Visualização de Objetos Tridimensionais Baseada em Interpolação Projetiva

ALDO NOGUEIRA[1], ELAINE PRATA[1], LUIZ VELHO[2](ORIENTADOR)

[1]UERJ–Universidade do Estado do Rio de Janeiro - Rua São Francisco Xavier, 524, 20550-900 Rio de Janeiro, RJ, Brasil
{aldo,eprata}@users.codigolivre.org.br
[2]IMPA–Instituto de Matemática Pura e Aplicada - Estrada Dona Castorina, 110, 22460-320 Rio de Janeiro, RJ, Brasil
lvelho@visgraf.impa.br

Abstract. Este artigo descreve a implementação do algoritmo de *View Morphing* descrito na tese de Steven Seitz [1] utilizado na interpolação de vistas na visualização de objetos tridimensionais. Este é o tema do trabalho de final de curso dos alunos Aldo Nogueira e Elaine Prata.

1 Introdução

Computação Gráfica é comumente reconhecida como o processo de geração de imagens através do computador. Nesse artigo preferimos fazer distinção entre as áreas relacionadas na Figura 1 que explicamos a seguir. Para um entendimento melhor sobre o assunto, veja as referências [2, 6].

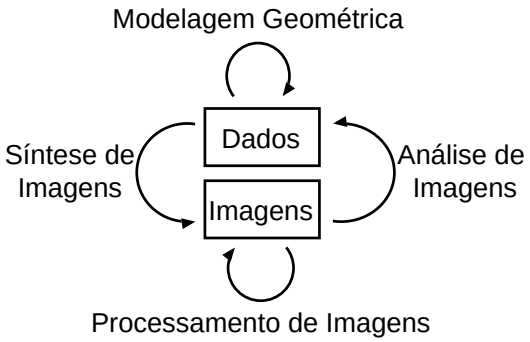

Figura 1: Computação Gráfica.

Síntese de imagens é o processo de obtenção de imagens a partir de dados geométricos. É o caso da indústria cinematográfica, por exemplo, onde utiliza-se o computador para criar cenas que são difíceis ou até impossíveis de se obter por outros meios.

Análise de imagens é o processo inverso ao de síntese. Partimos de imagens obtidas com câmeras reais e extraímos a geometria dos objetos em cena. É também chamado de Visão Computacional. Essas técnicas são empregadas em reconhecimento de terreno em robôs dotados de câmeras, por exemplo.

Processamento de Imagens é a manipulação dentro apenas do domínio das imagens. Correção de cor, superposição de imagens (composição), *warping* (deformação da imagem) e filtros de detecção de contorno são exemplos dessa área.

Modelagem Geométrica é a manipulação das entidades matemáticas que definem os objetos da cena como vértices, arestas e outros. Isso inclui translação, rotação, escalamento e deformações de figuras geométricas, subdivisão poligonal de uma superfície, etc.

As áreas acima não são isoladas entre si e existem técnicas que não se encaixam totalmente dentro de uma delas. O processo utilizado neste trabalho está quase completamente dentro de Processamento de Imagens, que o faz diferente de abordagens anteriores para o problema e traz diversas consequências que pretendemos discutir.

2 Visualização de objetos 3d

O problema que foi a motivação para esse trabalho é o de visualização tridimensional de objetos do mundo real dentro do computador. Queremos ter, por exemplo, uma escultura em madeira na tela do computador. Aqui, tridimensional significa que devemos poder visualizar o objeto a partir de qualquer ângulo, ou de outra forma, girar o objeto em frente à câmera virtual.

Em grande parte dos trabalhos de Computação Gráfica temos um processo onde um artista cria virtualmente ambientes, personagens e objetos usando programas chamados modeladores. Adiciona-se a isso texturas e cores dos componentes da cena, fontes de iluminação e câmeras para assim sintetizar uma foto dessa cena. Esse processo é bastante complexo quando se deseja criar algo com semelhança com a realidade, então, uma técnica que possa simplificar o trabalho em algum ponto é bastante bem vinda.

2.1 Possíveis soluções

O problema enunciado possui um conjunto de soluções até agora estudadas. Vamos citar aqui algumas soluções conhe-

cidas.

A abordagem mais simples do ponto de vista de implementação seria criar um programa que tem como dados de entrada um conjunto de fotos do objeto em posições conhecidas. O usuário interage com este programa informando a posição da câmera virtual. O programa escolhe então a foto no conjunto cuja posição é a mais próxima da câmera especificada pelo usuário.

Outra solução seria, a partir de um conjunto de imagens de entrada, extrair informações tridimensionais sobre o objeto para então construir o modelo tridimensional do mesmo. Com base no modelo obtido, e na especificação da câmera virtual, a imagem desejada é renderizada. Essa técnica utiliza Análise de Imagens seguida de Síntese de Imagens, ou seja, partimos do domínio das imagens para o das formas geométricas para voltar ao das imagens.

Ainda na linha de extração de informações tridimensionais podemos citar a utilização de *scanners* 3D. Esses equipamentos são capazes de extrair informações de objetos tridimensionais do mundo real. No filme "A Era do Gelo" [4], por exemplo, um artista criou uma escultura em argila de um tigre. Essa escultura foi escaneada com um scanner 3D baseado em caneta e braço articulado. O mesmo modelo foi manipulado, esticado, torcido, até criar todos os personagens tigres do filme. Em seguida foram acrescentados esqueletos aos modelos para fazer a animação e por último texturas e iluminação. Na Figura 2 vemos o mesmo processo ser aplicado ao personagem Sid, a preguiça.

Figura 2: Escultura em argila, escaneamento e modelo tridimensional

2.2 Motivos da escolha do algoritmo

Três fatores foram determinantes na escolha do algoritmo utilizado neste trabalho: complexidade, recursos (equipamentos) e qualidade do resultado final.

Vamos analisar brevemente as três soluções citadas na seção 2.1. A solução que apenas seleciona uma foto no conjunto é a mais simples, não exige equipamentos caros ou de difícil acesso porém requer um número muito grande de fotos para gerar uma animação suave. O trabalho para se obter essa grande quantidade de fotos com precisão é muito grande, além do tamanho do arquivo que pode impedir sua distribuição pela Internet. O que queremos então é partir de um número reduzido de fotos e chegar a uma animação bastante suave.

A solução que constrói um modelo poligonal a partir das fotos embora não requeira recursos sofisticados de equipamento é bastante complexa envolvendo problemas como: reconhecimento de contorno, manipulação de objetos poligonais etc. Além disso, uma vez que o modelo é obtido, é necessário gerar a imagem obtida pela câmera especificada. Ou seja, além do trabalho de visão computacional existe o trabalho de síntese de imagens. A síntese de imagens é um processo bastante complexo que pode ser subdividido em questões como: iluminação, recorte, visibilidade etc. A qualidade do resultado final depende dos algoritmos utilizados na solução dos problemas apresentados.

A solução que utiliza scanner 3d requer custo elevado com os equipamentos necessários. Além disso, também requer síntese de imagens, ou seja, a reconstrução da imagem a partir de uma estrutura de dados que, no caso, descreve um objeto 3D.

Este trabalho portanto, tem como proposta, resolver o problema da visualização de objetos tridimensionais no computador através de uma solução que se mantenha no domínio das imagens utilizando equipamentos acessíveis e alcançando com isso um resultado de qualidade.

3 Computação Gráfica Baseada em Imagens

O problema apresentado e que este trabalho tem como objetivo resolver é a visualização de objetos no computador. A partir de um conjunto de imagens do objeto, obter novas vistas do objeto. Este é um dos temas mais recorrentes na área de computação gráfica atualmente.

A abordagem clássica para este problema é obter o modelo tridimensional da cena. Este modelo especifica a geometria da cena e as propriedades das superfícies (como cada superfície interage com a luz - textura). Nesta abordagem, a busca por um resultado com bom grau de realismo resulta em modelos complexos. Isto dificulta a implementação do algoritmo e sua execução requer muito tempo de processamento. Além disso, o tempo de renderização depende da complexidade do modelo.

Uma abordagem alternativa é conhecida como *Image Based Modeling and Rendering* (Modelagem e Síntese Basedas em Imagem). Nesta abordagem, o mundo é modelado como uma coleção de imagens. Veja Figura 3. Para gerar uma imagem arbitrária cada pixel da dessa imagem é mapeado em pixels das imagens originais através de uma determinada função. Esta é na verdade uma aproximação da função plenótica introduzida por Adelson e Berger.

Dado um ponto de visão, a função plenótica descreve o

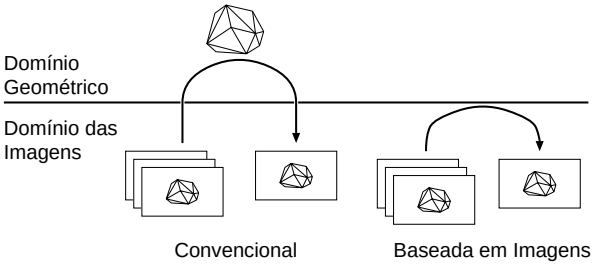

Figura 3: Técnicas e Domínios

fluxo de luz em cada direção do espaço. Uma parametrização possível da função plenótica é $p = p(\theta, \phi, x, y, z)$, onde θ e ϕ são os ângulos de longitude e latitude respectivamente e x, y, e z são as coordenadas euclidianas do ponto de visão O (veja a Figura 4). Outras parametrizações podem incluir tempo (t) e comprimento de onda do raio luminoso (λ).

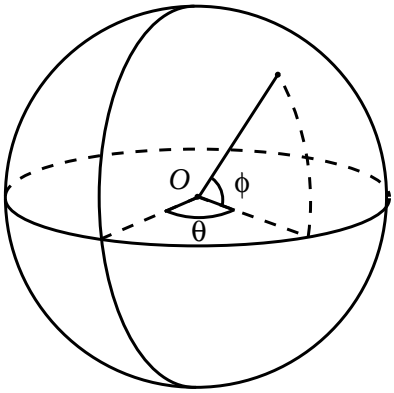

Figura 4: Parametrização da Função Plenótica

A função plenótica é impossível de ser obtida computacionalmente, desta forma, existem várias estratégias usadas para obter diferentes amostragens parciais desta função como descritas no livro [5] (capítulo 16). A abordagem utilizada nesse trabalho possui ponto de visão e direção variáveis representados por uma coleção de imagens para diferentes posições de câmera.

Os métodos para renderização baseados em imagens possuem as seguintes vantagens: realismo (uma vez que as imagens intermediárias são obtidas a partir de um conjunto de imagens de uma cena do mundo real), velocidade (não importa quão complexa é uma cena, o tempo de renderização depende apenas da densidade amostral da imagem). Para uma lista mais detalhada das vantagens que os métodos baseados em imagens possuem veja o artigo [7].

Entre as aplicações da área de Computação Gráfica Baseada em Imagens podemos destacar Realidade Virtual, compressão de vídeo (a taxa de compressão pode ser melhorada guardando-se apenas frames num intervalo regular ao invés de guardar todos os frames, no momento da reprodução os frames intermediários são gerados a partir dos frames que foram armazenados) e vídeo conferência.

4 View Morphing para Visualização de Objetos 3D

O trabalho deste artigo baseou-se principalmente, no que diz respeito a algoritmo, no capítulo 3 da tese de doutorado "Image-Based Transformation of Viewpoint and Scene Appearance" de Steven Seitz [1]. Esta seção começa com uma descrição breve de Morphing e de View Morphing como está na tese de Seitz e em seguida apresenta as diferenças para o nosso trabalho, justificando os motivos.

4.1 Morphing

Tradicionalmente, o algoritmo de Morphing tem como objetivo gerar uma animação transformando um objeto em outro. A entrada é composta de duas imagens, uma lista introduzida pelo usuário mapeando características correspondentes dos objetos nas duas imagens e um valor de 0% a 100% indicando o grau de transformação do primeiro objeto no segundo. O algoritmo aplica um *warping* em cada imagem e então junta-as em uma só com um *blending*.

Na fase de warping, as imagens são distorcidas de forma que as características mapeadas pelo usuário nos dois objetos se aproximem. A posição dessas características é determinada através de uma interpolação linear entre as posições nas imagens originais, estando mais próxima de uma ou de outra de acordo com o grau de transformação. Já na fase de blending, essas duas imagens são combinadas de forma que a cor de um ponto da imagem final pode ser obtida como combinação linear das cores proporcionalmente ao grau de transformação. Veja a Figura 5.

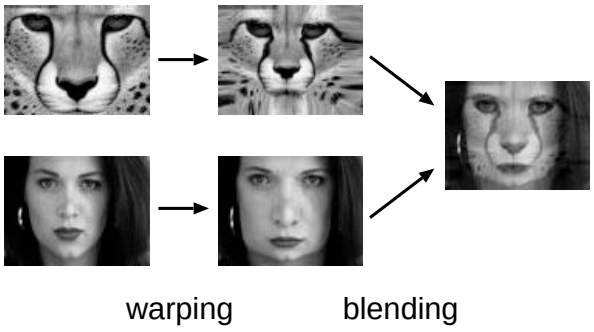

Figura 5: Resumo do Morphing.

Para um texto mais completo sobre as transformações envolvidas no Morphing, veja o livro [5] ou também o paper para a Siggraph '92 de Thaddeus Beier [3].

4.2 View Morphing

O algoritmo de View Morphing usa o já bastante conhecido algoritmo de Morphing como um de seus passos intermediários. Relembrando, o objetivo do algoritmo de View Morphing é, a partir de duas fotos de pontos de vista diferentes de um objeto, obter uma terceira vista arbitrária manipulando as imagens com base nas informações das duas câmeras originais e da final especificada pelo usuário. Como demonstrado na tese de Seitz [1] na seção 3.7, sabemos que um Morphing simples é capaz de gerar as imagens que queremos para alguns casos; aqueles onde as vistas originais são paralelas, nos rendendo como resultado, uma outra vista paralela. O conjunto de casos acima é muito restrito. A idéia do View Morphing é ampliar esse conjunto acrescentando transformações projetivas nas imagens antes e após o Morphing. A sequência é então: pré-warping, morphing e pós-warping.

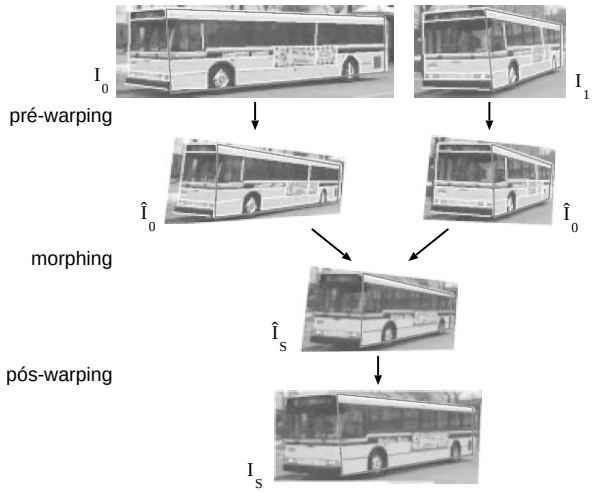

Figura 7: Imagens intermediárias do View Morphing.

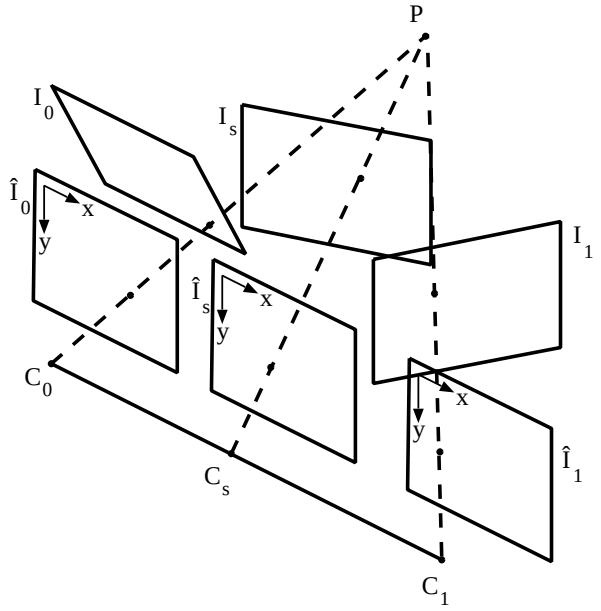

Figura 6: Esquema das vistas.

Na Figura 6 temos uma representação esquemática das transformações que ocorrem no view morphing. Nela I_0 e I_1 representam as imagens originais, $\hat{I}_0$ e $\hat{I}_1$ as imagens intermediárias após o pré-warping, $\hat{I}_s$ o resultado do morphing e, por último, I_s é imagem final. As imagens $\hat{I}$ são vistas paralelas. Os centros óticos das câmeras estão representados por C_0, C_1 e C_s. O esquema da Figura 6 está representado com imagens na Figura 7.

Além dos dados de entrada necessário para o Morphing, precisamos das informações de posição das duas câmeras que geraram as imagens originais e a da câmera final especificada pelo usuário. Durante o pré-warping, reprojetamos as fotos para alterar a direção em que foram obtidas para torná-las paralelas, em seguida aplicamos o morphing que gera como saída outra imagem paralela e por último reprojetamos novamente esse resultado para obter a foto na direção que foi especificada. A reprojeção é explicada, incluíndo as matrizes de transformação, nas seções 3.9.1 e 3.9.2 da tese de Seitz [1].

4.3 Diferenças

No trabalho de Seitz, cada fase gera como saída, imagens que são entrada para a próxima. A vantagem é que podemos utilizar um algoritmo de morphing comum no processo, passando os pontos e as imagens do pré-warping e pegando a imagem resultante para o pós-warping. Essa abordagem de gerar imagens intermediárias implica que temos que escolher a direção das câmeras paralelas do passo de morphing com algum critério, senão podemos obter imagens esticadas demais. Isso poderia ter como consequências o aumento do uso de memória e processador pelo algoritmo e talvez alguma perda de qualidade das imagens.

Além disso, existem casos em que as transformações necessárias para obtenção de vistas paralelas (pré-warping) não são possíveis. Na configuração em que as vistas são paralelas, o centro ótico de uma camera está fora do campo de visão da outra. Uma vista singular ocorre quando o centro ótico de uma das cameras está no campo de visão da outra. Uma vez que as transformações que constituem o pré-warping não realizam alterações no campo de visão, vistas singulares não podem ser reprojetadas para vistas paralelas.

Nesse trabalho, optamos, por motivos didáticos, construir o nosso próprio algoritmo de morphing. Para simpli-

ficá-lo, usamos especificação por pontos e não por *feature* (característica) como descrito no paper [3]. O warping por pontos foi feito usando o conceito de funções com peso inverso à distância (veja na referência [5]). Na verdade a escolha não teve como motivo o próprio morphing, mas desejávamos simplificar as transformações projetivas quando tivéssemos que implementar o view morphing. Cada característica é representada por um segmento orientado e a sua transformação envolve algumas operações a mais do que com um ponto.

O fato de construirmos o morphing facilitou a implementação de View Morphing permitindo o mapeamento inverso (veja na referência [6]) em uma operação única. Para cada pixel da imagem final, aplicamos as transformações (pós-warping^{-1}, morphing^{-1} e pré-warping^{-1} nesta ordem) levando-o nas imagens fonte. Assim, os problemas decorrentes das imagens intermediárias não ocorrem, permitindo a utilização em um número maior de par de imagens, incluindo vistas singulares.

5 Implementação do Algoritmo

5.1 Softwares utilizados

As tecnologias foram escolhidas levando em consideração a facilidade de utilização (ambiente de desenvolvimento gráfico, construtor de interface, etc), licença livre e rapidez na execução. Seguindo esses critérios encontramos diversas opções e no final contou também a familiaridade com as ferramentas. Decidimos por usar o IDE Anjuta e, conseqüentemente, a biblioteca GTK+, o construtor de interfaces Glade e linguagem C e C++.

O Anjuta é um ambiente que integra várias ferramentas de desenvolvimento do GNU/Linux, incluindo o GNU Make, GCC, CVS, Indent, diff, Glade, etc em uma interface gráfica intuitiva. A todo momento temos a liberdade de, se quisermos, deixar o Anjuta e continuar a desenvolver com as ferramentas tradicionais pois na verdade os comandos gráficos apenas manipulam arquivos de configuração como os Makefiles. Isso nos dá flexibilidade e praticidade.

A biblioteca de *widgets* (controles da interface gráfica) GTK+ (Gimp ToolKit) foi criada inicialmente para substituir a biblioteca proprietária Motif no Gimp (Gnu Image Manipulation Program) e é muito popular hoje no mundo do código livre. É usada no ambiente Gnome, OpenOffice.org, Mozilla, etc. Está escrita em C, mas existem diversas conexões (*bindings*) para outras linguagens como C++, Java, Perl, Python, etc, permitindo que uma determinada interface seja escrita na linguagem que mais ofereceça vantagens ao desenvolvedor. A licença é GPL (General Public License). Recentemente a Sun Microsystems adotou o Gnome 2 como interface gráfica padrão do sistema operacional Solaris.

Para fazer os testes do algoritmo, utilizamos o Blue Moon Rendering Tools (BMRT) [9]. É um programa de síntese de imagens compatível com o padrão Renderman [10] estabelecido pela Pixar. Este programa é a única exceção à regra de se utilizar software livre, além disso, na época do começo do trabalho, era distribuído gratuitamente como binário para GNU/Linux, mas agora, depois da compra pela Nvidia, não é mais. Procuramos outras alternativas livres, como por exemplo o Aqsis (Renderman) e o POV-Ray (padrão próprio).

Utilizamos também, as ferramentas CVS para controle de versão e o doxygen para extração automática de documentação a partir do fontes.

5.2 Modelagem

A nossa implementação pode ser dividida em três partes: a biblioteca de Morphing e View Morphing com classes em C++, a interface gráfica com GTK+ e um arquivo misto de C e C++ que liga os eventos gerados da interface a comandos na biblioteca.

Na biblioteca, temos as seguintes classes:

Vector que representa um vetor tridimensional. Aqui estão codificadas as operações (soma, produtos, etc) entre vetores.

Matrix uma matriz 4×4 projetiva. É também responsável por realizar a operação de produto de matrizes, criar transformações projetivas e aplicá-las a um vetor.

Color uma cor no sistema RGB de 24 bits. A operação de blending é codificada aqui.

Image uma imagem composta de uma matriz de cores e suas dimensões. É capaz de ler um arquivo nos formatos Jpeg, Tiff, PNG, entre outros utilizando a bilioteca Imlib.

Camera uma câmera virtual que compreende a posição de seu centro ótico, sua orientação, seu Campo de Visão (*Field Of View*) e as dimensões da imagens produzida por ela. Esta câmera produz as matrizes (Matrix) que correspondem à sua projeção.

Mapping é o mapeamento de pontos correspondentes entre as duas imagens do objeto.

Morphing realiza o processo de morphing entre duas imagens utilizando um mapeamento (Mapping) como base.

ViewMorphing herda de Morphing, acrescentando as funcionalidades para manipulação de câmeras e sobrescreve o método que produz a imagem final levando em consideração as transformações projetivas.

6 Testes, comparações e resultados

Nesta seção, apresentamos resultados que obtivemos com a implementação deste artigo. Utilizamos objetos 3d encontrados como exemplo na própria distribuição binária do BMRT ou em sites relacionados a Renderman.

A sequência de imagens à esquerda foi obtida utilizando-se o programa de síntese de imagens BRMT. A imagem da direita foi feita com o nosso algoritmo a partir da primeira e última imagens da sequência com a câmera virtual na mesma posição da imagem do meio. Veja a Figura 8.

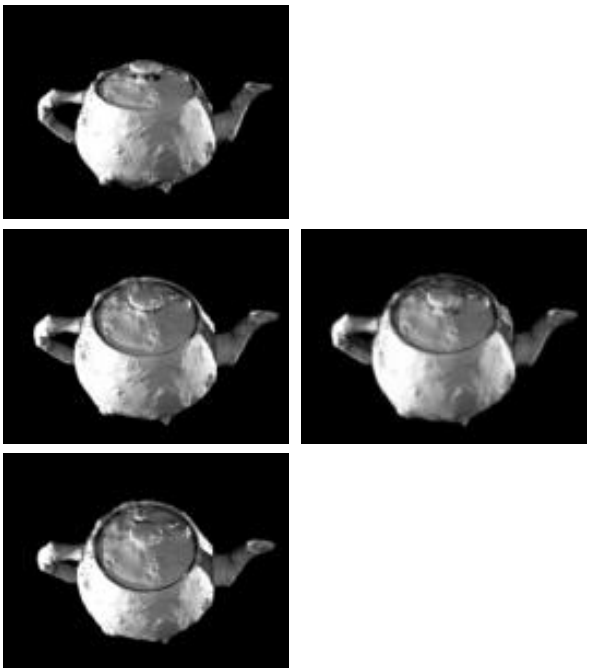

Figura 8: Comparação dos resultados.

Na imagem resultado na figura 8, podemos perceber que há uma perda de definição na região indicada. Isto ocorre porque uma parte do objeto não está visível na primeira imagem, tornando-se visível na segunda. Este problema é conhecido como *hole* e será abordado na seção 8.

Na Figura 9, vemos a interface do programa. Na esquerda temos as fotos originais com os pontos mapeados pelo usuário, na direita, a imagem final para 76.03% de transformação.

Na figura 9 é possível notar uma ondulação no queixo da face. Esta ondulação ocorre como consequência das limitações do algoritmo de warping utilizado neste trabalho – warping especificado por pontos.

7 Conclusão

O trabalho apresentado realiza a implementação do algoritmo de View Morphing proposto por Steven Seitz [1].

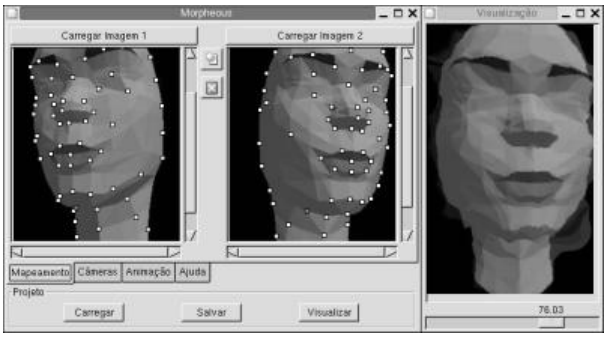

Figura 9: Interface com um outro exemplo rodando.

Este algoritmo realiza o morphing após transformações de câmera que permitem a aplicação do morphing simples para vistas paralelas.

Para tanto, é necessário utilizar imagens de objetos. As imagens utilizadas foram imagens virtuais obtidas a partir de um modelo 3d renderizado a partir das vistas desejadas. Teoricamente imagens reais também podem ser utilizadas se contiverem informações sobre a posição da câmera. Porém, obter essas imagens com tal precisão não é um processo simples. As imagens reais por sua vez poderiam ser obtidas com a utilização de uma câmera fotográfica normal seguida da utilização de um scanner ou com uma câmera digital.

Os próximos passos para o aprimoramento da aplicação desenvolvida são os testes com imagens reais para análise da qualidade obtida e formulação das alterações necessárias. As tendências na área de Renderização Baseada em Imagens são, entre outras: correspondência automática entre as imagens e algoritmos para seleção do conjunto mínimo de imagens que representa a cena como citado em [11].

Além disso, existem várias outros desdobramentos neste mesmo campo de pesquisa. Aqui o objetivo é obter uma vista nova a partir de vistas conhecidas de cenas estáticas. Existem trabalhos em que as novas vistas são obtidas a partir de cenas dinâmicas [12].

O futuro deste trabalho poderá ser acompanhado através do site `morpheous.codigolivre.org.br`, onde em breve teremos o download do código fonte, testes, documentação e etc.

8 Limitações do Algoritmo

Algumas das limitações do algoritmo apresentado estão relacionadas a mudanças na visibilidade do objeto. Mudanças na visibilidade do objeto visualizado podem causar dois tipos de problemas: dobras (*folds*) e buracos (*holes*) [1, 11].

Uma dobra ocorre quando uma superfície visível na imagem inicial se torna invisível na imagem final. Neste caso, vários pixels na imagem inicial são mapeados num

único ponto da imagem final. Existem técnicas que eliminam dobras utilizando informações de profundidade.

Em um buraco, ocorre o oposto. Uma superfície não visível na imagem inicial se torna visível na imagem final. Ao contrário das dobras, buracos não podem ser eliminados sempre apenas com informações adicionais da imagem. Soluções alternativas incluem definir uma cor padrão para o fundo da imagem, interpolação entre os pixels vizinhos ou adicionar imagens para uma melhor amostragem.

9 Aplicações

Durante as pesquisas e o desenvolvimento deste projeto, tomamos conhecimento de algumas aplicações possíveis para os algoritmos aqui utilizados.

O algoritmo de morphing simples que é utilizado quando as imagens têm vistas paralelas tem aplicação, por exemplo, em investigações de pessoas desaparecidas. A foto da pessoa desaparecida pode ser confrontada com a de parentes (comumente os pais) mais velhos para se ter uma idéia da aparência da pessoa após alguns anos. Este tipo de procedimento muitas vezes é veiculado pela televisão. Esta mesma técnica foi utilizada no famoso clipe de Michael Jackson "Black or White" também citado em [3].

O view morphing pode ser utilizado na visualização de objetos 3d em várias situações: visualização de peças, aplicativos didáticos de geometria e também na indústria de entretenimento. Recentemente, os filmes Matrix e Matrix - Reloaded utilizaram esta técnica [15]. O View Morphing também tem aplicação em simulações e Realidade Virtual, na representação de ambientes virtuais através de coleções de imagens e interpolação entre as mesmas para gerar novas imagens.

Referências

[1] Steven Maxwell Seitz, *Image-Based Transformation of Viewpoint and Scene Appearance*, University of Wisconsin - Madison (1997).

[2] Jonas Gomes e Luiz Velho, *Computação Gráfica, Volume 1*, Série de Computação e Matemática (IMPA-SBM) (1998).

[3] Thaddeus Beier, *Feature-Based Image Metamorphosis*, Silicon Graphics Computer Systems (1992).

[4] Chris Wedge, Carlos Saldanha, *The Making of Ice Age*, Fox Home Entertainment, Blue Sky (2002)

[5] Jonas Gomes, Lucia Darsa, Bruno Costa, Luiz Velho, *Warping and Morphing of Graphical Objects*, San Francisco: Morgan Kaufmann (1998).

[6] Jonas Gomes, Luiz Velho, *Computação Gráfica: Imagem*, Série de Computação e Matemática (IMPA-SBM) (1994).

[7] Andrea Fusiello, *Image Based Rendering*, Università degli Studi di Verona (2003).

[8] Bruce Eckel, *Thinking in Patterns - Problem-Solving Techniques using Java*, MindView, Inc. (2003).

[9] *Blue Moon Rendering Tools User Manual- release 2.6*, Exluna, Inc. (2000).

[10] *The RenderMan Interface Version 3.2*, Pixar, Inc. (2000).

[11] Shenchang Eric Chen, Lance Williams, *View Interpolation for Image Synthesis*, Apple Computer, Inc. (1993).

[12] Jiangjian Xiao, Cen Rao, Mubarak Shah, *View Interpolation for Dynamic Scenes*, University of Central Florida (2002).

[13] Havoc Pennington, *GTK+/Gnome Application Development*, Que Publishing (1999).

[14] Karl Fogel, *Open Source Development with CVS* (capítulos livres), The Coriolis Group (2000).

[15] *Bullet Time Walk Through* (`whatisthematrix.warnerbros.com`), Warner Bros. (1999)

[16] Stan Birchfield, *An Introduction to Projective Geometry (for computer vision)*, (1998)