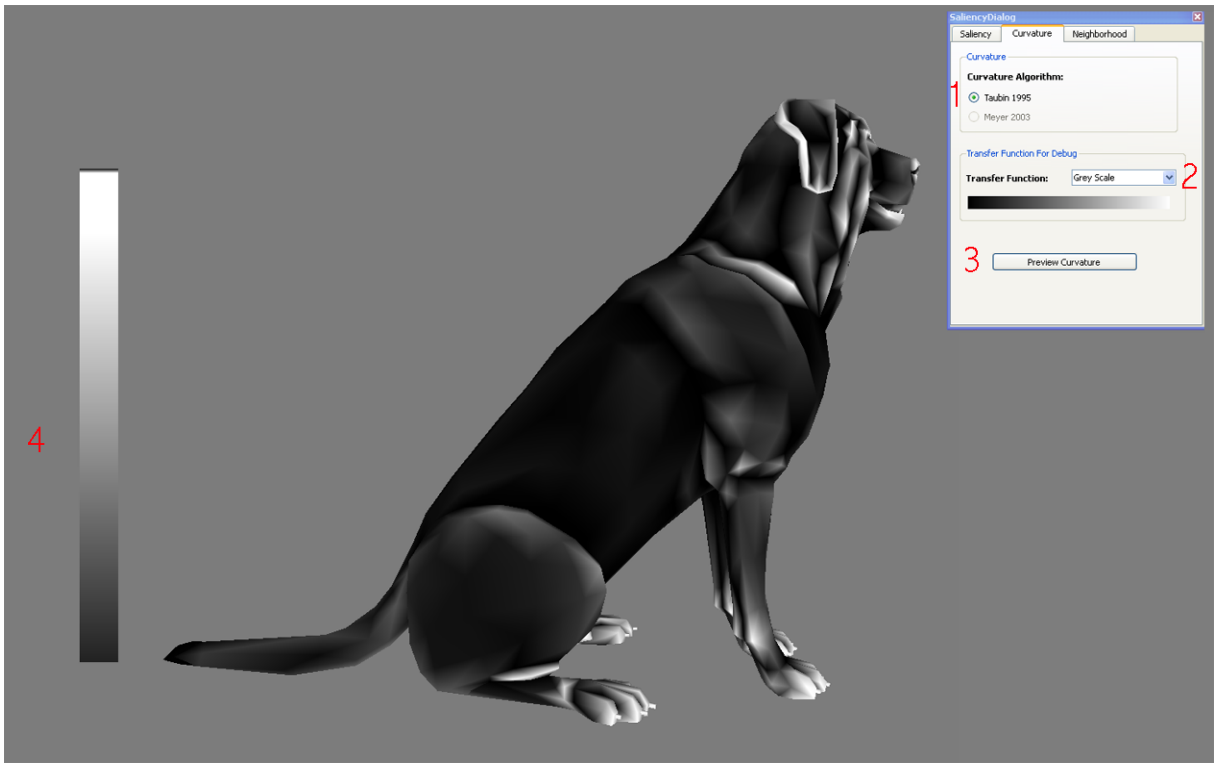


Figura 3.4: Interface utilizada para a visualização da curvatura média do objeto 3D. No campo (1), o usuário pode selecionar o algoritmo desejado para o cálculo da curvatura (Atualmente só pode ser usado o algoritmo de Taubin [1995]). Já na lista (2), o usuário pode escolher que função de coloração usar. O botão (3) realiza o cálculo e a coloração da curvatura média e a escala (4) mostra a função de coloração na vertical.

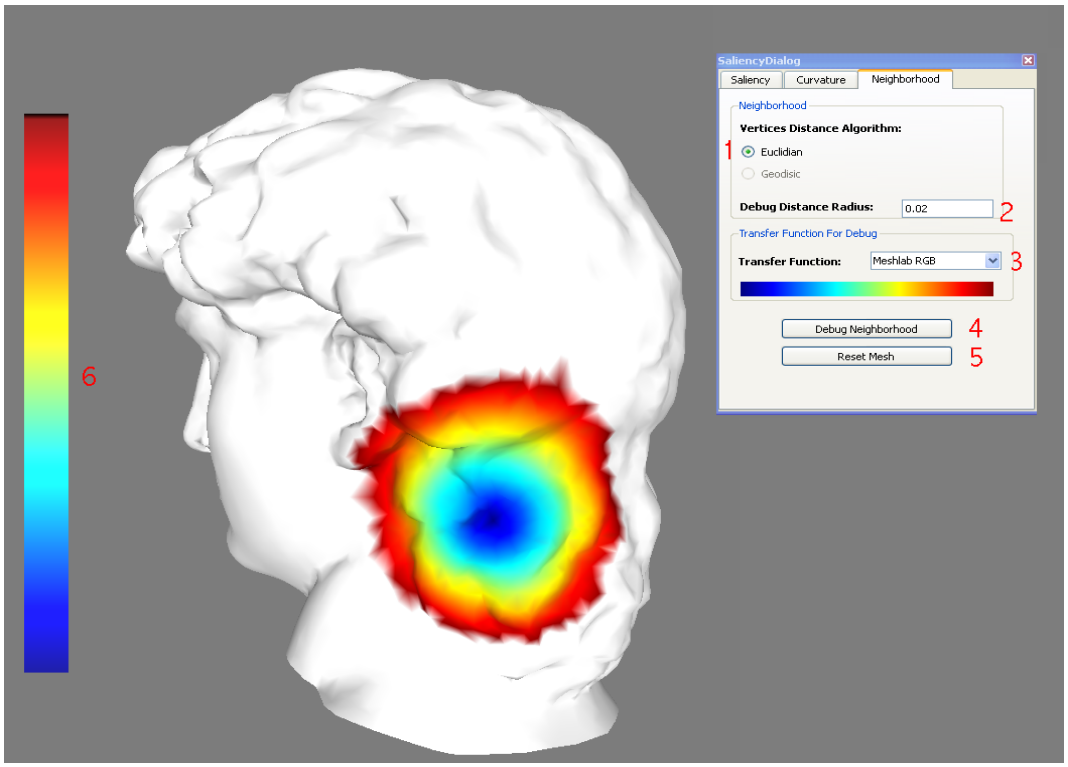


Figura 3.5: Interface utilizada para a visualização da vizinhança com o algoritmo escolhido e a sua representação gráfica. No campo (1), pode ser selecionado o algoritmo de distância entre vértices (Atualmente só o algoritmo de distância Euclidiana pode ser ativado). Já no campo (2), o usuário pode definir qual o tamanho do raio usado no cálculo da vizinhança. Na lista (3) há a possibilidade da escolha da função de transferência de cor e a sua representação vertical em (6). No botão (5), o usuário pode limpar a coloração da malha e com o botão (4), executar o cálculo da vizinhança a partir de um vértice aleatório.

3.3 Cálculo da Saliência

Diversas etapas foram realizadas durante o desenvolvimento do projeto. Para o cálculo das duas principais curvaturas em cada um dos vértices, foi utilizada a biblioteca ***VCG Lib*** com o algoritmo de Taubin [1995]. Após encontrados esses dois valores principais, foi realizada uma média aritmética sobre os mesmos resultando no valor da curvatura média do vértice. Para relacionar um vértice com o valor da sua curvatura média, foi utilizada a função $\mathscr{C}(\mathbf{v})$, onde $\mathbf{v}$ representa um dos vértices que compõem a malha de objetos 3D. Após o cálculo destes valores para todos os vértices, é possível dar início aos cálculos relacionados à obtenção da saliência em cada uma das escalas selecionadas.

Antes de explicar como realizamos o cálculo da saliência nas escalas escolhidas, precisamos detalhar um pouco mais o nosso algoritmo de vizinhança. Para tal processo, foi utilizada a implementação de adjacência entre vértice e face da *VCG Lib* melhor explicado na Figura 3.6. Nela, é exemplificado o funcionamento das adjacências de um determinado vértice. A partir de um vértice $\mathbf{v}$, é possível percorrer todas as faces f_i que são adjacentes ao mesmo. Desta forma,

é possível percorrer a vizinhança de um determinado vértice, de forma recursiva, e encontrando os vértices presentes nas faces adjacentes, marcando faces e vértices já visitados, e definindo se a distância entre ele e o vértice central é menor que a distância máxima desejada. O Algoritmo 1 explica melhor a forma como foi implementada a busca de vértices vizinhos na nossa aplicação.

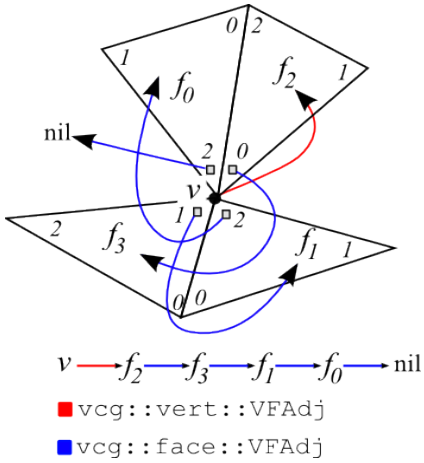


Figura 3.6: Exemplifica como funciona a implementação das faces adjacentes a um determinado vértice **v**. O ponteiro para *nil* indica que chegou ao fim das faces adjacentes. A cor vermelha representa um ponteiro partindo de um vértice e a azul, partindo de uma face [vcg, 2009].

Após um maior conhecimento sobre o pré-processamento para o cálculo de curvatura média na malha e do nosso algoritmo de vizinhança, é possível iniciar a etapa do cálculo de saliências nas escalas selecionadas pelo usuário, através da interface da nossa ferramenta de edição. Antes de calcular a mesma com os parâmetros desejados, o algoritmo verifica se já existe um mapa de saliência em algumas das escalas, passadas na lista de escalas a serem calculadas, e com os mesmos parâmetros, salvo em um arquivo de extensão ".log". Caso exista, o mapa de saliência daquela escala será carregado a partir do arquivo e a escala sairá da lista de escalas que precisam ser calculadas. Esta função visa poupar tempo com o cálculo de algo que já foi previamente calculado e armazenado.

Para as escalas restantes na lista de escalas escolhidas pelo usuário, o valor da saliência terá de ser calculado para a malha 3D em cada uma delas. Para este passo, devido ao alto custo computacional na busca pela vizinhança de um vértice, foi realizada uma mudança. Ao invés de calcular a saliência para toda a malha, compondo o objeto 3D, em cada uma das escalas, calculamos a saliência em todas as escalas para cada um dos vértices que formam a malha do objeto 3D. Desta forma, conseguimos a vizinhança ao redor do vértice, em todas as escalas necessárias, em um só passo sabendo que as menores escalas estarão contidas nas maiores (Figura 3.7) e a partir da vizinhança. Logo, é possível calcular a saliência no vértice em todas as escalas como pode ser visto no Algoritmo 2.

Todos os mapas de saliência que precisaram ser calculados, ou recalculados, são normalizados para valores no intervalo [0, 1] e armazenados em um arquivo. Nos casos em que alguma

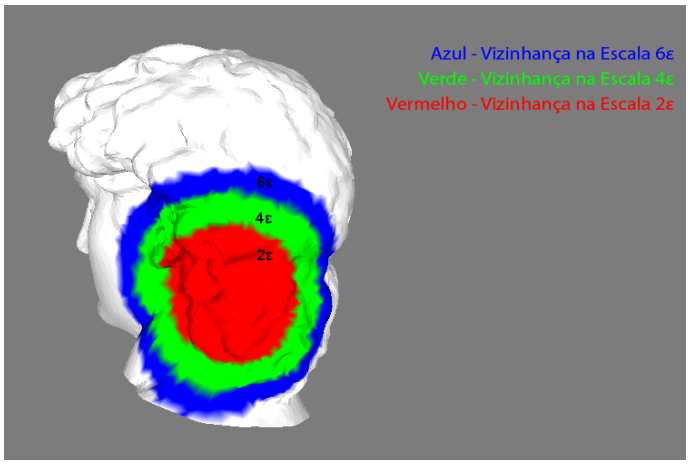


Figura 3.7: Interface utilizada para controlar os valores dos limiares do nosso *shader*. Os valores controlados pelos *sliders* podem variar no intervalo de [0,1].

das escalas de saliência escolhida precisa ser calculada, ao invés de carregada através de um arquivo, é preciso também recalcular a saliência em múltiplas escalas. Sendo assim, todos os mapas de saliência em diferentes escalas têm o seu fator multiplicador de supressão calculado, através da fórmula citada no capítulo anterior, e são então multiplicados pelo mesmo. Depois, os valores de saliência de cada um dos vértices compondo a malha de objetos 3D em cada uma das escalas calculadas são somados resultando em um mapa de saliência em múltiplas escalas. Que, por sua vez, também é armazenado em um arquivo que informa a partir de que escalas ele foi calculado e qual o valor de ε. Dessa forma, podemos armazenar os valores para facilitar o processo de testes realizados na etapa com a renderização em NPR.

3.4 NPR Baseado em Saliência de uma Malha 3D

Uma tendência desde a época do lançamento do livro de Rost [2005], é a de substituir funcionalidades normalmente fixas no *hardware* gráfico por funcionalidades programáveis em *software*. Em OpenGL, duas grandes áreas que tiveram o aumento de programabilidade foram o processamento de vértices e processamento de fragmentos ou *pixels*. O primeiro envolve as operações que ocorrem por vértice na GPU, e normalmente está relacionado com transformações e iluminação. O segundo, por sua vez, representa estruturas que compõem a imagem a ser desenhada na tela e são criadas no processo de rasterização das primitivas gráficas. Em OpenGL, um fragmento contém todos os dados necessários para atualizar uma única posição no *frame buffer*. A linguagem de *shaders* de OpenGL foi desenvolvida para permitir que programadores pudessem expressar o processamento que deveria ocorrer nessas áreas, descritas acima, do *pipeline* de OpenGL [Rost, 2005].

Neste contexto, *shader* representa o código, escrito em *OpenGL Shading language* (GLSL), designado para a execução em um dos processadores programáveis. Sabendo que podem ser escritos códigos para o processamento de vértices e de fragmentos, tem-se então dois tipos de *shaders* que podem ser definidos em GLSL: *vertex shader* e *fragment shader*. Os *shaders*

Algoritmo *Cálculo de Vizinhança de V*
Entrada V_c - Vértice central
L_l - Lista de Valores das Escalas Ordenada de forma Ascendente.
Saída M - *Hash Map* que conecta um determinado vértice com o valor quadrado da sua distância até o vértice central V_c.

```
1  marque V_c como visitado
2  L_adj ←lista de faces adjacentes ao vértice V_c
3  para cada f em L_adj faça
4      se f não foi visitada então
5          marque a face f como visitada
6          {Para cada Vértice Contido na Face, repita o Algoritmo}
7          para cada v_i em f faça
8              se v_i não foi visitado então
9                  d^2 ←distânciaQuadrada entre V_c e v_i
10                 l_i ←null
11                 {Verifique se a Distância Quadrada é menor que algum dos valores}
12                 para cada valor limite l em L_l faça
13                     se d^2 ≤ l então
14                         l_i ←l
15                     senão
16                         pare
17                 se l_i ≠ null então
18                     {Recomece o Algoritmo a partir do passo 2 utilizando a lista de faces de
v_i ao invés da de V_c}
```

Algoritmo 1: Pseudo-código que representa a forma, recursiva, como foi realizada a busca pela vizinhança de um vértice V_c em nossa aplicação.

Função *ComputarSalienciaEmV*
Entrada L - Lista das Escalas a terem a saliência computada
v - Vértice a ter Saliência computada
Saída L_s - Lista com a saliência de v em cada uma das escalas.

```
1  {Esta função retorna mapas que contêm os vértices que são vizinhos de v em cada uma das
   escalas e o valor da distância ao quadrado para cada um dos vértices.}
2  n ←vizinhancaDeVEmEscalas(v, L)
3  para cada e em L faça
4      {O Algoritmo utilizado foi o Algoritmo de Lee et al. [2005].}
5      L_s ←compute a saliência em v na escala e.
6  devolva L_s
```

Algoritmo 2: Função que calcula a saliência em um vértice v em múltiplas escalas.

escritos em GLSL são chamados de *OpenGL Shaders* para diferenciá-los dos desenvolvidos em outras linguagens que também suportam essa liberdade como *High Level Shading Language - HLSL*, *RenderMan* e *Cg Shaders* [Rost, 2005].

Devido ao suporte, já existente no *Meshlab*, para a aplicação de *shaders* desenvolvidos em GLSL e em HLSL, optamos por desenvolver um *shader* para a Renderização Não Foto-realística sobre o modelo 3D. Contudo, o sistema do *Meshlab* possui algumas limitações. No mesmo, não há uma forma de passar informações da aplicação para o *vertex shader* e, por isso, foi necessário fazer algumas alterações na base do processo de renderização do *Meshlab* e do *VCG Lib* adicionando um novo parâmetro que indica se há informações, contidas no objeto 3D, que devem ser passadas para o *vertex shader* ou não. Dessa forma, pudemos passar parâmetros de entrada para o nosso *vertex shader*, tornando assim, possível a utilização dos valores do mapa de saliência para a realização dos cálculos para a renderização em NPR.

Com o desenvolvimento do nosso *OpenGL shader*, tentamos agregar conteúdo artístico à imagem final a partir dos valores encontrados no mapa de saliência. Para tal objetivo, utilizamos uma combinação de silhuetas com as informações obtidas durante o cálculo de saliências. Para as silhuetas, utilizamos um algoritmo de apenas um passo, sobre o objeto 3D de forma direta, pois dessa forma poderíamos utilizar os valores de saliência contidos nos vértices do mesmo. O algoritmo utilizado visa encontrar silhuetas baseado no coseno do ângulo θ formado entre a normal $\vec{N}$ do vértice e o vetor que parte do olho do observador em direção ao ponto v: definido por $\vec{E}$ (Figura 3.8). Caso o ângulo θ seja menor que um determinado limiar, o vértice será considerado silhueta.

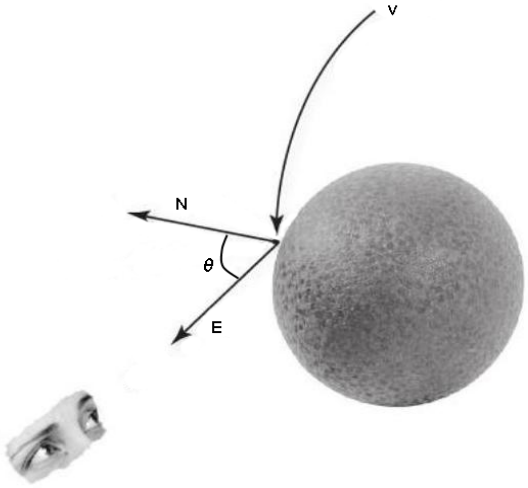


Figura 3.8: Podemos ver o vértice v e o vetor que vai do mesmo ao olho do usuário, denominado vetor $\vec{E}$. Além deles, podemos também visualizar o vetor normal em v: $\vec{N}$. O ângulo θ formado entre $\vec{N}$ e $\vec{E}$ é quem define se o vértice v será marcado como silhueta ou não. Imagem retirada de Rost [2005].

Além das informações obtidas através das silhuetas, o nosso *shader* visa destacar partes de saliência do objeto 3D. Desta forma, utilizamos o coseno do ângulo θ como um indicador de

inclinação da direção da face, que contém o vértice, em relação ao olho do observador. Caso o coseno seja 0, o ângulo formado entre os vetores $\vec{N}$ e $\vec{E}$ é de 90^o. Logo, este vértice pertence a uma face inclinada para uma direção perpendicular à câmera. Caso v não seja considerado silhueta, esse ponto terá o seu indicador de inclinação utilizado como um fator multiplicador sobre a saliência $\mathscr{S}_i(\mathbf{v})$ do mesmo na escala i selecionada, com intuito de destacá-lo caso o seu valor estiver entre dois limiares (mínimo e máximo) de saliência. A interface que permite o controle desses limiares citados acima e do limiar de silhueta pode ser vista e melhor compreendida na Figura 3.9 e mais detalhes referentes aos valores encontrados na figura podem ser vistos a seguir:

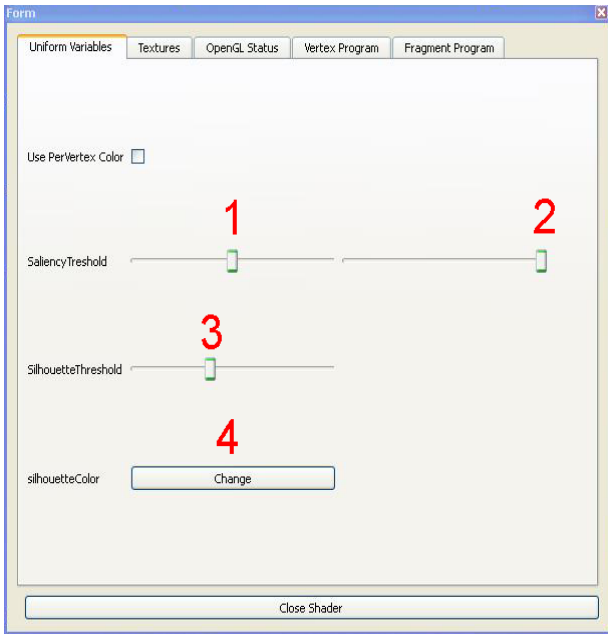


Figura 3.9: Interface utilizada para controlar os valores dos limiares do nosso *shader*. Os valores controlados pelos *sliders* podem variar no intervalo de [0,1].

1. Esta barra controla o valor do liminar mínimo de saliência.
2. Esta outra barra controla o valor do liminar máximo de saliência.
3. Este *slider* controla o valor do limiar de silhueta.
4. Este botão permite que o usuário altere a cor da silhueta mostrada na imagem.

Vale ressaltar que as imagens geradas não necessariamente contém um valor artístico mais alto, pois este conceito não é algo facilmente mensurável. Por esta razão, as ferramentas desenvolvidas permitem maior liberdade ao usuário para que o mesmo possa atingir os resultados que deseja. No próximo capítulo, serão demonstrados alguns resultados e o potencial de liberdade dado pela interface da ferramenta.

CAPÍTULO 4

Resultados

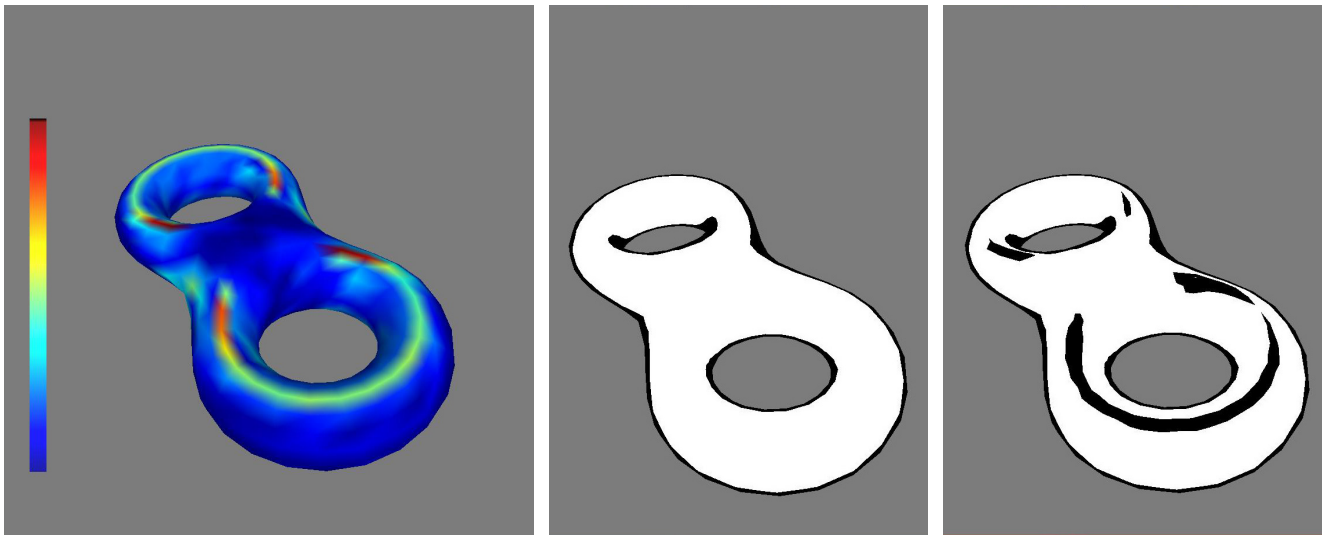
Os *shaders* desenvolvidos na seção 3.4 serão analisados neste capítulo, sob a perspectiva de conteúdo artístico. Essa não é uma análise fácil de ser realizada e, por isso, será feita de uma forma que não julga se algum modelo foi renderizado com uma boa qualidade artística. A razão disso é que o conceito que define um bom resultado artístico é bastante abstrato e depende de conhecimento mais específico. Desta forma, serão apresentados aqui os melhores resultados sob a perspectiva de informação perceptual e com parâmetros ajustados para tal. Para a explicação mais detalhada dos resultados, foram escolhidos os seguintes objetos:

Na Figura 4.1, foi utilizado o objeto *Bitorus* que possui apenas 768 vértices. Na Figura 4.1 (a), podemos visualizar a saliência em múltiplas escalas do mesmo com a sua função de coloração. Na Figura 4.1 (b), é possível visualizar a representação do nosso método de calcular silhuetas, dando uma boa noção do contorno e da forma do objeto. Já na Figura 4.1 (c), podemos visualizar o *shader* composto pela silhueta e por algumas informações dos valores de saliência. Na terceira imagem, é possível ter uma maior percepção de volume do objeto e isso se dá, devido ao valor agregado pela informação proveniente da saliência.

Já na Figura 4.2, podemos ver o modelo 3D do objeto *Knot* que teve resultados bastante expressivos no que diz respeito ao volume. Esse objeto é um pouco diferente do geral, pois devido ao seu formato fino e a sua forma entrelaçada, ele precisou ter o seu valor de ε modificado de 0,3% para 0.09% para conseguir obter resultados de sua vizinhança sem atravessar a malha inteira e pegar vizinhos não tão próximos (Este pode ser um modelo exemplo para casos em que a distância Euclidiana é prejudicial). Na Figura 4.2 (a), vemos o mapa de saliência em múltiplas escalas e com o valor alterado de ε. Já na Figura 4.2 (b), é possível ver o contorno e a rugosidade interna da malha. Na Figura 4.2 (c), vemos o resultado final que consegue expressar melhor a sua forma e os seus pontos mais diferenciados.

O último objeto 3D escolhido para ser apresentado neste capítulo foi a Vênus de Milo. Na Figura 4.3 (a), é possível visualizar o mapa de saliência do modelo na escala 6ε. Na Figura 4.3 (b), vemos uma silhueta mais grossa que define a forma da Vênus. Já na Figura 4.3 (c), é possível ver a sutil diferença entre esta imagem e a imagem sem o conteúdo de saliência. Com a informação de saliência, é possível visualizar melhor o umbigo da estátua além de valorizar os traços que dividem as pernas da mesma, dando uma maior noção do volume.

Com o intuito de aumentar o poder de reutilização do que foi produzido neste trabalho, serão detalhados todos os valores utilizados. Para todos os casos aqui apresentados, o valor utilizados de ε foi de 0.3% da diagonal principal da *bounding box* do modelo (com exceção do modelo *Knot* que utilizou $\varepsilon = 0.09\%$). Para os mapas de saliência que estão explicitados como calculados em múltiplas escalas, as escalas que compõem os mesmos são: 2ε, 3ε, 4ε, 5ε e 6ε. Para os valores utilizados no controle dos *shaders*, a Tabela 4.1 contém a informação



(a) Saliência em Múltiplas Escalas (b) Silhueta (c) Silhueta com saliência

Figura 4.1: À esquerda, em (a), podemos visualizar a saliência em múltiplas escalas do modelo do *Bitorus*. Na escala apresentada, azul representa pontos de baixa saliência e vermelho, de altas saliências. Já em (b), é possível visualizar a imagem renderizada em NPR com o algoritmo de silhuetas desenvolvido. Em (c), é possível visualizar o *shader* que renderiza a silhueta e algumas informações de pontos salientes.

dos mesmos. Para testar esses resultados utilizando o *Meshlab* é necessário colocar os mesmos na pasta de *shaders* do sistema. Dentro do sistema, basta acessar a aba "Render", e o item "Shaders" do menu. Lá será possível encontrar o *shader* "Silhouette.gdp", basta selecioná-lo e inserir os valores contidos na Tabela 4.1. Para realizar testes com barras de *slider*, basta executar o *shader* "Silhouette2.gdp" que contém barras que ajudam a manipular os valores de forma mais dinâmica.

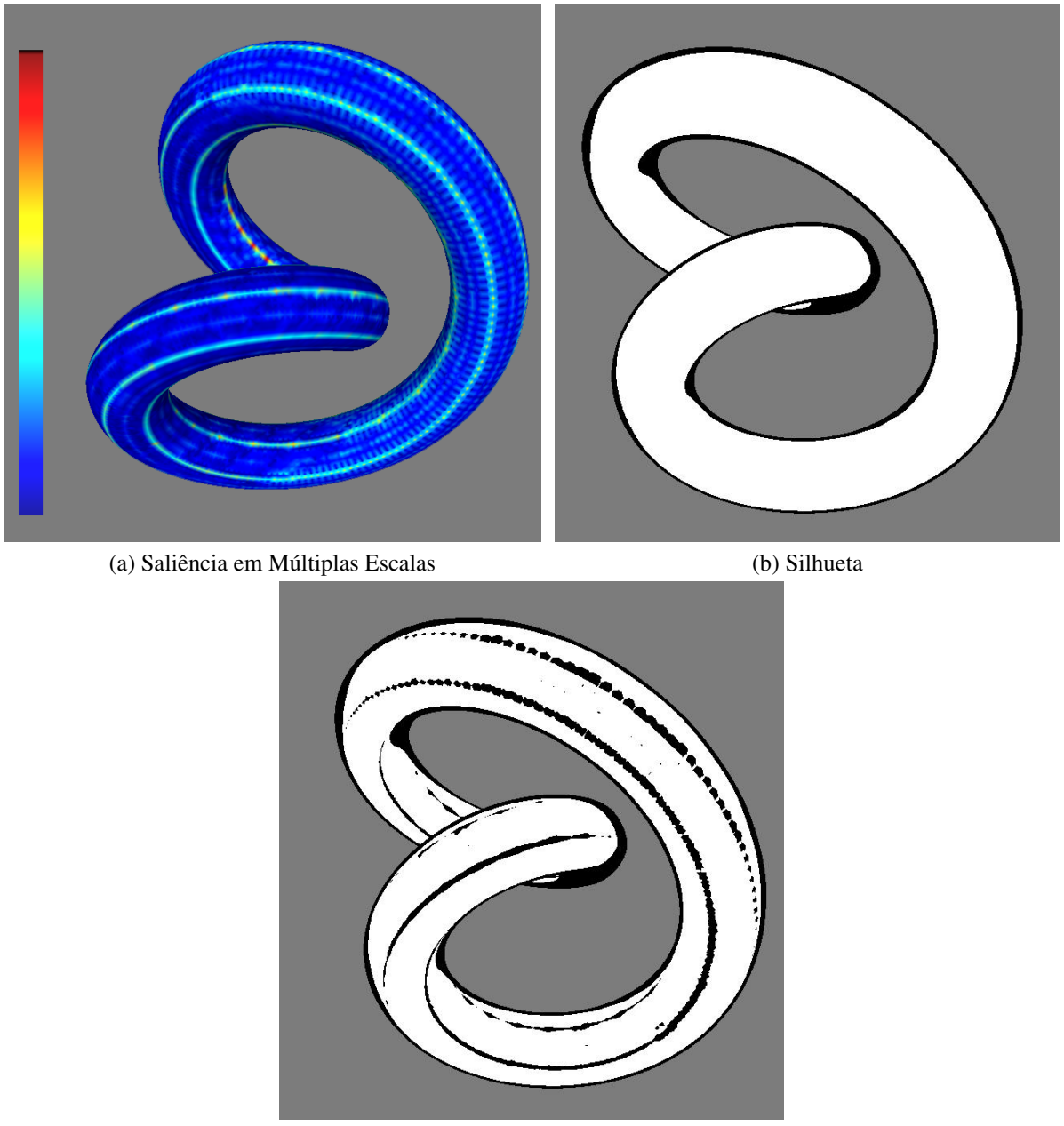
Tabela 4.1: Valores Utilizados nos Exemplos Mostrados. Na coluna "Escala", é informado qual escala foi utilizada para os valores de saliência da imagem. "L_{sa} Mín." representa o limiar mínimo utilizado para os valores de saliência. "L_{sa} Máx.", por sua vez, representa o limiar máximo que limita o intervalo utilizado para os valores de saliência. "L_{si}" representa o limiar utilizado para a silhueta apresentada nas figuras.

	Escala	L_{sa} Mín.	L_{sa} Máx.	L_{si}
Bitorus	Múltiplas Escalas	0,3	0,8	0,3
Knot	Múltiplas Escalas	0.25	1,0	0,3
Venus	6ε	0,5	1,0	0,5

4.1 Interface do *Shader*

Nesta seção, serão demonstradas mais algumas imagens exemplificando o controle da interface sobre os resultados nas mesmas. Na Figura 4.4, podemos visualizar o modelo *Horse*, com 35.000 polígonos, e utilizando a saliência 6ε. Na Figura 4.4 (a), foi utilizado um menor limiar mínimo de saliência. Desta forma, muitos pontos marcados como salientes foram desenhados. Na Figura 4.4 (b), foi utilizado um limiar mínimo maior, resultando em uma imagem com os poucos elementos mais salientes sendo destacados. Já na Figura 4.5, podem ser vistos os resultados visuais obtidos com diferentes valores no liminar de silhueta, tornando a mesma, mais grossa ou mais fina, de acordo com o valor, escolhido no controlador, pelo usuário.

Vale ressaltar que todas as figuras expostas neste capítulo, tem como finalidade demonstrar os resultados obtidos com o projeto. Porém, o julgamento do valor artístico de cada uma das figuras, não foi levado em consideração por depender de um critério diferente de avaliação. Todos os modelos demonstrados com os valores de configuração citados, estão presentes neste documento por terem sido julgados como de melhores resultados, mas não necessariamente os que contêm maior valor artístico. Este último conceito, deverá ser definido pelo próprio usuário e, por essa razão, são expostas as ferramentas de controle visando que o mesmo obtenha os resultados desejados.

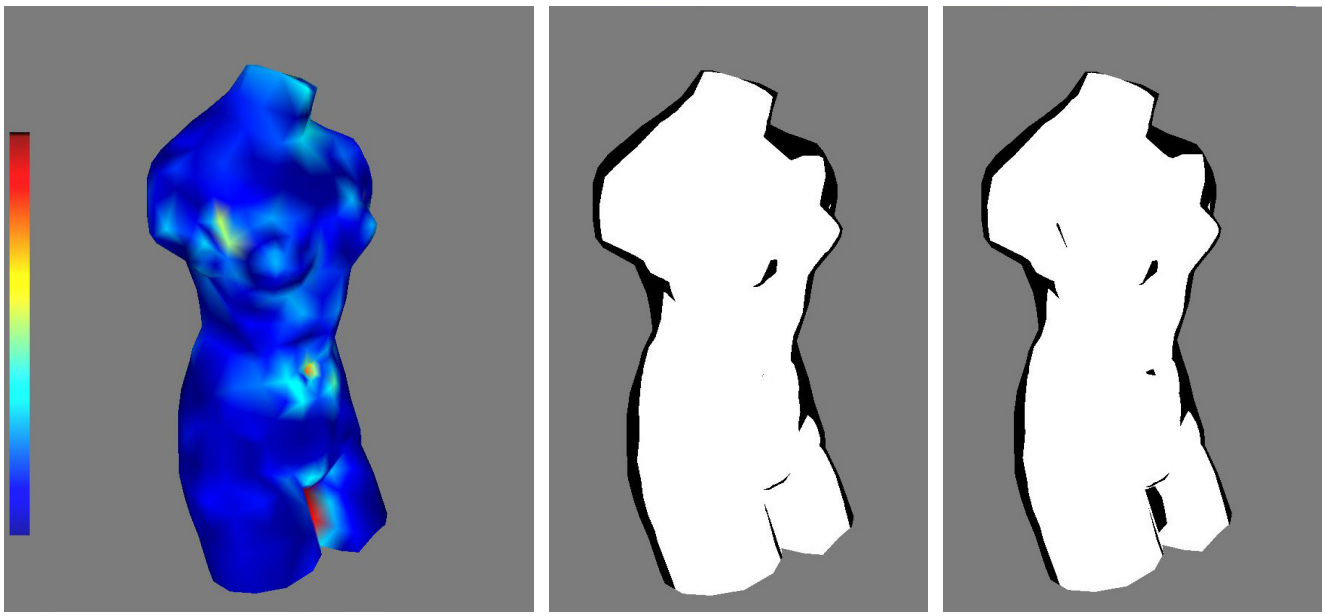


(a) Saliência em Múltiplas Escalas

(b) Silhueta

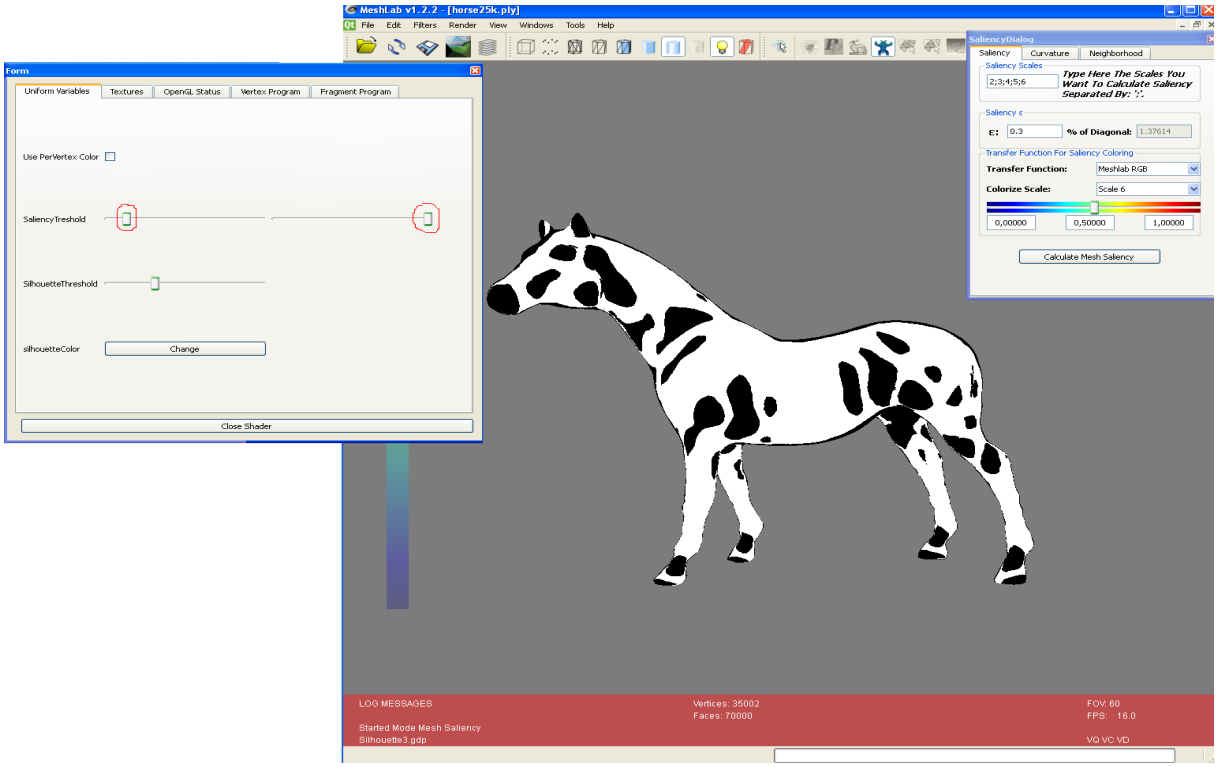
(c) Silhueta com saliência

Figura 4.2: À esquerda, em (a), podemos visualizar a saliência em múltiplas escalas do modelo do *Knot*. Na escala apresentada, azul representa pontos de baixa saliência e vermelho, de altas saliências. Já em (b), é possível visualizar a imagem renderizada em NPR com o algoritmo de silhuetas desenvolvido. Em (c), é possível visualizar o *shader* que renderiza a silhueta e algumas informações de pontos salientes.

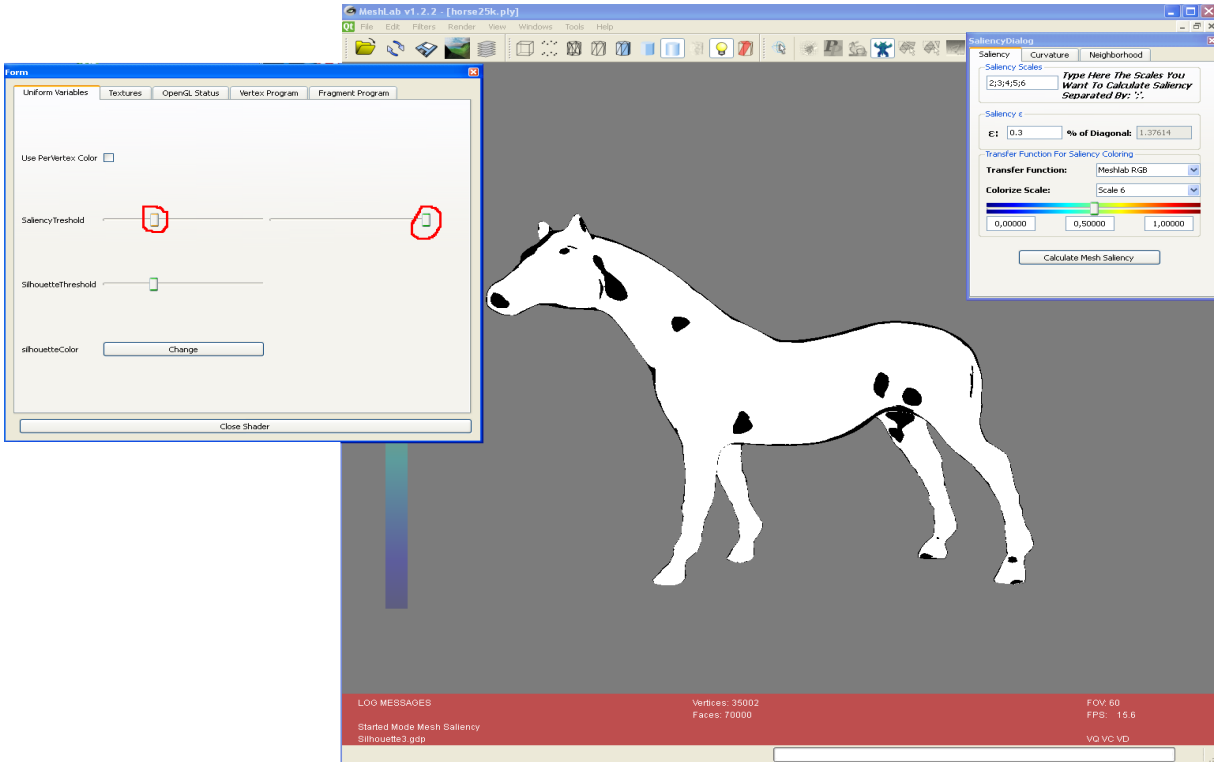


(a) Saliência na Escala 6ε (b) Silhueta (c) Silhueta com saliência

Figura 4.3: À esquerda, em (a), podemos visualizar a saliência na escala 6ε do modelo do *Venus*. Na escala apresentada, azul representa pontos de baixa saliência e vermelho, de altas saliências. Já em (b), é possível visualizar a imagem renderizada em NPR com o algoritmo de silhuetas desenvolvido. Em (c), é possível visualizar o *shader* que renderiza a silhueta e algumas informações do mapa de saliência.

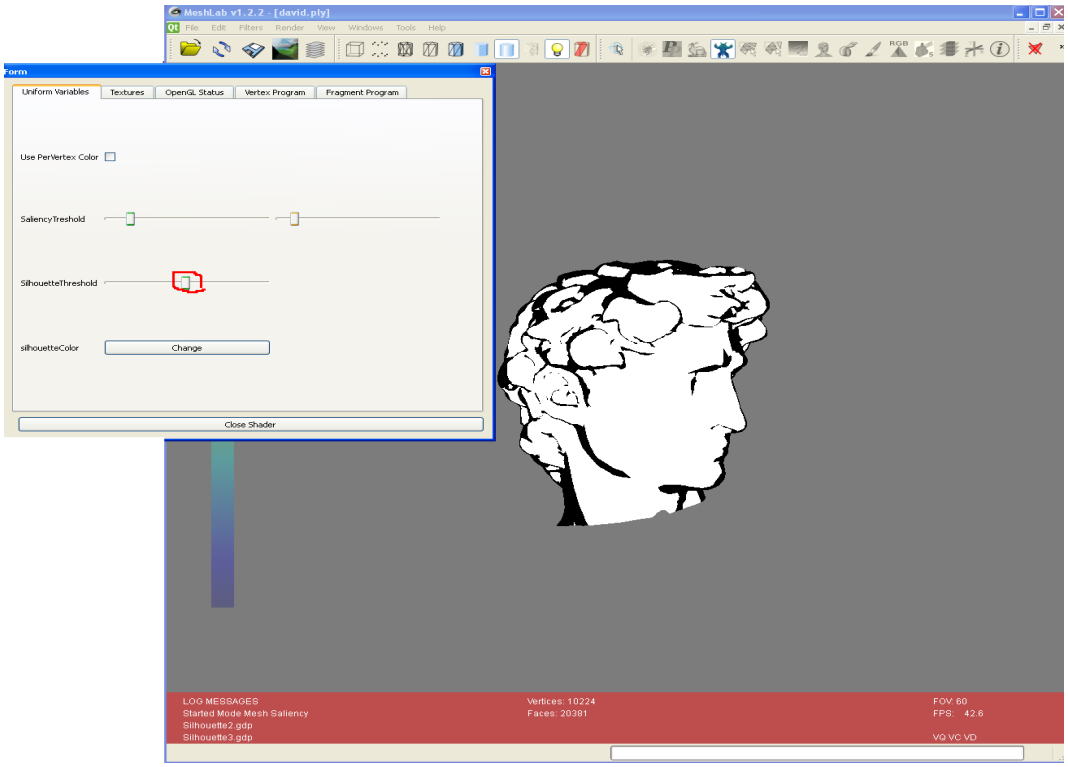


(a)

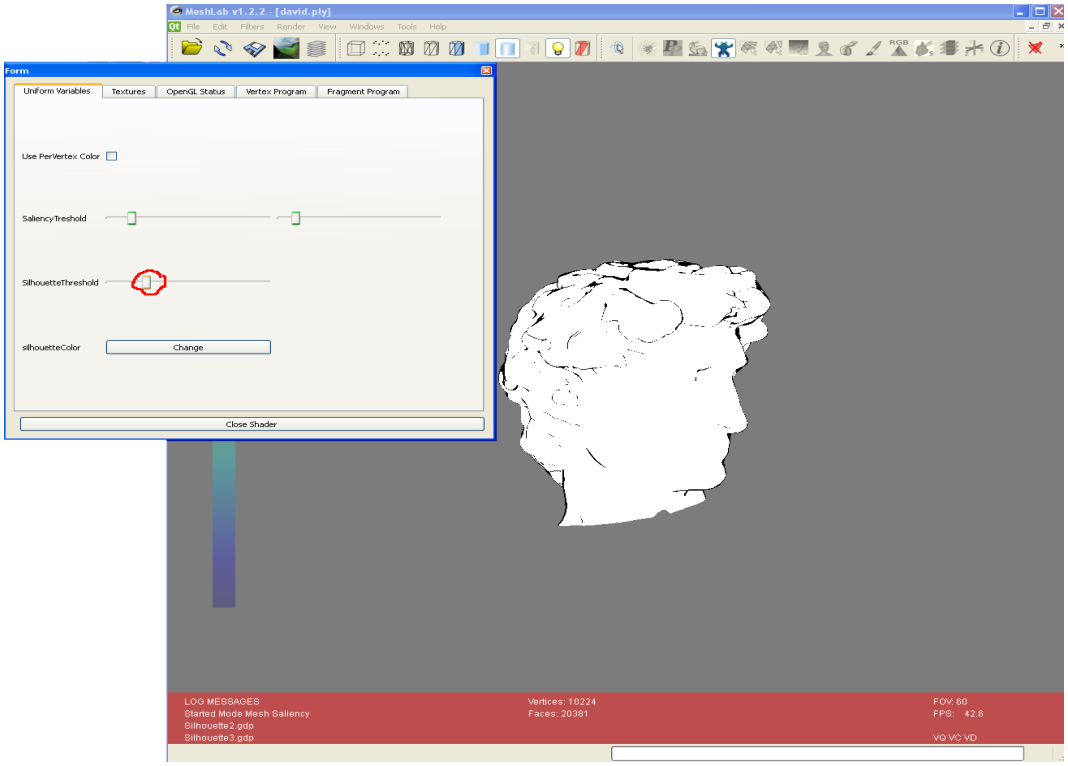


(b)

Figura 4.4: Pode-se visualizar o modelo *horse*, com 35.000 polígonos, renderizado com o nosso *shader* em escala 6ε. À esquerda, em (a), pode-se visualizar que foi usado um liminar mínimo de saliência menor que o utilizado em (b). Dessa forma, em (a), podem ser vistas mais regiões marcadas salientes que em (b).



(a)



(b)

Figura 4.5: Pode-se visualizar o modelo *David*, com 10.000 polígonos, renderizado com o nosso *shader* em múltiplas escalas. À esquerda, em (a), pode-se visualizar que foi usado um liminar de silhueta mais alto que o utilizado em (b). Dessa forma, em (a), podem ser vistos detalhes mais grossos de silhueta que os vistos em (b).

CAPÍTULO 5

Conclusão

Este trabalho apresentou um estudo sobre a área de Computação Gráfica baseada em Percepção, tendo em vista o seu foco em saliência sobre imagens bidimensionais e sobre modelos de objetos 3D. O principal objetivo dessa pesquisa foi de desenvolver o algoritmo de Lee et al. [2005] e utilizar as informações baseadas em percepção com o intuito de gerar imagens, em NPR, contendo conteúdo artístico baseado em percepção. Para tal, foi utilizado um avançado sistema de processamento de malhas que compõem um objeto 3D: o *Meshlab*. Paralelamente a esta implementação do algoritmo, foi realizado um estudo sobre a área de renderizações não foto-realísticas visando unir estas duas áreas para expressar imagens com um maior conteúdo visual.

Dentro deste escopo, pode-se concluir que as metas traçadas no início deste trabalho foram cumpridas de forma satisfatória. A proposta inicial de implementar o algoritmo descrito por Lee [2005], foi concluída com êxito e o fruto deste trabalho pode resultar em diversas aplicações além das propostas neste documento. Além desta implementação, a criação do *shader* composto por silhuetas de um objeto 3D e sua percepção visual foi realizada com êxito e permite ao usuário final uma boa manipulação sobre os valores dos limiares. No capítulo 2, demos vários exemplos de campos já explorados e diferentes usos da informação contida na própria geometria do objeto 3D. Porém, a área de NPR é apenas um exemplo de uma das áreas que pode beneficiar-se deste trabalho aqui descrito.

Vários trabalhos podem se beneficiar a partir do modelo computacionacial de cálculo de saliências descrito neste trabalho. Demos vários exemplos de campos já explorados e existem diversos campos ainda não explorados e informações diferentes a serem utilizadas tanto a partir da geometria como a partir de outros fatores diversos. Até o presente momento, não há nenhuma outra forma de captar saliências para um modelo 3D inteiro, independente de posição de câmera, que se utilize de uma informação proveniente de outra fonte se não geométrica. Por isso, há ainda reais possibilidades de outros fatores serem levados em consideração no cálculo da saliência de uma malha de objeto 3D. Em nosso trabalho, foi levado em consideração o ângulo de inclinação de uma face para saber se a saliência do mesmo dá garantia de maior visibilidade ou não.

5.1 Trabalhos Futuros

Durante o desenvolvimento desta ferramenta de edição, houve uma certa aproximação com a comunidade de desenvolvimento do *Meshlab*. A mesma, mostrou-se bastante interessada em agregar a ferramenta de edição desenvolvida, capaz de calcular as saliências em malhas que

compõem objetos 3D com o algoritmo de Lee, ao seu conjunto de funcionalidades. Por esta razão, os laços vêm sendo estreitados com o grupo e há grandes possibilidades de, após algumas adequações realizadas para cumprir com os padrões de desenvolvimento para o sistema, termos a ferramenta descrita neste trabalho, presente nas próximas versões de distribuição do *Meshlab*. Para isso, porém, existem algumas exigências feitas pela comunidade de desenvolvimento do sistema, com intuito de padronizar e manter um controle de qualidade das ferramentas e filtros desenvolvidos para o mesmo. Assim que forem terminadas as adequações, haverão novos contatos visando a finalização e a agregação do nosso trabalho neste sistema de tamanha importância.

Com o intuito de possibilitar a inferência do usuário final sobre o cálculo da saliência, durante o desenvolvimento deste algoritmo, surgiu a idéia de criar uma ferramenta que possibilite ao usuário atribuir valores a áreas consideradas salientes pelo mesmo. Desta forma, caso houvesse alguma informação considerada importante e que não teve o destaque devido, seria possível que o mesmo destacasse a área através de uma ferramenta de coloração. A partir desta alteração, o algoritmo perderia a característica de ser completamente automatizado, mas permitiria que alguns usuários mais exigentes destacassem áreas consideradas, pelos mesmos, como salientes. Desta forma, há uma grande possibilidade dessa funcionalidade ser desenvolvida, juntamente com a ferramenta de edição do cálculo de saliências, e enviada também à comunidade do *Meshlab*.

Além destes passos citados, há a idéia de inserir uma nova etapa no pré-processamento da malha 3D visando atingir uma maior velocidade no cálculo da saliência em objetos contendo uma grande quantidade de vértices. Uma das idéias é utilizar o auxílio das *kd-trees* para pré-processar os dados da malha e facilitar a tarefa de encontrar e medir a distância quadrada entre um vértice central e a sua vizinhança. Com um maior trabalho dedicado a esse pré-processamento, é possível agilizar o cálculo das saliências, facilitando ainda mais os seus possíveis testes a serem realizados em diversos outros ramos e tornando a ferramenta para calculá-lo no *Meshlab* mais interessante [Heinzle et al., 2008].

APÊNDICE A

Configuração do Qt

Antes de conseguir compilar o código fonte do *Meshlab* é necessário já ter instalado e configurado na máquina o Kit de Desenvolvimento de *Software* (*Software Development Kit* - SDK) do Qt. Todas as informações contidas neste apêndice foram realizadas para a versão 4.5.2 do Qt. Os passos para instalar esse SDK de forma que o mesmo seja capaz de gerar a solução, para o projeto do *Meshlab*, em *Visual Studio* não são fáceis de encontrar e por essa razão, serão detalhados adiante.

1. Inicialmente, basta ter o instalador da versão *open source* do SDK do Qt, para o sistema operacional desejado, e executá-lo. Durante a instalação no *Windows*, caso não tenha em sua máquina, faz-se necessária também a instalação do MingGW que será proposta durante a instalação do SDK. Não é necessária a instalação do código fonte do mesmo.

2. Após ter o SDK do Qt instalado na máquina, é preciso configurá-lo de forma específica para o computador, no qual será usado, e esse processo pode demorar até 4 horas. Para tal, é necessário definir a plataforma de desenvolvimento do código-fonte do *Meshlab*. No nosso caso, utilizamos o *Microsoft Visual Studio 2008*. Após ter essas duas ferramentas instaladas, é preciso abrir o *Visual Studio Command prompt* e acessar a pasta onde está a sua instalação do Qt. Normalmente colocado em **C:/Qt/$versão$**, onde $versão$ corresponde à versão do Qt instalada. Para o desenvolvimento deste projeto, foi utilizada a versão 4.5.2 do Qt.

3. Após acessar a pasta de instalação do Qt com o *Visual Studio Command prompt*, precisa ser digitado o comando de configuração com as devidas propriedades escolhidas para uso futuro. Na configuração para este trabalho, o comando de configuração foi o seguinte:

   configure -msvc2008

   Este processo dura algo em torno de 10 a 15 minutos e vai configurar o Qt com o *Visual Studio*, em sua versão 2008, na máquina.
   Após o Qt estar configurado em sua máquina, é preciso utilizar o comando: **nmake** para que o Qt seja compilado com as configurações escolhidas, no passo anterior, e utilizado com o *Visual Studio* (Este processo dura algo em torno de 4 horas).

4. Após terminados os passos acima, o Qt estará configurado para uso na máquina.

Referências Bibliográficas

Autodesk - fake or foto, nov 2009. Disponível em `http://area.autodesk.com/fakeorfoto`.

Google scholar - portal de busca por literatura acadêmica, nov 2009. Disponível em `http://scholar.google.com.br/schhp`.

Meshlab, nov 2009. Disponível em `http://meshlab.sourceforge.net/`.

Qt - um sistema de desenvolvimento de interface com o usuário multi=plataforma, nov 2009. Disponível em `http://qt.nokia.com/`.

Vcg library, nov 2009. Disponível em `http://vcg.sourceforge.net/index.php/Main_Page`.

Visual computing lab, nov 2009. Disponível em `http://vcg.isti.cnr.it/joomla/index.php?option=com_frontpage&Itemid=1`.

Barni, R., Varshney, A., & Comba, J. Salient clustering for view-dependent multiresolution rendering. In *Proceedings of XXII SIBGRAPI 2009*, 2009.

Britto Neto, L. S. & Carvalho, B. M. Renderizações não fotorealísticas para estilização de imagens e vídeos usando areia colorida. *SIBGRAPI '08: XXI Brazilian Symposium on Computer Graphics and Image Processing*, 2008.

Brosz, J., Samavati, F., & Sousa, M. C. Silhouette rendering based on stability measurement. In *SCCG '04: Proceedings of the 20th spring conference on Computer graphics*, pages 157–167, New York, NY, USA, 2004. ACM.

Cherlin, J. J., Samavati, F., Sousa, M. C., & Jorge, J. A. Sketch-based modeling with few strokes. In *SCCG '05: Proceedings of the 21st spring conference on Computer graphics*, pages 137–145, New York, NY, USA, 2005. ACM.

Chung, A. J., Deligianni, F., Hu, X.-P., & Yang, G.-Z. Visual feature extraction via eye tracking for saliency driven 2d/3d registration. In *ETRA '04: Proceedings of the 2004 Symposium on Eye tracking Research & Applications*, pages 49–54, New York, NY, USA, 2004. ACM.

Feixas, M., Sbert, M., & González, F. A unified information-theoretic framework for viewpoint selection and mesh saliency. *ACM Transactions on Applied Perception*, 6(1):1:1–1:23, Feb. 2009.

Garland, M. & Heckbert, P. S. Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

Gilles, S. *Robust Description and Matching of Images*. PhD thesis, University of Oxford, 1998.

Gooch, A. A. & Gooch, B. Using non-photorealistic rendering to communicate shape. In *ACM SIGGRAPH 1999 Course Notes, Course on Non-Photorealistic Rendering*, 1999.

Green, S., Salesin, D., Schofield, S., Hertzmann, A., Litwinowicz, P., Gooch, A., Curtis, C., & Gooch, B. *Non-Photorealistic Rendering*. SIGGRAPH 99 Course Notes, 1999.

Hagedorn, B. & Döllner, J. High-level web service for 3d building information visualization and analysis. In *GIS '07: Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, pages 1–8, New York, NY, USA, 2007. ACM.

Haller, M. & Sperl, D. Real-time painterly rendering for mr applications. In *GRAPHITE '04: Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 30–38, New York, NY, USA, 2004. ACM.

Halper, N., Schlechtweg, S., & Strothotte, T. Creating non-photorealistic images the designer's way. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 97–ff, New York, NY, USA, 2002. ACM.

Heinzle, S., Guennebaud, G., Botsch, M., & Gross, M. A hardware processing unit for point sets. In *GH '08: Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware*, pages 21–31, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

Hertzmann, A. Introduction to 3d non-photorealistic rendering: Silhouettes and outlines. SIGGRAPH 99 Course Notes, 1999.

Howlett, S., Hamill, J., & O'Sullivan, C. An experimental approach to predicting saliency for simplified polygonal models. In *APGV '04: Proceedings of the 1st Symposium on Applied Perception in Graphics and Visualization*, pages 57–64, New York, NY, USA, 2004. ACM.

Itti, L., Koch, C., & Niebur, E. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(11):1254–1259, 1998.

Jaimes, A. Posture and activity silhouettes for self-reporting, interruption management, and attentive interfaces. In *IUI '06: Proceedings of the 11th international conference on Intelligent user interfaces*, pages 24–31, New York, NY, USA, 2006. ACM.

Julesz, B. *Dialogues on perception*. Cambridge, Mass. : MIT Press, 1995.

Kadir, T. & Brady, M. Saliency, scale and image description. *Saliency, Scale and Image Description*, 45(21):83–105, Nov. 2001.

Kalnins, R. D., Davidson, P. L., Markosian, L., & Finkelstein, A. Coherent stylized silhouettes. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 856–861, New York, NY, USA, 2003. ACM.

Kim, S. Y., Maciejewski, R., Isenberg, T., Andrews, W. M., Chen, W., Sousa, M. C., & Ebert, D. S. Stippling by example. In *NPAR '09: Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*, pages 41–50, New York, NY, USA, 2009. ACM.

Lavoué, G. A local roughness measure for 3d meshes and its application to visual masking. *ACM Trans. Appl. Percept.*, 5(4):1–23, 2009.

Lee, C. H., Varshney, A., & Jacobs, D. W. Mesh saliency. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 659–666, New York, NY, USA, 2005. ACM.

Lee, S., Kim, G. J., & Choi, S. Real-time tracking of visually attended objects in interactive virtual environments. In *VRST '07: Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology*, pages 29–38, New York, NY, USA, 2007. ACM.

Lee, S., Kim, G. J., & Choi, S. Real-time tracking of visually attended objects in virtual environments and its application to lod. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):6–19, January 2009.

Lewis, J. P., Fong, N., XueXiang, X., Soon, S. H., & Feng, T. More optimal strokes for npr sketching. In *GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 47–50, New York, NY, USA, 2005. ACM.

Li, Z., Ma, L., Jin, X., & Zheng, Z. A new feature-preserving mesh-smoothing algorithm. *Vis. Comput.*, 25(2):139–148, 2009.

Liu, Y.-S., Liu, M., Kihara, D., & Ramani, K. Salient critical points for meshes. In *SPM '07: Proceedings of the 2007 ACM symposium on Solid and physical modeling*, pages 277–282, New York, NY, USA, 2007. ACM.

Longhurst, P., Debattista, K., & Chalmers, A. A gpu based saliency map for high-fidelity selective rendering. In *AFRIGRAPH '06: Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 21–29, New York, NY, USA, 2006. ACM.

Neisser, U. Visual search. *Scientific American*, 210(6):94–102, 1964.

Northrup, J. D. & Markosian, L. Artistic silhouettes: a hybrid approach. In *NPAR '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 31–37, New York, NY, USA, 2000. ACM.

O'Sullivan, C., Howlett, S., McDonnell, R., Morvan, Y., & O'Conor, K. Perceptually adaptive graphics. *The Eurographics Association*, 2004.

Pop, M., Duncan, C., Barequet, G., Goodrich, M., Huang, W., & Kumar, S. Efficient perspective-accurate silhouette computation and applications. In *SCG '01: Proceedings of the seventeenth annual symposium on Computational geometry*, pages 60–68, New York, NY, USA, 2001. ACM.

Rost, R. J. *OpenGL(R) Shading Language (2nd Edition)*. Addison-Wesley Professional, 2005.

Rusinkiewicz, S., Cole, F., DeCarlo, D., & Finkelstein, A. Line drawings from 3d models. In *SIGGRAPH '08: ACM SIGGRAPH 2008 classes*, pages 1–356, New York, NY, USA, 2008. ACM.

Taubin, G. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 902, Washington, DC, USA, 1995. IEEE Computer Society.

Tsotsos, J. K., Culhane, S. M., Winky, W. Y. K., Lai, Y., Davis, N., & Nuflo, F. Modeling visual attention via selective tuning. *Artificial Intelligence*, 78(1-2):507–545, October 1995.

Viola, I., Sousa, M. C., Ebert, D., Andrews, B., Gooch, B., & Tietjen, C. Ieee visualization tutorial on illustrative display and interaction in visualization. The Eurographics Association, 2006.

Zhihong, M., Guo, C., & Mingxi, Z. Robust detection of perceptually salient features on 3d meshes. *Vis. Comput.*, 25(3):289–295, 2009.

Este volume foi tipografado em LaTeX na classe UFPEThesis (www.cin.ufpe.br/~paguso/ufpethesis).