Instituto de Matemática e Estatística - Universidade de São Paulo

MAC 499 – Trabalho de Formatura Supervisionado

Reconstrução e Síntese de Cenários Tridimensionais a partir de Imagens Estereoscópicas

Alunos: Daniel Ferreira Santos
Eduardo Bretones Mascarenhas Apolinário

Orientador: Marcel Jackowski
30 de Novembro de 2009

Conteúdo

Lista de Figuras

1 Introdução

1.1 Estereoscopia

Estereoscopia é o fenômeno que ocorre naturalmente quando observamos uma cena com nossos olhos. Nossos olhos observam a mesma cena, porém através de ângulos ligeiramente diferentes, resultando em uma imagem diferente para cada olho. Nosso cérebro, a partir dessas duas imagens, consegue estimar a distância de cada objeto observado em decorrência das pequenas diferenças em cada imagem. Desta forma temos a noção de profundidade dos objetos que observamos.

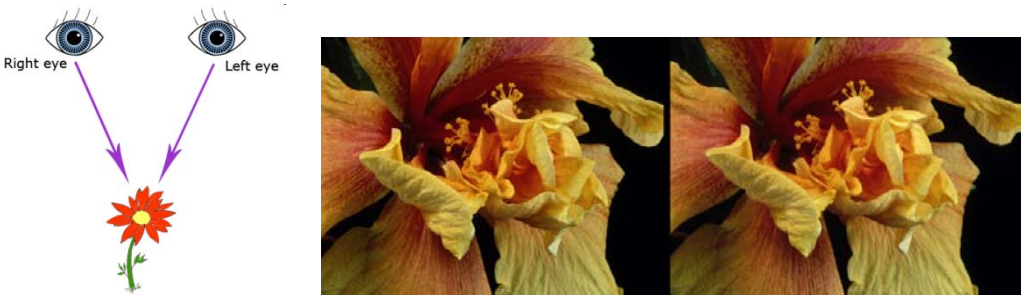


Figura 1: Estereoscopia

1.2 Aplicação no trabalho

O conceito de estereoscopia no sistema visual humano é aplicado de forma análoga no trabalho. Os elementos das imagens são reconhecidos e o software faz as análises de distanciamento entre os pontos, os cálculos para estimar a profundidade dos elementos e a criação do cenário tridimensional.

A ferramenta adicional existente no software é a inclusão no cenário criado de objetos 3D, de forma que o objeto se comporte levando em consideração os objetos já existentes na cena e a iluminação ambiente ou até mesmo uma iluminação artificial criada pelo programa.

Nosso objetivo é fazer com que todo o cenário resultante fique o mais próximo possível da realidade.

1.3 Motivação

Nosso interesse pela computação gráfica, juntamente com a tarefa de simulação de ambientes reais com elementos virtuais, despertou a vontade de trabalhar com o tema de imagens estereoscópicas e lidar com o resultado tridimensional de suas manipulações. Assunto esse já trabalhado por outras pessoas, gostaríamos de desenvolver um software próprio com o máximo de funções, mas considerando o tempo limitado que tivemos para a realização desse trabalho de considerável complexidade e grande dedicação.

1.4 Aplicações do software

A proposta do projeto é possibilitar a utilização do software em tarefas de simulação ambientes, intervenção virtual em projetos de design de interiores ou arquitetônicos e também cenários de efeitos especiais para filmes.

2 Objetivos

O trabalho tem como objetivo a construção de um software em que a partir de imagens estereoscópicas irá construir um cenário tridimensional onde será possível sintetizar objetos no interior da cena.

2.1 O Funcionamento do Software

O funcionamento do programa se dará da seguinte maneira: de posse de duas imagens estereoscópicas, a partir da análise de pontos determinados na imagem, programa irá determinar a posição tridimensional dos mesmos em relação à câmera.

A partir do conhecimento desses dados é construído um modelo tridimensional do ambiente em questão. As imagens originais são utilizadas como textura deste modelo, aumentando a fidelidade e o realismo do modelo 3D.

Concluída a modelagem do cenário tridimensional, o usuário poderá interagir com o ambiente sintetizando novos elementos na cena, esses elementos se moveriam na cena projetando sombras e iluminações, sendo capazes de colidir levando em conta todos os elementos identificados na cena.

3 Pesquisa

Ao longo do desenvolvimento do trabalho foram pesquisados diversos papers, livros e site. Certamente a tarefa de pesquisa foi essencial para atingirmos o resultado final do nosso trabalho, através dela pudemos aprender diversas técnicas que nos possibilitaram a criação de um software eficiente e com funcionalidades.

Todos os papers e livros estudados e sites visitados estão citados ao longo da monografia, na parte em que está especificado como foi feito o desenvolvimento do software, assim como uma lista de referencias ao final.

4 Ferramentas

A construção do software foi feita na linguagem C#, para isto utilizamos o software gratuito Microsoft Visual C# 2008 Express Edition. Para a simulação dos ambientes 3D utilizamos o CSGL, uma biblioteca OPENGL[11] para C#. A biblioteca dos modelos 3D que podem ser inseridos na cena foi criada utilizando o Software Autodesk 3D Studio Max e exportados para um formato que pode ser lido pelo software desenvolvido.

5 Etapas do desenvolvimento do Projeto

5.1 Algoritmo de reconhecimento estéreo através de cores:

Esse algoritmo reconhece os elementos a partir das cores, ou seja, se uma imagem contém uma bola de cor vermelha numa certa posição, o trabalho do algoritmo é reconhecer ambas as bolas vermelhas que estão nas duas imagens em posições diferentes, de forma automática. Este algoritmo é baseado na idéia do paper STEREO MATCHING WITH COLOR-WEIGHTED CORRELATION, HIERARCHICAL BELIEF PROPAGATION AND OCCLUSION HANDLING[1].

5.1.1 O funcionamento do algoritmo

O software discretiza a imagens em áreas de acordo com a intensidade das cores. Em seguida compara as posições relativas das áreas de cada intensidade e gera um mapa de diferenças. Então utiliza o mapa de diferenças para criar o mapa de disparidades, o programa parte do princípio de que quão maior a diferença da posição um objeto entre uma imagem e outra, mais longe (ou perto, se a diferença for negativa) este se encontra da câmera.

5.1.2 Discretização das imagens

O programa mostrou apenas resultados satisfatórios com uma discretização da imagem em 8 ou mais faixas de intensidade (slices) para cada canal de cor (Red, Green e Blue), o que torna a computação dos resultados custosa.

Os slices são a discretizaçao da intensidade da cor dos canais, por exemplo, a intensidade de azul que pode ir de 0% a 100% , é discretizada em n faixas.Uma discretização em 4 faixas resultaria em [0%, 25%], [25%,50%], [50%,75%], [75%,100%].

Os steps são a quantidade de vezes que a imagem é processada, alterando o lugar onde a imagem é discretizada, O resultado final é a média de todos os steps.

5.1.3 Interpretação dos elementos

Identificamos também que o programa interpreta objetos com partes escondidas atrás de outros como se ele só existisse onde estivesse visível na imagem, interferindo, assim, no cálculo da diferença das posições do objeto nas duas imagens estereoscópicas.

Outro problema encontrado foi que áreas onde a diferença de tonalidade se torna baixa, como sombras, penumbras ou áreas brilhosas, o programa não discretiza corretamente, considerando todas estas áreas como um só objeto, interferindo também no cálculo das distâncias.

5.1.4 Resultados

Simulação 1: Cubos e Cilindros

O caso ideal de imagem seriam todos os objetos separados e com cores bem diferentes.

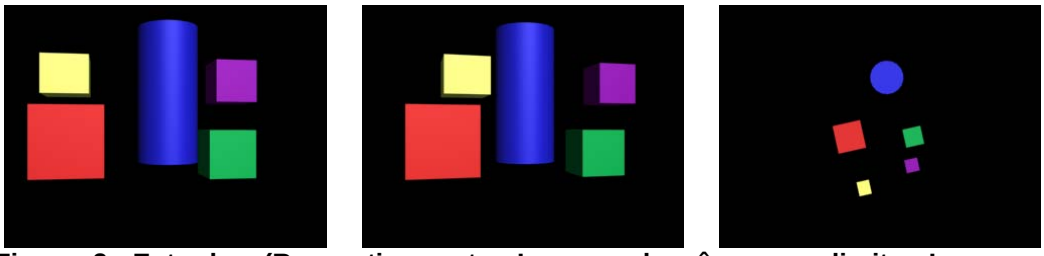


Figura 2: Entradas (Respectivamente: Imagem da câmera a direita; Imagem da câmera a esquerda; Disposição dos objetos)

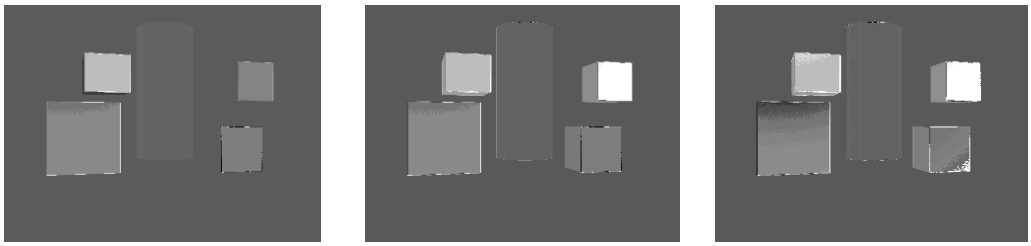


Figura 3: Saídas (Respectivamente: 4 Slices, 4 Steps; 8 Slices, 4 Steps; 16 Slices, 4 Steps)

Simulação 2: Bule de Chá

Neste caso podemos ver algumas dificuldades para o algoritmo expressar um bom resultado da disposição dos objetos na imagem.

Figura 4: Entradas (Respectivamente: Imagem da câmera a direita; Imagem da câmera a esquerda)

Figura 5: Saídas (Respectivamente: 4 Slices, 4 Steps; 8 Slices, 4 Steps; 16 Slices, 4 Steps)

Como podemos observar, não é possível encontrar a distância do background.

Com 4 slices o programa não mostra bom desempenho, apenas figuras bem simples. Para 16 slices o programa começa a reconhecer pequenos detalhes na foto erroneamente. A melhor faixa de resultado foi obtida com 8 slices.

Simulação 3: Fotografia

Com foto o programa não mostrou bons resultados.

Figura 6: Entradas (Respectivamente: Imagem da câmera a direita; Imagem da câmera a esquerda)

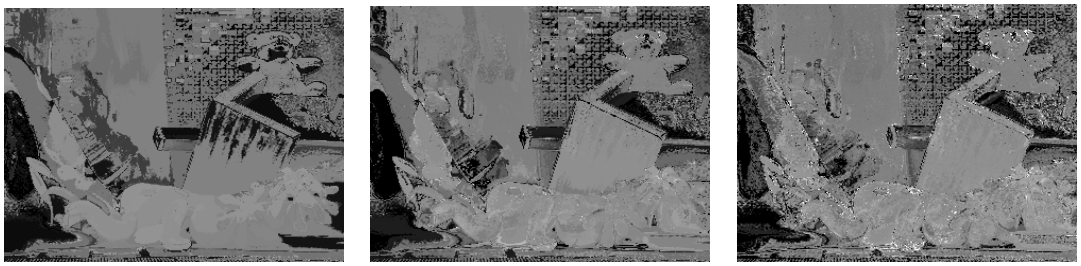


Figura 7: Saídas (Respectivamente: 4 Slices, 4 Steps; 8 Slices, 4 Steps; 16 Slices, 4 Steps)

5.1.5 Aprimoramento do algoritmo de reconhecimento por cores

Vista as limitações que encontraríamos com o algoritmo de reconhecimento estéreo de profundidade dos objetos através de cores, partimos para uma segmentação por linhas.

Este novo algoritmo trabalha da mesma forma que o citado anteriormente, contudo este separa a imagem em várias faixas horizontais e as trata separadamente. Em seguida junta as faixas, formando a imagem total.

Este método nos traz melhores resultados, pois como a área de cálculo é menor, o processamento torna-se mais rápido e preciso.

5.1.6 Resultados

Não foi possível seguir esta abordagem, pois os resultados obtidos não atendiam nossas expectativas. O algoritmo tem um bom funcionamento com imagens simples e mau funcionamento com imagens complexas.

Para ambientes internos, objetivo do nosso trabalho, por ser um ambiente complexo, o algoritmo não resultou em um bom desempenho. Por exemplo, a diferenciação sutil de cores era uma limitação vista nos testes feitos com o algoritmo, sendo assim partimos para outra abordagem.

5.2 Pesquisa de algoritmos de terceiros

Foram pesquisados algoritmos que satisfizessem parte do nosso objetivo no reconhecimento de padrões em imagens densas, os quais seriam personalizados de acordo com nossa necessidade de modo a realizar a correspondência estereoscópica e obtenção da profundidade dos elementos.

5.2.1 Lista e comentários das pesquisas

- MIDDLEBURY, VISION[2]

Esse site contém uma listagem de códigos e comparações entre eles, no entanto os códigos são mal documentados.
O site foi uma ótima fonte de pesquisa de papers.

- SEGMENT-BASED STEREO MATCHING USING BELIEF PROPAGATION AND A SELF-ADAPTING DISSIMILARITY MEASURE[3]

Este paper apresenta resultados muito bons. Contudo é muito complexo e o entendimento do funcionamento do algoritmo necessita de conhecimento e domínio de técnicas avançadas de reconhecimento estereoscópico.

- A REGION BASED STEREO MATCHING ALGORITHM USING COOPERATIVE OPTIMIZATION[4]

Foi inspiração para a criação de reconhecimento estéreo baseado em cores, citado anteriormente.

Este método separa as imagens em regiões distintas e correlaciona essas regiões com base nas suas características de forma, tamanho e cor.

- STEREO MATCHING WITH COLOR-WEIGHTED CORRELATION, HIERARCHICAL BELIEF PROPAGATION AND OCCLUSION HANDLING[1]

Este paper descreve um algoritmo que também utiliza o conceito de separação por cores. Após o reconhecimento é feito uma série de cálculos para o refinamento do resultado.

O algoritmo é muito complexo e específico para nossas necessidades, sendo difícil de entender os conceitos. Tentamos entrar em contato com os autores e todos eles já haviam abandonado o projeto.

- MACHINE VISION [5]

Capítulo 11:

Neste capítulo é explicado como é feito o reconhecimento de pontos de interesse e a correlação estereoscópica destes pontos. É descrito um algoritmo que identifica pontos relevantes na imagem, ou seja, pontos que tem algo incomum em uma imagem, e tenta encontrar esses mesmos pontos na outra imagem.

Resultado das pesquisas

Apesar de não utilizarmos algoritmos de terceiros a pesquisa foi extremamente fundamental, pois adicionou novos conceitos para a resolução dos nossos problemas.

5.2.2 Mudança de estratégia

Como o foco do nosso trabalho está voltado para a construção de cenários tridimensionais e a síntese de objetos nesse cenário, optamos depois de muita pesquisa, desenvolvimento e testes, por não fazer o reconhecimento automático dos elementos das imagens estereoscópicas. O que foi feito foi o desenvolvimento de uma interface para realizar a correspondência estereoscópica de forma manual.

5.3 Inserção manual de pontos para correspondência estereoscópica.

Criamos uma interface gráfica que faz a correspondência dos pontos inseridos pelo usuário na imagem.

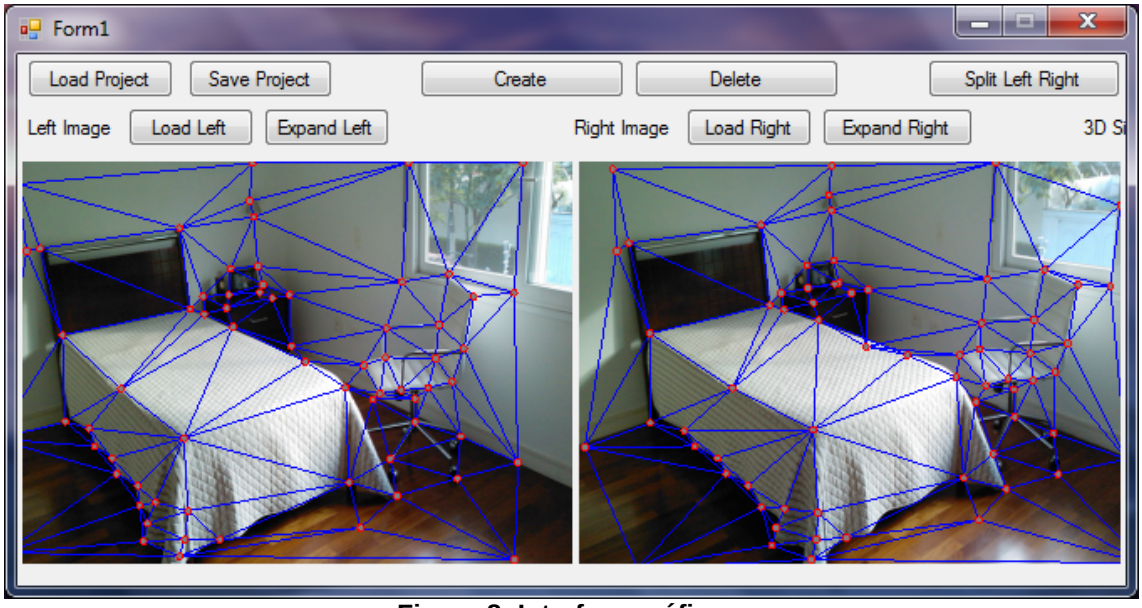


Figura 8: Interface gráfica

A interface para criação dos pontos é dividida em duas áreas, uma para a imagem esquerda, outra para a direita.

A criação desses pontos inicia quando o usuário entra no modo de criação e clica na imagem para criar um ponto na posição desejada. O usuário pode mover os pontos clicando e arrastando os pontos desejados. O ponto selecionado é automaticamente pintado de branco nas duas imagens, para facilitar o usuário identificar qual é o ponto correspondente na outra imagem.

Além disso, a interface possui várias ferramentas como zoom nas imagens, abertura de arquivo, salvamento do resultado, divisão da tela em dois sempre visando a facilitação da usabilidade do software pelo usuário.

5.3.1 Explicação das ferramentas

Load Project	Carrega e salva os pontos inseridos nas duas imagens e o endereço do arquivo das imagens esquerda e direita
Save Project	
Create	Entra e sai do modo de Exclusão dos pontos. Onde clicar na imagem um ponto é criado na mesma posição nas duas imagens.
Delete	Entra e sai do modo de Exclusão dos pontos. O ponto clicado é excluído.
Load Left	Abre a janela para carregar o arquivo referente à imagem esquerda e direita.
Load Right	
Expand Left	Maximiza a imagem esquerda ou direita para que o usuário tenha mais controle para criar e posicionar os pontos
Expand Right	
Split 2	Maximiza as imagens de forma lado a lado, para poder editar e comparar a posição dos pontos nas duas imagens.

5.3.2 Possibilidade

A estrutura do software foi programada para permitir a inserção de um módulo de correspondência automática, sem necessidade de alterar o programa.

5.4 Triangulação

Nessa etapa fizemos a triangulação dos pontos segundo o algoritmo de triangulação de Delaunay. O resultado da triangulação é mostrado em tempo real, concomitantemente à inserção ou edição dos pontos pelo usuário.

A triangulação será usada posteriormente para criação da malha 3D.

5.4.1 Como funciona a Triangulação de Delaunay

A triangulação de Delaunay consiste em unir um conjunto de pontos formando triângulos de forma a minimizar o tamanho das arestas e maximizar os ângulos de cada triangulo. Esta triangulação possui ótimos resultados para criação superfícies.

Para criar uma triangulação de Delaunay, é tomada a sua distribuição de dados encontrando todos os triângulos definidos por três pontos da

distribuição, resultando em um círculo que passa por estes três pontos, mas que não inclua nenhum outro ponto.

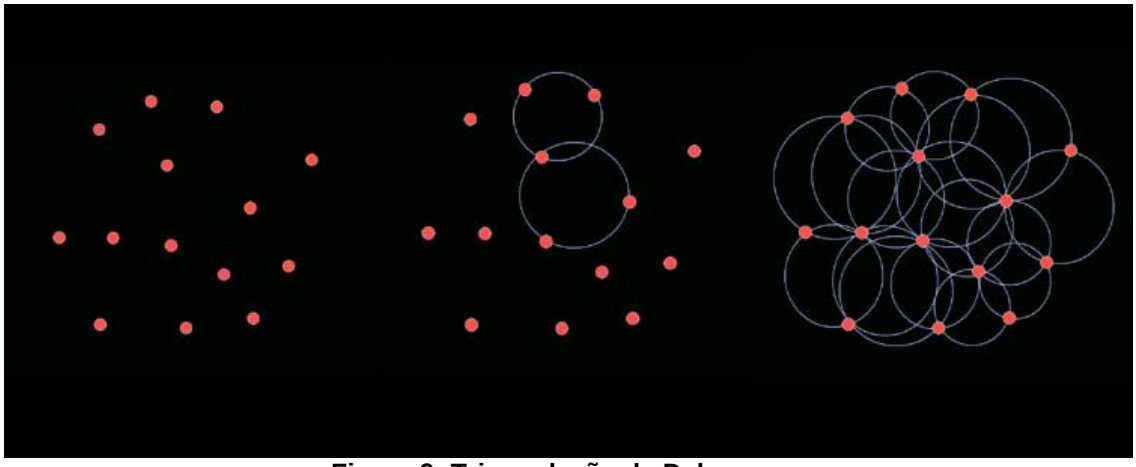


Figura 9: Triangulação de Delaunay

Para cada conjunto de três pontos satisfazendo a condição de Delaunay encontrado, gere um triângulo.

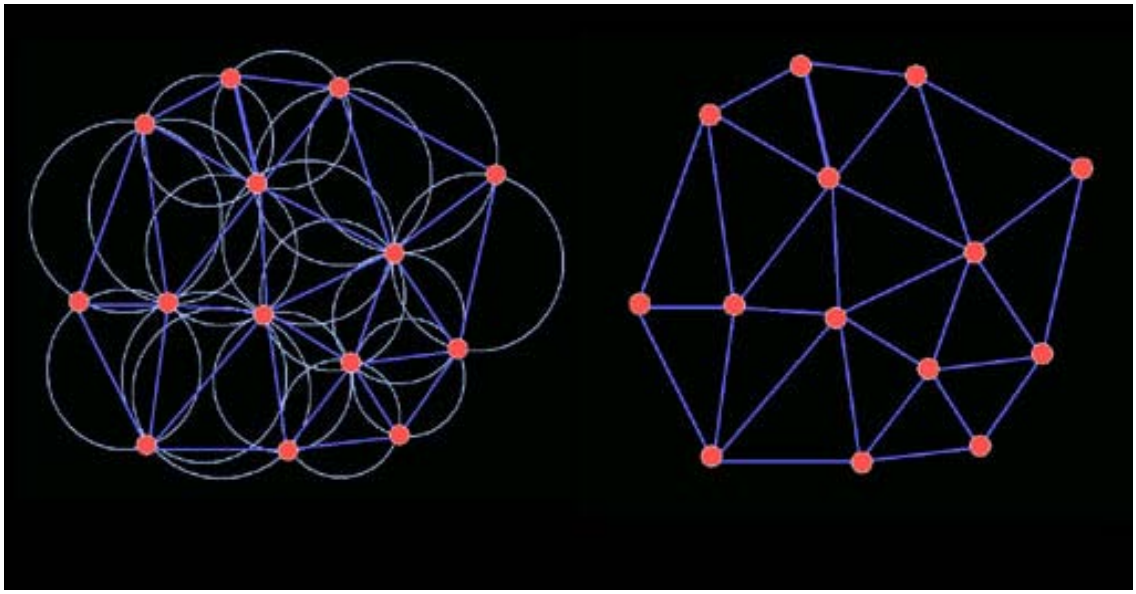


Figura 10: Triangulação de Delaunay

Este método possui ótimos resultados para criação de terrenos, ou no nosso caso, um cenário tridimensional.

Primeiramente os pontos são triangulados na superfície z = 0, logo depois estes pontos são elevados à sua própria altura. O resultado é uma malha tridimensional onde cada ponto está na altura certa e as regiões dos triângulos possuem uma superfície que consiste na interpolação dos pontos, conforme a figura.

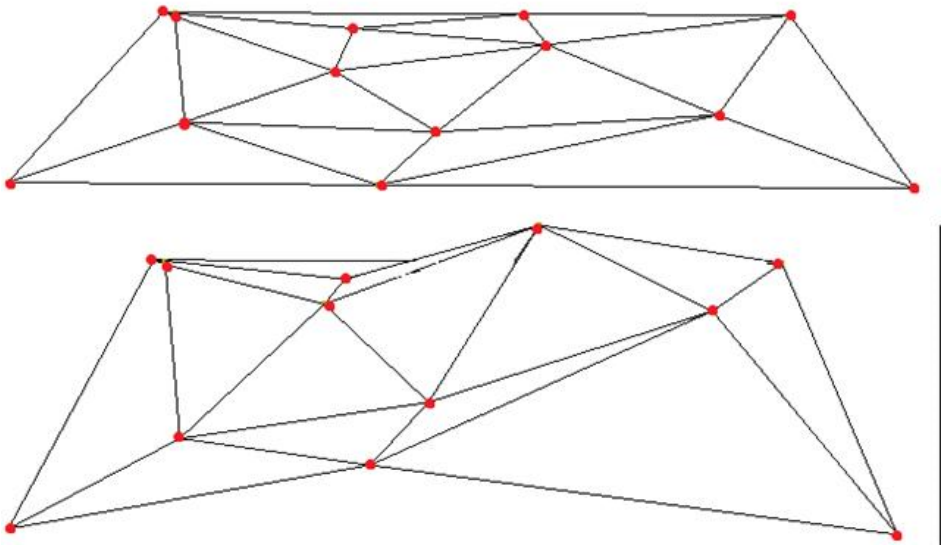


Figura 11: triangulação utilizada para criar malhas 3D

O algoritmo da triangulação foi usada do projeto aberto DELAUNAY TRIANGULATION IN .NET 2.0 BY MORTEN NIELSEN [7].

A biblioteca foi devidamente modificada para integrar o programa já construído. A estrutura de pontos da biblioteca continha apenas as coordenadas x e y, sendo assim foram adicionados as coordenadas x1 e y1, correspondendo respectivamente às coordenadas x e y da outra imagem associada, e também o eixo z, para guardar a informação profundidade de cada ponto.

5.5 Criação da malha tridimensional

A malha tridimensional é criada a partir do mapa de profundidade, para isso é feita uma estimação dos pontos 3D utilizando as informações obtidas. Após isso é feito o cálculo da profundidade de cada ponto.

5.5.1 Cálculo da profundidade

Criamos uma função que transforma a quíntupla (x, y, z, x_1, y_1) para (x, y, z) através da seguinte formula: $d = (x - x1)$

Quanto mais discrepante a posição dos pontos nas duas imagens maior a distância d.

Depois normalizamos o d em [0,1], então as coordenadas tridimensionais (x, y, z) são obtidas da seguinte forma:

$(x, y, z) = (\alpha x(d^2 + 1), \alpha y(d^2 + 1), d^2)$

onde α é o ângulo de abertura da câmera.

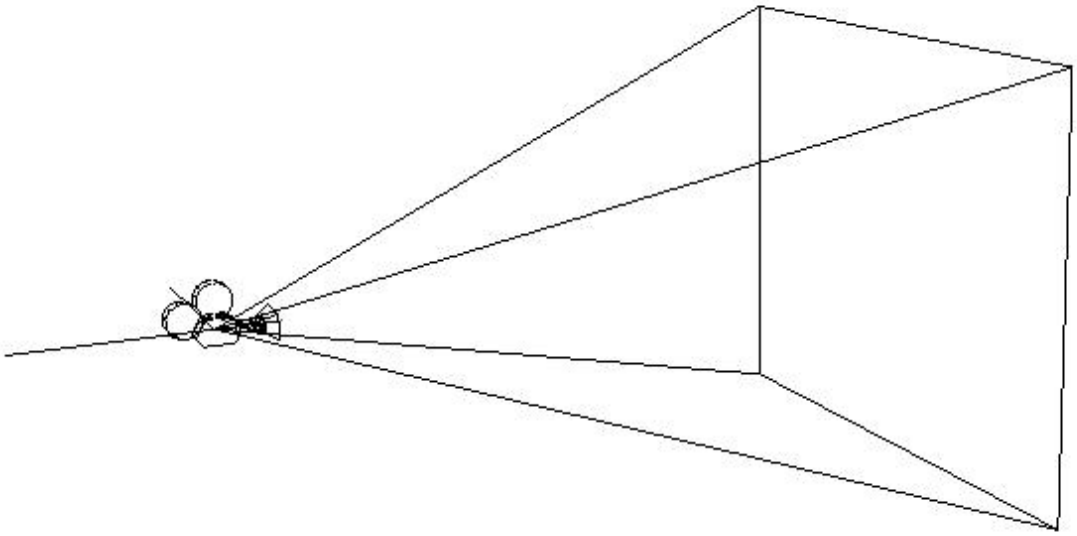


Figura 12: Ângulo de abertura da câmera

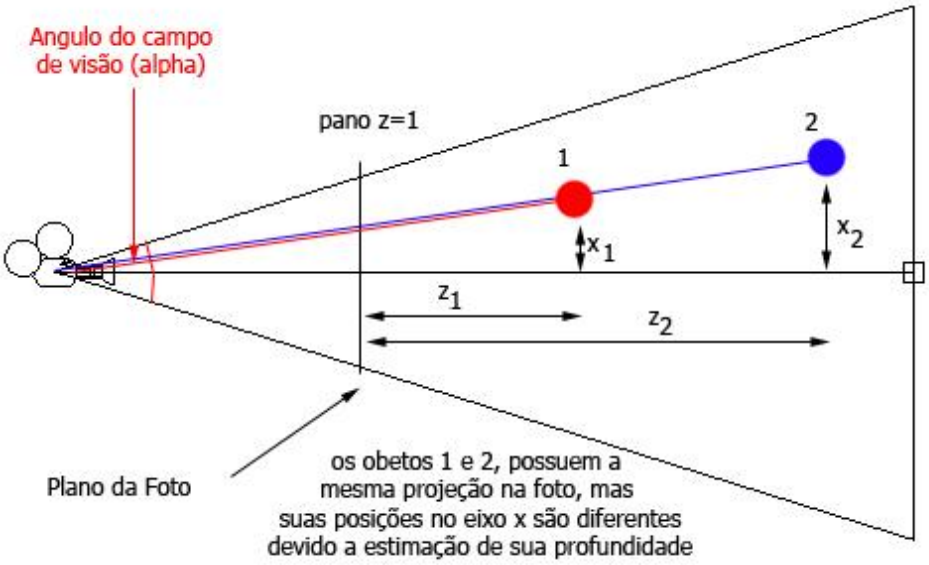


Figura 13: Esquema do cálculo de profundidade

5.6 Exibição do ambiente 3D criado utilizando a ferramenta OpenGL

Foi utilizada a biblioteca aberta CSGL[8], que fornece a integração do C# com o OpenGL.

O ambiente 3D foi criado com a união dos triângulos obtidos na etapa de triangulação e os pontos posicionados conforme o resultado da função de cálculo de profundidade. Em seguida foi aplicada a textura da foto conforme as

coordenadas da imagem esquerda. Assim, os triângulos foram desenhados conforme a imagem da foto, fornecendo realismo à cena.

5.6.1 Pesquisa de sombra – OpenGL

A fim de representarmos a sombra dos objetos inseridos no ambiente 3D foi preciso muita pesquisa, pois depois de muito trabalho e desenvolvimento vimos que na seria uma funcionalidade fácil de ser implementada.

Um livro muito bom que utilizamos na pesquisa foi o COMPUTER GRAPHICS USING OPENGL[6], com ele foi possível fazer sombra projetada. A sombra era projetada desenhando-se o objeto em uma projeção a partir do ponto de luz em um plano.

No entanto, o que tínhamos não era muito útil, pois não queríamos projetar sombra em apenas um plano, mas sim em todo o cenário que por sua vez é composto de diversas faces e ângulos diferentes.

Sendo assim, pesquisamos um novo método de criação de sombra utilizando shadow volumes. Aprendemos esse novo método no NEHE[9] que é um portal que reúne informações para desenvolvedores em OpenGL.

5.6.2 Interface gráfica para controles do cenário tridimensional

Foi criada uma interface gráfica para facilitar a visualização do resultado tridimensional

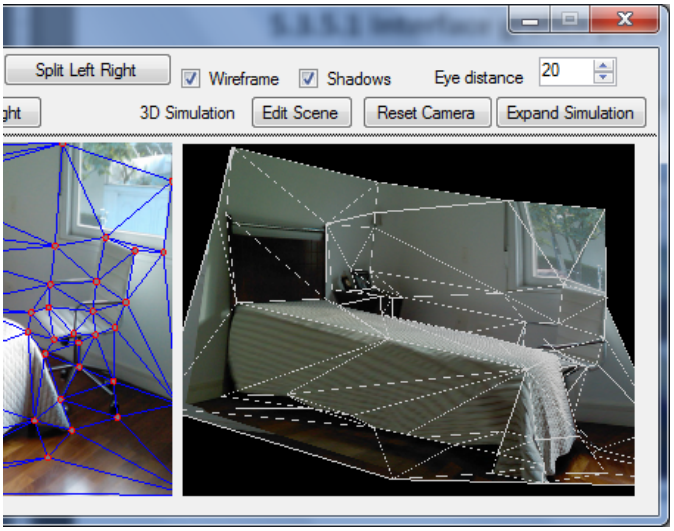


Figura 14: Resultado tridimensional

Foram adicionados controles de movimentação e rotação da câmera com o mouse, funcionando da seguinte maneira:

- Botão esquerdo e movimentação do mouse: translação da câmera (pan).
- Botão direito e movimentação do mouse: rotação da câmera em torno do objeto.

Wireframe	Liga e desliga a exibição de wireframe (linhas brancas no objeto 3d)
Shadows	Liga e desliga as sombras causadas por objetos inseridos no cenário
Eye Distance	Configura a distancia que as duas fotos foram tiradas. No caso, uma distancia maior intensifica a profundidade dos elementos
Edit Scene	Abre a caixa de diálogo para inserção de objetos na cena
Reset Camera	Volta a posição da câmera para a inicial.
Expand Simulation	Maximiza a área da simulação.

5.7 Inserção de objetos

É possível inserir objetos no cenário tridimensional criado, sendo que a sua inserção é fiel à cena, conservando as propriedades de luminosidade e levando em consideração os objetos já contidos no cenário.

5.7.1 Interface gráfica para inserção e Edição de objetos

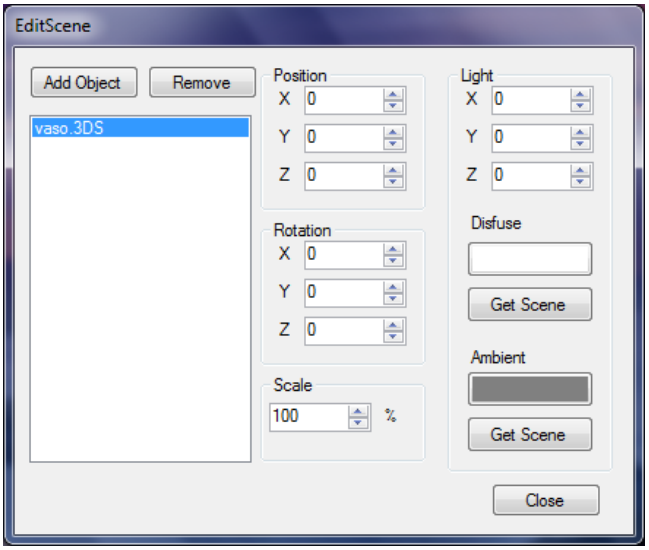


Figura 15: Interface gráfica de edição

A interface serve para inserir e controlar todos os parâmetros de cada objeto inserido e da luz do ambiente. A luz do ambiente interage com a foto original projetando sombras dos objetos na mesma.

- Lista de controles

Add object	Abre a janela para escolher o arquivo do objeto a ser inserido
Remove	Remove o objeto selecionado na lista
Position x,y,z	Exibe e altera os atributos da posição do objeto
Rotation x,y,z	Exibe e altera os atributos de rotação do objeto
Scale	Exibe e altera o tamanho do objeto
Light x,y,z	Exibe e altera a posição x,y,z do ponto de luz
Difuse (botão)	Exibe a cor e Abre a caixa de escolha da cor da luz direta
Ambient (botão)	Exibe a cor e Abre a caixa de escolha da cor da luz ambiente
Difuse, ambient, Get Scene	Verifica a iluminação da cena e modifica a cor da iluminação para se adequar a mesma da cena.
Close	Fecha a janela de edição

5.7.2 Formato de arquivo dos objetos

O formato 3ds é o formato mais popular de modelos 3D. Não é o mais completo atualmente, contudo sua popularidade cresceu durante o final da década de 1990, em conseqüência da popularidade do software 3D Studio. Dessa forma tornou-se um formato muito bem documentado e com muitos leitores em diversas linguagens de programação.

Sendo assim, esse foi o formato escolhido, pois é bastante conhecido e já tínhamos familiaridade com os softwares que geram o arquivo.

5.7.3 Interação do arquivo 3ds com o software

A fim de que o objeto 3D criado pudesse ser inserido no cenário, foi utilizado um loader de arquivo de modelos no formato 3ds (3d studio). O loader utilizado foi o SALMONREADER[9], um código independente escrito por Scott Ellington.

O código original utilizava a biblioteca TAO, uma biblioteca cross plataform de engine gráfica (OpenGL, SDL, etc). O código foi modificado para trabalhar em conjunto com a biblioteca CSGL utilizada no programa, ao invés da TAO.

Dessa forma foi possível programar a síntese de objetos na cena, carregando o arquivo utilizando o loader implementado.

A classe dos modelos inseridos ("Model") também foi modificada para armazenar alguns atributos de controle do objeto como posição, rotação, escala e nome.

5.8 Iluminação e sombras

Essa parte do desenvolvimento do software foi realmente muito complicada, ao final os objetos vão interagir com a iluminação do cenário projetando sombras.

Para isso foi escolhido o gerador de sombra do tipo stencil, em que o OpenGL determina uma região onde a sombra será projetada e escurece esta área. Para isso era necessário criar um sólido do objeto que é chamado shadow volume, ou seja, volume de sombra do objeto.

A sombra é obtida criando a intersecção do volume de sombra com o plano onde a sombra é projetada

Para criar o volume de sombra, é preciso identificar a silhueta do objeto visto pela luz, e assim criar um objeto que é a prolongação de todos estes vértices da silhueta da direção da luz até as faces onde a sombra será projetada. Por exemplo, o volume de sombra de uma bola é um cone o de um quadrado é uma pirâmide:

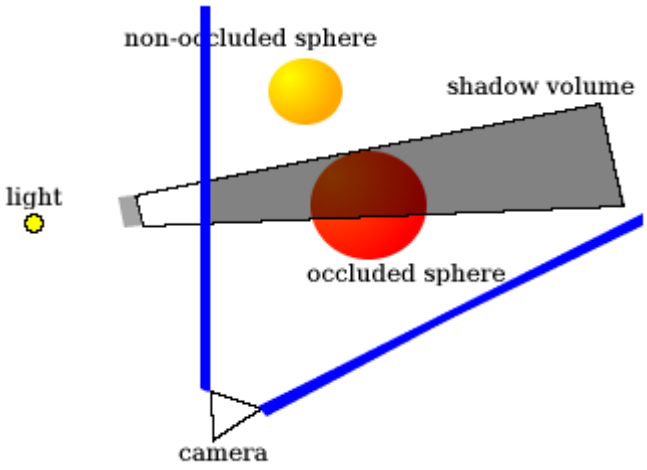


Figura 16: Volume de sombra

Para isso são desenhadas todas as faces do solido de luz voltada para a câmera. Após isso são subtraídas apenas as faces que estão do lado oposto a visão da câmera, restando apenas a projeção no plano que recebe a sombra.

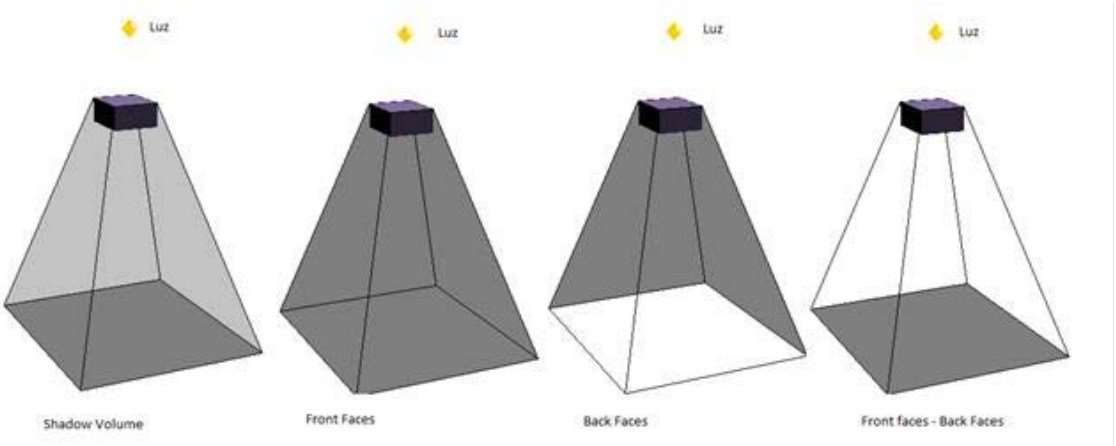


Figura 17: Cálculo de sombra

No programa, ao invés de se calcular a silhueta do objeto, e calcular o volume de sombra do objeto inteiro é feito o cálculo do volume de sombra de cada face, gerando um conjunto de pequenos volumes em que a união dos volumes cria a sombra do objeto.

Apenas as faces que estão voltadas para a luz foram selecionadas para projetar a sombra, pois economiza processamento em desenhar os volumes. As faces que não estão voltadas para a luz criavam efeitos colaterais não desejados na sombra, áreas claras onde não deveriam existir, prejudicando o resultado final.

6 Resultados

Ao final do projeto, todas as etapas implementadas foram integradas resultando num programa em que a utilização se dá de modo rápido e o resultado ocorre em tempo real.

Os resultados das simulações com objetos virtuais mostram-se fieis, se comparados com a mesma foto com um objeto real.

A seguir estão os resultados obtidos nos testes feitos com o software finalizado:

6.1 Simulação 1

Figura 18: Simulação 1 - Original

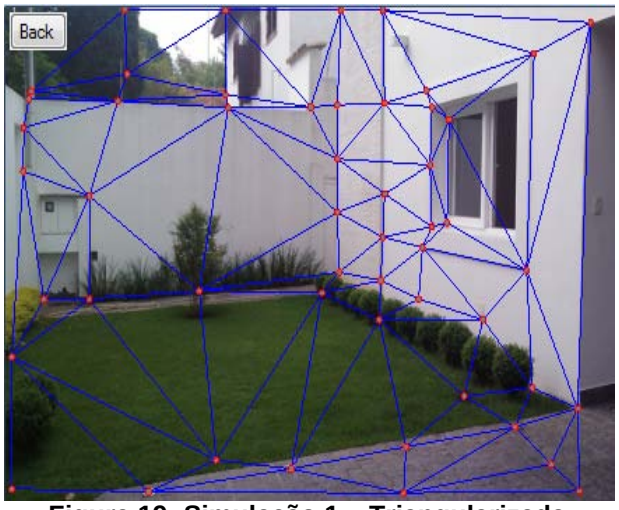


Figura 19: Simulação 1 – Triangularizada

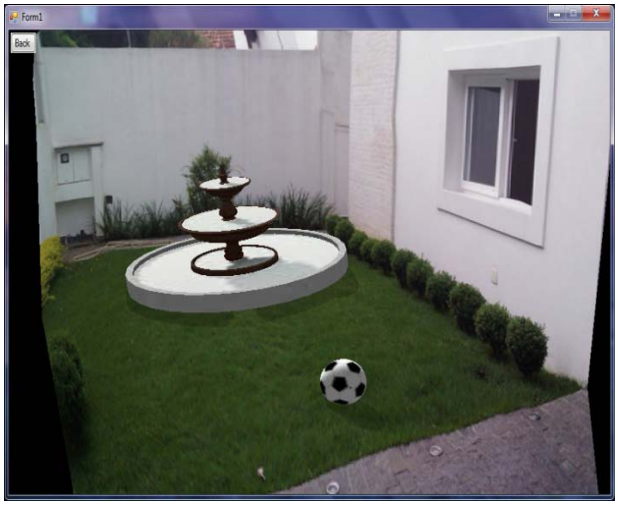


Figura 20: Simulação 1 - Final

6.2 Simulação 2

Figura 21: Simulação 2 - Original

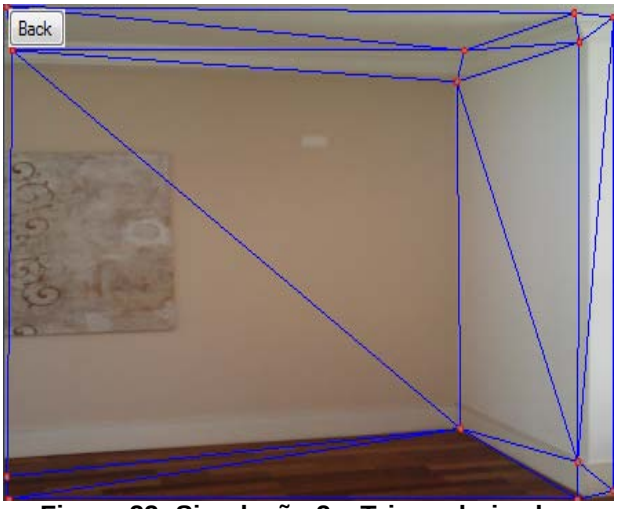


Figura 22: Simulação 2 – Triangularizada

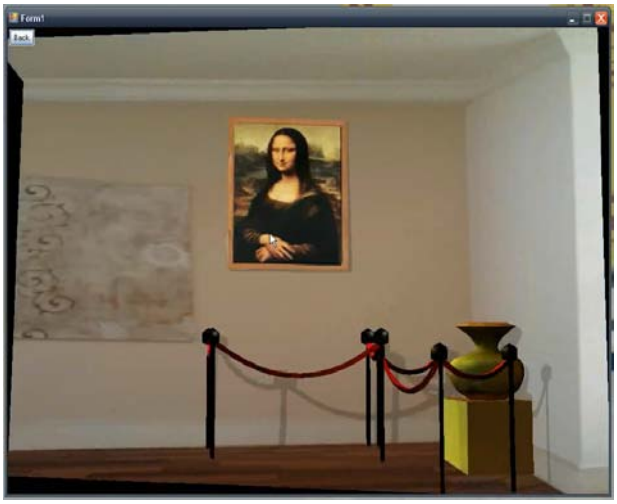


Figura 23: Simulação 2 - Final

6.3 Simulação 3

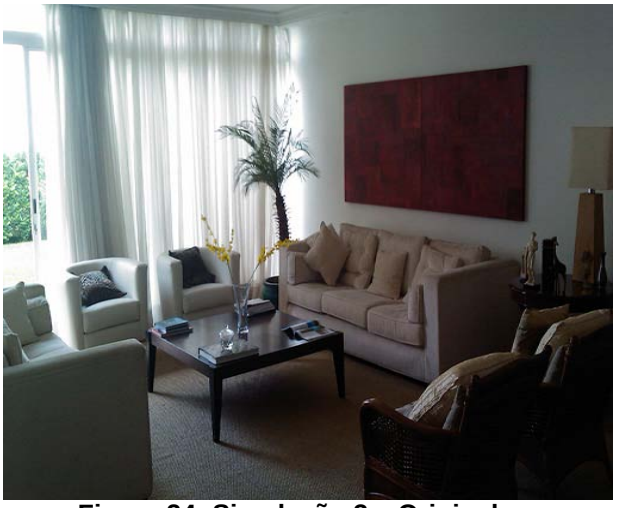


Figura 24: Simulação 3 – Original

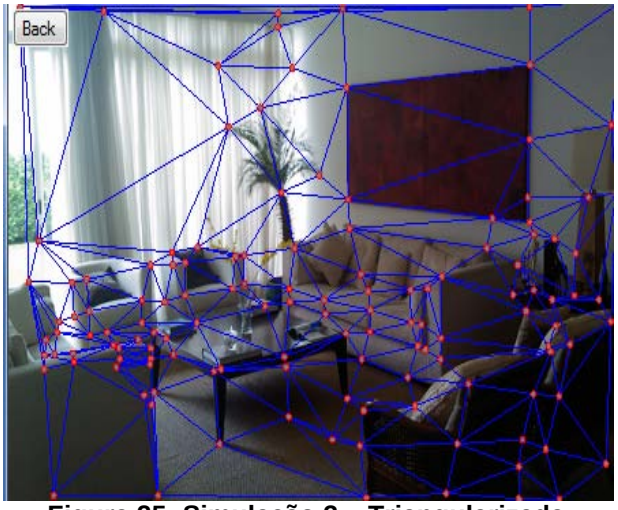


Figura 25: Simulação 3 – Triangularizada

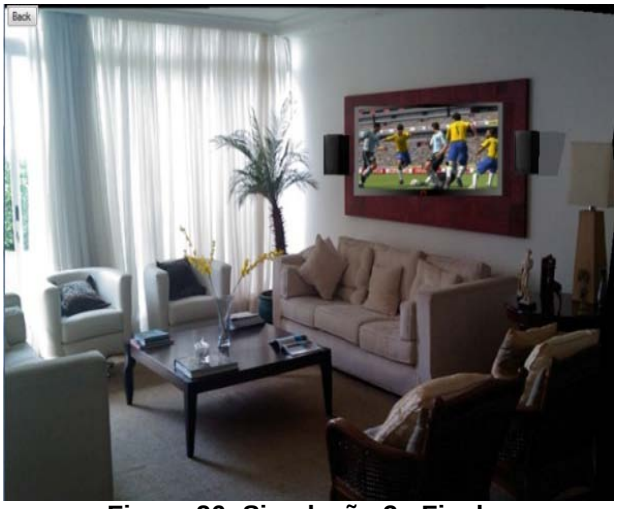


Figura 26: Simulação 3 - Final

6.4 Simulação 4

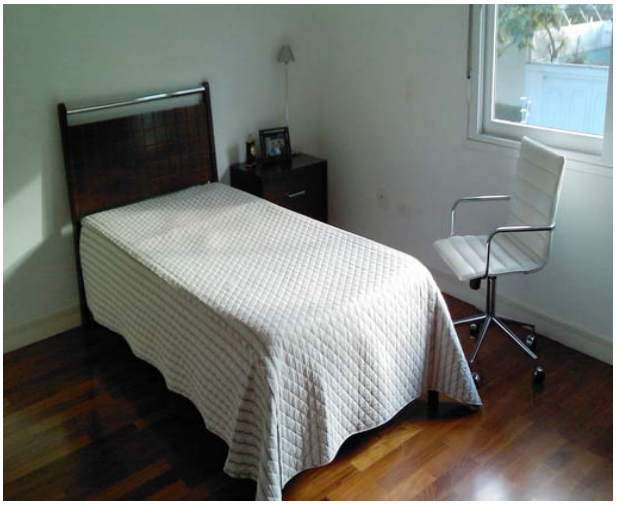


Figura 27: Simulação 4 – Original

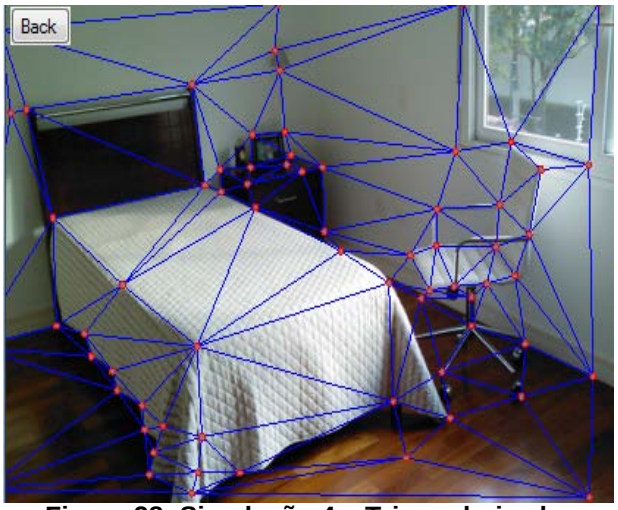
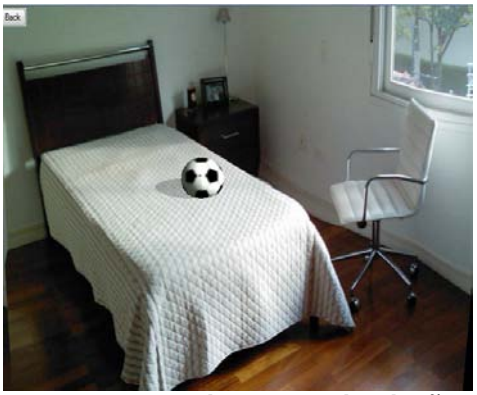


Figura 28: Simulação 4 – Triangularizada



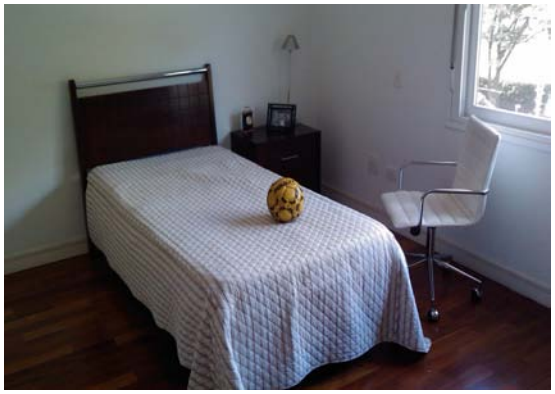


Figura 29: Simulação 4 – Final e Objeto real

7 Conclusão

O resultado final do trabalho foi muito recompensador. A fim de comprovar a eficiência e usabilidade do software desenvolvido fizemos testes com usuários diferentes, uns já mais familiarizados com computadores e outros menos, e dentro de cada grupo pessoas de diferentes idades.

O resultado dos testes com o publico nos permitiu fazer mínimas adaptações melhorando ainda mais a usabilidade do programa. Refazendo os testes pudemos ver que o uso do programa estava com boa usabilidade, além de uma interface amigável e auto-explicativa ao usuário.

Com relação a melhorias que podem ser feitas no trabalho uma delas seria o reconhecimento estéreo automático dos elementos contidos nas imagens estereoscópicas, evitando dessa forma a inserção manual dos pontos em ambas as imagens pelo usuário.

8 Parte Subjetiva (Eduardo)

Esta parte do trabalho contém um relato pessoal de como o Trabalho de Formatura foi desenvolvido. Desde uma listagem e comentários das disciplinas que me ajudaram na idealização, elaboração e conclusão do trabalho, até os desafios e que tive no trabalho.

Gostaria de fazer um agradecimento ao meu amigo Daniel pela ótima interação que tivemos nesse ano de desenvolvimento do trabalho e agradecer também toda a ajuda e disposição do professor Marcel que foi muito útil para chegarmos ao resultado final.

8.1 Desafios e Frustrações

O início do Trabalho de Formatura foi bastante conturbado, principalmente pela dificuldade de encontrarmos um orientador com um tema que fosse do interesse ou algum que aceitasse o tema que estava sendo proposto.

Primeiramente a idéia era desenvolver um projeto na área de Engenharia de Software, por ser uma disciplina que me despertou muito interesse devido ao seu conteúdo de gestão de software e administração de projetos. Visto que seria muito difícil encontrar um orientador para o tema que estávamos propondo, decidi partir para outra linha de pesquisa que tinha interesse.

Durante o curso de Computação Gráfica fiquei bastante motivado com a disciplina, por toda a parte de interação entre codificação e o resultado visual que obtinha a cada programa feito, gerando muita satisfação. Sendo assim, não poderia ser outro tema a ser escolhido para o Trabalho se não Computação Gráfica.

Definido o tema juntamente com a minha dupla começamos todo um trabalho de planejamento e definição de metas a serem cumpridas no decorrer do ano de desenvolvimento do trabalho.

O começo do trabalho foi o mais complicado, principalmente porque queríamos fazer um software extremamente complexo com muitas automatizações e reconhecimento de imagens o que foi ficando claro que não seria possível em tão pouco tempo.

A mudança de estratégia deu-se em tempo limite pelo fato de ter sido tentado, juntamente com nosso orientador, mantermos o plano de metas e a abrangência definida inicialmente.

Após entendermos que não poderíamos estender nosso trabalho para além do foco principal de construção de cenários tridimensionais e síntese de objetos no seu interior, o projeto transcorreu, ainda que com bastante pesquisa ao longo de todo o projeto, parte essa em que enfrentamos dificuldades com

códigos mal documentados e dificuldade na comunicação com seus desenvolvedores, aumentando nosso trabalho.

Ao final do projeto obtivemos um resultado que, para o nosso nível alto de cobrança inicial, me gerou uma imensa satisfação em ver o funcionamento do software.

Foi uma excelente experiência e muito prazeroso o desenvolvimento desse trabalho e ver o seu resultado final.

8.2 Disciplinas relevantes para o trabalho

Certamente quase todas as disciplinas contribuíram para a realização do trabalho de formatura, se não de forma direta, indiretamente. Vou listar abaixo as disciplinas que julgo que mais contribuíram para o desenvolvimento do trabalho:

MAC 122 – Princípios de Desenvolvimento de Algoritmos

Nessa disciplina foi onde percebi o que realmente seria o curso de Ciência da Computação e toda a sua beleza e complexidade. Todos os programas feitos ao longo do curso e as aulas que tive contribuíram de forma estrutural para que o trabalho de formatura fosse realizado. Comecei a ganhar alguma experiência nessa disciplina.

MAC 323 – Estrutura de Dados

Certamente outra disciplina estrutural na minha formação, o trabalho dedicado nela e todo o aprendizado resultante disso foi muito mais do que útil para a realização do trabalho, mas sim algo de extremo valor profissional e pessoal. A experiência na realização dos trabalhos propostos nessa disciplina, assim como o ganho de conhecimento foi essencial para que o trabalho saísse do plano das idéias e tomasse forma.

MAC 332 – Engenharia de Software

Essa disciplina é do meu ponto de vista extremamente interessante e importante na formação do aluno do BCC, acredito que mais matérias no caráter dessa disciplina poderiam ser dadas, pois ela influenciou diretamente na codificação do Trabalho de Formatura e no seu planejamento.

MAC 417 – Visão e Processamento de Imagens

A disciplina influenciou diretamente no nosso trabalho, aprendemos conceitos de tridimensionalidade, iluminação de ambientes, como se comportava o sistema visual humano, posicionamento e foco de câmera e etc.

MAC 420 – Introdução a Computação Gráfica

Essa foi a disciplina mais atuante e relevante na execução do Trabalho de Formatura, os conceitos aprendidos na disciplina foram essenciais para que o software fosse concluído. Os programas feitos na disciplina foram um ótimo treino e aprendizado na integração e fixação de todos os conceitos aprendidos como textura, imagem, câmera, luz, OpenGL e etc.

MAC 446 – Princípios de Interação Homem Computador

Essa disciplina me trouxe um grande aprendizado em interfaces gráficas, em como fazer a máquina interagir com a pessoa que está por trás através de uma interface auto-explicativa, amigável e de fácil utilização. Assunto esse que tratamos com prioridade no desenvolvimento do nosso software neste trabalho.

8.3 Futuro

O resultado final do trabalho foi ótimo para mim, mas ainda tenho o interesse de refiná-lo e ainda inserir novas funcionalidades que contribuam com a melhoria e automatização do software, como o reconhecimento automático de imagem na geração do cenário tridimensional e outras idéias que possam vir a aparecer ao longo do tempo.

9 Parte Subjetiva (Daniel)

Posso dizer que dentre todos os trabalhos que realizei durante o meu curso, o resultado final do nosso trabalho de formatura foi o mais gratificante de todos. Pude realizar um projeto que tive motivação pessoal, pois se tratava de uma Área que tenho muita afinidade, a de computação gráfica.

Sou muito grato àqueles que me ajudaram na realização deste projeto, principalmente ao nosso orientador Marcel e meu colega de dupla Eduardo. Sem este trabalho em equipe não seria possivel finalizar este projeto principalmente com a qualidade de resultados que obtivemos.

9.1 Desafios e Frustrações

Tivemos muita dificuldade no inicio do projeto. Primeiramente não conseguimos encontrar um professor disposto a nos orientar o nosso trabalho de formatura na nossa primeira escolha area de engenharia de software.

Tendo em vista a indisposição dos professores a nos fornecer ajuda, resolvemos abordar outras áreas de nosso interesse para que fosse possivel encontrar um orientador disposto.

Finalmente encontramos a diposição o Professor Marcel, que, além de mostrar grande interesse sobre o tema do projeto, se propôs a ajudar e fornecer material didático para realizarmos o projeto.

Durante o principio dos desenvolvimentos encontramos uma enorme barreira para criar um software automatizado de reconhecimento estereoscópico, devido a grande complexidade do problema, baixa qualidade dos resultados eontrado, a má documentação das pesquisas já existentes, e a dificuldade de nos comunicar com profissionais da área. Isso me gerou um grande desânimo, pois havia muito trabalho, muita pesquisa e nenhum resultado prático. Sempre voltávamos para a "estaca zero".

Devidamente orientado pelo professor Marcel, a mudanca da estratégia da realização do trabalho me revigorou o ânimo, pois finalmente consegui ver um resultado positivo de nosso trabalho. Este meu ânimo prevaleceu durante todo o desenvolvimento até o resultado final, que como já havia dito anteriormente, foi muito gratificante.

9.2 Disciplinas relevantes para o trabalho

MAC 122 – Princípios de Desenvolvimento de Algoritmos

Esta disciplina foi fundamental, pois me deu toda a base sobre programação e técnicas de programação. Foi o primeiro contato com um curso de verdade na área.

MAC 323 – Estrutura de Dados

Esta disciplina foi o meu amadurecimento como programador. Fiquei surpreso com as automatizações que aprendi durante a disciplina. Muitas das estruturas de dados utilizadas no projeto como listas, hash tables e estruturas foram aprendidas nesta disciplina.

MAC 417 – Visão e Processamento de Imagens

Posso dizer que utilizamos todo o conhecimento de visão computacional no projeto. Principalmente para compreender todos os materiais que pesquisei em reconhecimento estereoscópico de imagens e estimação de profundidade.

MAC 332 – Engenharia de Software

Fundamental para o gerenciamento de trabalho, plano de desenvolvimento e alocação da nossa mão de obra nas estapas do projeto. Além de projetar as classes de forma modular faciltando a implementação e manutenção do software.

MAC 446 – Princípios de Interação Homem Computador

Graças a experiência adquirida nesta disciplina tive a sensibilidade de pensar na usabilidade do software, não apenas na utilidade final. Assim foi realizado implementações de usabilidade na interface do projeto, como zoom, divisão da tela nas imagens e agrupamento dos controles e botões de forma intuitiva.

MAT0139 – Álgebra Linear para Computação

Fundamental para entender muitos dos conceitos de computação gráfica como espaço vetorial, e transformações de variedade afim atravéz de matrizes de transformações.

MAC 420 – Introdução a Computação Gráfica

Esta disciplina merece maior destaque, pois mais da metade do desenvolvimento é relacionado à programação em OpenGL, cálculos espaciais tridimensionais, desenho de faces e vertices, projeção de textura e sombras e transformações espaciais.

9.3 Futuro

Pretendo refinar os resultados para não ser somente um resultado artístico, mas também poder resultar em resultados precisos para reconhecimento tridimensional de peças e ambientes.

Outro objetivo é implementar o reconhecimento automático de pontos correspondentes estereoscópicamente.

10 Bibliografia

[1] Q. Yang, L. Wang, R. Yang, H. Stewénius, and D. Nistér. Stereo matching with color-weighted correlation, hierarchical belief propagation and occlusion handling. PAMI 2008.

[2] Página: Middlebury, vision
http://vision.middlebury.edu/

[3] A. Klaus, M. Sormann and K. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. ICPR 2006.

[4] Z. Wang and Z. Zheng. A region based stereo matching algorithm using cooperative optimization. CVPR 2008.

[5] JAIN, Ramesh, KASTURI, Rangachar and SCHUNCK, Brian. *Machine Vision.*

[6] HILL, Francis and KELLEY, Stephen. *Computer Graphics using OpenGL, 3rd ed.*

[7] Delaunay Triangulation in .NET 2.0 by Morten Nielsen.
http://local.wasp.uwa.edu.au/~pbourke/papers/triangulate/morten.html

[8] CSGL
http://csgl.sourceforge.net/

[9] Nehe - lição 27
http://nehe.gamedev.net/

[10] Salmon reader by Scott Ellington
http://www.salmonsalvo.net/blog/?p=19

[11] OpenGL
http://www.opengl.org/

[12] Visual C# Express
http://csgl.sourceforge.net/

[13] Página pessoal: Eduardo Bretones M. Apolinário contendo os resultados
http://www.linux.ime.usp.br/~eduapo/mac499

[14] Página pessoal: Daniel Ferreira Santos contendo os resultados
http://www.linux.ime.usp.br/~artsjedi/mac499