Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Análise de Imagens Tomográficas da Ciência do Solo em Ambiente de Realidade Virtual"

LEONARDO CASTRO BOTEGA

São Carlos
Agosto/2008

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

B748ai

Botega, Leonardo Castro.
Análise de imagens tomográficas da Ciência do Solo em ambiente de realidade virtual / Leonardo Castro Botega. -- São Carlos : UFSCar, 2009.
213 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2008.

1. Processamento de imagens. 2. Realidade virtual. 3. Tomografia computadorizada. 4. Porosidade do solo. I. Título.

CDD: 006.6 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

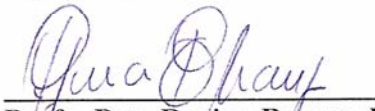
"Análise de Imagens Tomográficas da Ciência do Solo em Ambiente de Realidade Virtual"

LEONARDO CASTRO BOTEGA

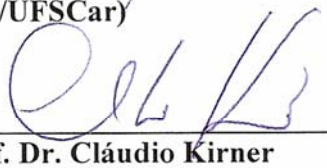
Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

Membros da Banca:

Prof. Dr. Paulo Estevão Cruvinel
(Orientador – EMBRAPA – Instrumentação Agropecuária)



Profa. Dra. Regina Borges de Araújo
(DC/UFSCar)



Prof. Dr. Cláudio Kirner
(UFOP)

São Carlos
Agosto/2008

AGRADECIMENTOS

Primeiramente a Deus pelo dom da vida e sustentação de minhas forças.

Aos meus pais João Luiz Botega e Salete Piemonte de Castro Botega, os quais depositaram grande confiança e apoio em minha formação acadêmica.

A minha irmã Mariana de Castro Botega, cujos conselhos contribuíram imensamente para a continuidade do presente trabalho.

Ao meu orientador Dr. Paulo Estevão Cruvinel pela excelente orientação, paciência e sábios conselhos.

Ao Dr. Álvaro Macedo da Silva pelo apoio e colaboração científica em cessão das imagens tomográficas de resolução micrométrica utilizadas neste trabalho.

A minha companheira Claudia Cristina da Silveira Campos pela paciência, atenção e cuidado.

Aos amigos da Universidade Federal de São Carlos: Ricardo Rios, Marcio Campos, Reginaldo Gotardo, Alexandre Melo, Vinícius Durelli, Tatiane Nogueira, Cristiane Santana, Simone Borges, Talita Perciano, Débora Correa, Alexandre Levada e Eliane Pereira pelo companheirismo e constante troca de experiências e conhecimento.

Aos amigos da Embrapa Instrumentação Agropecuária: Maurício F. L. Pereira e Marcos A. M. Laia pela amizade e auxílio em dúvidas diversas.

RESUMO

Este trabalho apresenta um ambiente de Realidade Virtual dedicado à análise de imagens tomográficas da Ciência do Solo. Sua arquitetura utiliza modelos provenientes de um algoritmo de reconstrução volumétrica (3-D) adicionado a diversos processos gráficos de manipulação e visualização, de tal forma a possibilitar imersão e interação do usuário com a cena virtual. Sua validação foi realizada com base em um estudo de caso envolvendo análise de porosidade de amostras de solos agrícolas, os quais apresentam caminhos preferenciais para o fluxo de água e solutos (*fingering*). Resultados ilustram a verificação consistente dos caminhos preferenciais das amostras de solos agrícolas, analisadas com base no ambiente desenvolvido.

PALAVRAS-CHAVE: Processamento de Imagem, Realidade Virtual, Tomografia Computadorizada, Porosidade de Solos Agrícolas.

ABSTRACT

This work presents a Virtual Reality environment dedicated to the analysis of tomographic images of Soil Science. Its architecture uses models proceeding from a volumetric (3-D) reconstruction algorithm, summed with several graphics processes of manipulation and visualization, in a way to provide immersion and interaction of the user with the virtual scene. Its validation was performed based on a case study involving the analysis of the porosity of the agricultural soil samples, which presents preferential paths for the water and solute flow (fingering). Results illustrate the consistent verification of the preferential paths of the agricultural soil samples, analyzed based on the developed environment.

KEYWORDS: Image Processing, Virtual Reality, Computerized Tomography, Porosity of Soil Samples.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Introdução

A análise de imagens tomográficas para inspeção do interior de um corpo ou objeto de forma não destrutiva, tornou-se uma prática altamente difundida na área de visualização científica. Tal prática passou a ser de grande importância em muitas aplicações. Entretanto, tem-se constatado que, efetivamente, o uso de imagens bidimensionais muitas vezes não é suficiente para uma análise completa dos dados, sendo necessária a utilização de imagens tridimensionais.

Paralelamente, no Brasil, tem-se visto poucas unidades de pesquisa com o desenvolvimento de projetos utilizando Realidade Virtual, destinados a potencializar e complementar as técnicas de analise requeridas em diagnósticos não invasivos. Parte desta limitação se deve às dificuldades técnicas e ao seu alto custo. Porém, tem sido observada neste cenário de realidades, a interação envolvendo diferentes áreas do conhecimento, com o propósito de aperfeiçoar e democratizar o uso desses recursos no processo ensino-treinamento-aprendizagem.

A literatura registra um conjunto de trabalhos que envolvem, em seu conteúdo, o desenvolvimento de ambientes de Realidade Virtual e Tomografia Reconstrutiva. Esses trabalhos envolvem reconstrução de imagens tomográficas, bidimensionais ou tridimensionais, e foram desenvolvidos em diversas instituições nacionais e internacionais. Sistemas como descrito por Perdigão e colaboradores, no qual modelos tridimensionais são gerados a partir de imagens médicas (Perdigão *et al*, 2005),

algoritmos de interpolação (Prado e Traina, 1996), reconstrução de mandíbula (Villamil *et al*, 2005), implementação de *interface* gráfica em Java para reconstrução tomográfica em medicina nuclear (Silva e Andrade, 2004), ambiente virtual para planejamento de hepatectomia (Benes e Bueno, 2003) e prototipagem a partir de imagens reconstruídas (Bazan, 2004) (Souza *et al*, 2001), dentre outros.

O presente trabalho apresenta como objetivo a organização e implementação de um ambiente sintético, que possibilite a visualização, análise e manipulação de amostras produzidas por um algoritmo de reconstrução tridimensional de imagens tomográficas de raios-*x*, através de ferramentas gráficas computacionais e dispositivos não convencionais de Realidade Virtual, visando imersão e interação do usuário com as entidades da cena e possibilitando a análise não invasiva de amostras de solos agrícolas. Assim, o presente trabalho é organizado da seguinte forma:

No Capítulo 1 é realizada uma revisão bibliográfica sobre Realidade Virtual, abordando seu histórico, conceitos, sistemas, dispositivos e implementações, desde seu surgimento em meados dos anos 70 até os dias atuais.

O Capítulo 2 traz uma descrição teórica detalhada das técnicas de Tomografia Computadorizada de raios-*x* e sua aplicação em Ciência de Solos, abordando desde a aquisição de projeções, até a reconstrução bidimensional e tridimensional de imagens de amostras agrícolas.

O Capítulo 3 realiza uma descrição da interface de programação utilizada para a implementação do ambiente de análise de amostras tridimensionais, demonstrando sua aplicabilidade no desenvolvimento de sistemas de Realidade Virtual.

Já o Capítulo 4 apresenta a Estruturação Conceitual e Metodológica do presente trabalho, descrevendo em detalhes todos os processos implementados, exemplificados sob pseudocódigos.

No Capítulo 5 são demonstrados e comentados os Resultados obtidos após o desenvolvimento dos processos descritos no capítulo anterior. Um estudo de caso é também apresentado para a validação do sistema, considerando aplicações em Ciência de Solos.

Finalmente, no Capítulo 6 são apresentadas as conclusões obtidas, levando em consideração cada etapa do desenvolvimento e a sua aplicabilidade em processos de análise de amostras de solos agrícolas, além da apresentação de temas para trabalhos futuros.

Capítulo I
Realidade Virtual

A Realidade Virtual configura-se como uma interface avançada de terceira geração para aplicações computacionais, na qual o usuário pode interagir, em tempo real, a partir de um ambiente tridimensional sintético, utilizando dispositivos multisensoriais (Kirner *et al*, 1995).

A tecnologia surgiu com o pesquisador Ivan E. Sutherland, o qual desenvolveu o primeiro sistema gráfico interativo, o qual interpreta desenhos como dados de entrada e realiza associações com topologias conhecidas, gerando novos desenhos (Sutherland, 1963). Já o termo Realidade Virtual (RV) surgiu em meados dos anos 70, onde pesquisadores sentiram a necessidade de uma definição para diferenciar as simulações computacionais tradicionais dos mundos digitais que começavam a ser criados. Nasciam então as interfaces de terceira geração, onde interações eram produzidas sobre as situações geradas, utilizando-se de comandos não convencionais, diferenciando-se das interfaces dotadas apenas de reprodução multimídia, mantidas até então por interfaces bidimensionais de primeira e segunda geração (Krueger, 1977) (Bolt,1980) (Lanier, 1984). O termo é bastante abrangente, e logo, acadêmicos, desenvolvedores de *software* e pesquisadores procuram definir Realidade Virtual baseados em suas próprias experiências. Pimentel define Realidade Virtual como sendo o uso de tecnologia para convencer o usuário de que ele está em outra

realidade. Em geral, Realidade Virtual refere-se a uma experiência interativa e imersiva baseada em imagens gráficas tridimensionais geradas em tempo real por computador (Pimentel, 1995).

Machover afirma que a qualidade dessa experiência em Realidade Virtual é essencial, pois deve estimular ao máximo, de forma criativa e produtiva, o usuário. Os sistemas de Realidade Virtual também precisam fornecer uma reação de forma coerente aos movimentos do participante, tornando a experiência consistente (Machover, 1994). O principal objetivo desta nova tecnologia é fazer com que o participante desfrute de uma sensação de presença no mundo virtual (Jacobson, 1994). Para propiciar esta sensação de presença os sistemas de RV integram sofisticados dispositivos, os quais podem ser aplicados em ferramentas das mais diversas áreas, contribuindo para a análise e manipulação de representações virtuais. Estes dispositivos podem ser luvas de dados (Sun, 2007) (Immersion, 2007) e capacetes imersivos (*Head Mounted Displays*) (Sensics, 2007) (Darpa, 2007)(VRealities, 2007). Na prática, a RV permite que o usuário navegue e observe um mundo tridimensional sob seis graus de liberdade (6 DOF)[1]. Isso exige a capacidade do *software* de definir, e do *hardware* de reconhecer, seis tipos de movimento: para frente/para trás, acima/abaixo, esquerda/direita, inclinação para cima/para baixo, angulação à esquerda/à direita e rotação à esquerda/à direita. Na essência, a RV é um espelho da realidade física, na qual o indivíduo existe em três dimensões, tem a sensação de estar imerso no ambiente e tem a capacidade de interagir com o mundo ao seu redor. Os dispositivos de RV simulam essas condições, chegando ao ponto em que o usuário pode tocar virtualmente os objetos de um mundo virtual e fazer com que eles respondam, ou mudem, de acordo com suas ações (Von Schweber, 1995).

No final de 1986 uma equipe da NASA já possuía um ambiente virtual que permitia aos usuários ordenar comandos pela voz, escutar fala sintetizada e som 3-D, e manipular objetos virtuais diretamente através do movimento das mãos. O mais importante é que através desse trabalho verificou-se a possibilidade de comercialização de um conjunto de novas tecnologias, com o custo de aquisição e desenvolvimento cada vez mais acessível (Pimentel, 1995).

1

DOF (*Degrees of freedom*): Graus de liberdade, relativo aos movimentos de translação e rotação do dispositivo de RV.

A conscientização de que os empreendimentos da NASA tornavam-se tecnologias comercializáveis deu início a inúmeras pesquisas em RV no mundo inteiro. Organizações variando de empresas de *software* até grandes corporações de informática começaram a desenvolver e vender produtos e serviços ligados à Realidade Virtual. Em 1987, a VPL Research Inc. começou a vender capacetes e luvas digitais e em 1989 a AutoDesk apresentava o primeiro sistema de RV baseado num computador pessoal (PC) (Jacobson, 1994).

Apesar da RV ter sido inventada há mais de 30 anos ela tem a cada ano evoluído substancialmente. Seu custo permaneceu alto durante muito tempo devido aos equipamentos de alta tecnologia envolvidos. Entretanto, a evolução das indústrias de computadores e o grande avanço tecnológico possibilitaram a utilização da RV a um preço acessível, cerca de metade dos custos de dez anos atrás, fazendo com que deixasse de ser exclusividade de instituições de pesquisa ou governamentais (Machado, 1995).

No Brasil, um dos primeiros grupos de pesquisa em Realidade Virtual (RV) foi organizado no Departamento de Computação da Universidade Federal de São Carlos (DC/UFSCar), criado em outubro de 1995 (Kirner, 1995). Seu principal projeto, denominado AVVIC-PROTEM-CC, baseou-se na criação de um ambiente e aplicações de pesquisa de RV distribuída, provendo melhorias nas condições de visualização interativa e compartilhada em ambiente colaborativo. Neste âmbito, encontram-se na literatura da área trabalhos desenvolvidos que contribuíram para a popularização das técnicas e dispositivos de RV no País, tais como: Modelagem dinâmica de mundos virtuais (Schneider, 1997), detecção de colisão (Peruzza, 1997), ambiente virtual interativo tridimensional (Ipolito, 1997), suporte virtual para ensino a distância (Kubo, 1997), suporte para aplicações de RV e visualização (Santos, 1998). Ainda neste período, junto à Escola Politécnica da Universidade de São Paulo (EPUSP), outro trabalho relevante desenvolvido neste segmento foi a especificação e análise de um sistema distribuído de Realidade Virtual, a primeira tese de doutorado no país abordando RV (Araújo, 1996).

Atualmente, além do DC/UFSCar encontram-se no País outros grupos que desenvolvem estudos de RV (aproximadamente 30), onde segundo dados da Sociedade Brasileira de Computação (http://www.sbc.org), os principais estão localizados nas seguintes instituições: SVVR/LNCC, TecGraf/PUC, Interlab/USP,

LSI/USP, GRV/UNESP, GRVa/UFRJ, GRV/UFU, GMRV/UNIMEP, GRV/UFPE e LApIS/UNIVEM. Dentre os principais trabalhos desenvolvidos encontram-se: sistema de reconstrução de mandíbula (Villamil *et al*, 2005), ambiente virtual para planejamento de hepatectomia (Benes e Bueno, 2003), prototipagem a partir de imagens reconstruídas (Bazan, 2004) (Souza *et al*, 2001), ferramentas estereoscópicas para treinamento médico (Botega e Nunes, 2005), sistemas de geração de modelos tridimensionais a partir de imagens médicas (Perdigão *et al*, 2005), Frameworks de simulação de procedimentos médicos (Oliveira, 2006), simuladores de segmentação de imagens (Delfino, 2006), sistema de análise de amostras agrícolas tridimensionais (Botega e Cruvinel, 2007), framework de Realidade Aumentada baseado em FPGA (Lima *et al*, 2007), gerador de jogos utilizando Realidade Aumentada (Tsuda *et al*, 2007), integração de Realidade Aumentada em interação entre robôs (Calife *et al*, 2007), iluminação realística (Pessoa *et al*, 2008), gerador de aplicações multimídia com RV (Malfatti *et al*, 2008), estimador de profundidade em ambientes de RV (Sanches *et al*, 2008), incorporação de comandos de voz em ambientes de RV (Pizzolato *et al*, 2008) e sistema de rastreamento virtual (Teixeira *et al*, 2008).

1.1. Sistemas de RV

Os sistemas de RV diferem entre si de acordo com os níveis de imersão e de interatividade proporcionado ao usuário. Esses níveis são alcançados pelos diversos tipos de dispositivos de entrada e saída de dados do sistema, além da performance do computador que o hospeda. Existem algumas formas de classificação dos sistemas de RV. Shepherd (1993), identifica duas grandes classes: tele-presença, em que um ambiente sintético comum é compartilhado entre várias pessoas como uma extensão ao conceito de trabalho cooperativo suportado por computador, e tele-operação, onde robôs agem sobre um elemento, seja ele um corpo humano ou um produto sendo manufaturado. Entretanto, esses termos sofreram vários desdobramentos e mesmo inversões.

Segundo Araújo (1996), as aplicações de RV, em geral, são classificadas da seguinte forma: tele-colaboração, tele-presença e visualização científica 3-D. Sistemas de tele-colaboração implementados permitem aos usuários compartilhar um mesmo

espaço e manipular objetos, sentindo o peso dos mesmos por meio de dispositivos de *feedback*. Um sistema de tele-presença, ou tele-existência, estende as capacidades sensoriais de um usuário humano, bem como a suas habilidades de solução de problemas, para um ambiente remoto. Na tele-presença, também conhecida como tele-operação ou tele-robótica, o robô executa as tarefas fisicamente separadas de seu operador humano. As ações executadas pelo operador são traduzidas em ações executadas pelo robô em seu ambiente remoto, ao mesmo tempo em que é emitido *feedback* sensorial ao operador humano, que se sente como se estivesse realmente presente no ambiente remoto. A tele-presença pode ser mais claramente vista como uma técnica de visão computacional que realça a função intermediária entre o participante e o ambiente (Latta, 1994).

Já a Visualização Científica permite que grandes quantidades de dados gerados por simulações computacionais sejam traduzidas em representações visuais tridimensionais. Dados podem ser renderizados como pontos, linhas, curvas, superfícies, volumes, cores, e mesmo como sons. Também permite a manipulação dos modelos sob vários ângulos e posições, permitindo uma ampla exploração de propriedades matemáticas intrínsecas (Upson *et al*, 1989) (Hultquist *et al*, 1992) (Spencer, 2001).

Jacobson (1994) e Pimentel (1995) consideram que sistemas ou estilos de RV podem ser classificados como RV de Simulação, RV de Projeção, Realidade Aumentada ou Realçada *(Augmented Reality)*, Tele-presença, *Displays* Visualmente Acoplados (*Visually Coupled Displays*) e RV de Mesa.

A RV de Simulação trata-se do estilo mais antigo, originado com os simuladores de vôo desenvolvidos pelos militares americanos após a Segunda Guerra Mundial. Um sistema desse tipo basicamente imita o interior de um carro, avião ou jato, colocando o participante dentro de uma cabine onde se encontram monitores que apresentam um mundo virtual que reage aos comandos do usuário. Um sistema de RV de Simulação não processa imagens em estéreo, as imagens são geradas de forma bastante rápida. Em alguns sistemas as cabines são montadas sobre plataformas móveis, e os controles oferecem *feedback* tátil e auditivo (Burdea *et al*, 1994).

A RV de Projeção, também conhecida como Realidade Artificial, foi criada nos anos 70 por Myron Krueger (Krueger, 1977). Na RV de Projeção o usuário está fora do mundo virtual, mas pode se comunicar com os personagens virtuais. O sistema

VIDEOPLACE, criado por Krueger naquela época, capturava a imagem do usuário e projetava-a em uma grande tela que representava um mundo virtual nas quais os usuários podiam interagir uns com os outros ou com os modelos virtuais. Krueger usou o termo Realidade Artificial para descrever o tipo de ambiente criado pelo seu sistema, o qual não exigia que o participante vestisse ou usasse dispositivos de entrada.

Já a Realidade Aumentada (*Augmented Reality*) utiliza os dispositivos não convencionais de mais baixo custo para promover a imersão e interação do usuário com modelos 3-D. Utilizando-se de câmeras, marcadores de papel e técnicas de visão computacional, esta modalidade de RV captura a cena real onde se encontram os marcadores, reconhece a estrutura constante nos mesmos e insere o modelo virtual correspondente na cena real, a qual pode ser visualizada por monitores, *HMD's* ou algum sistema de projeção (Kirner *et al*, 2007). A Figura 1 demonstra o processo de reconhecimento dos marcadores e o posicionamento dos modelos virtuais na cena real.

Em sistemas de Realidade Aumentada (RA) mais custosos, utilizam-se dispositivos visuais transparentes, por onde o usuário pode ver dados, diagramas, animações e gráficos tridimensionais sem deixar de enxergar o mundo real, obtendo informações geradas por computador sobrepostas ao mundo real. Esses *displays* transparentes são chamados *heads-up-displays* (*HUD's*). O usuário pode, por exemplo, estar consertando algo e visualizando nos óculos os dados necessários a esta operação.

A RA visa aprimorar a percepção sensorial e pode ser entendida como uma forma de interface homem máquina de quarta geração que não tem um único foco de atenção, sendo que a interação se dá com o meio de forma global e ampliada. São características básicas de sistemas de RA: o processamento em tempo real, a combinação de elementos virtuais com o ambiente real e o uso de elementos virtuais concebidos em 3-D.

Por outro lado, a Tele-presença, utiliza câmeras de vídeo e microfones remotos para envolver e imergir o usuário profundamente no mundo virtual. Controle de robôs e exploração planetária são exemplos de pesquisas de Tele-presença em desenvolvimento. Contudo, existe também um grande campo em aplicações médicas, onde são utilizadas câmeras de vídeo e cabos de fibra óptica em intervenções

cirúrgicas para auxiliar a visualização dos corpos de seus pacientes. Através da RV eles podem, literalmente, entrar no paciente, diretamente no ponto de interesse (Heeter, 1992) (Steuer, 1992).

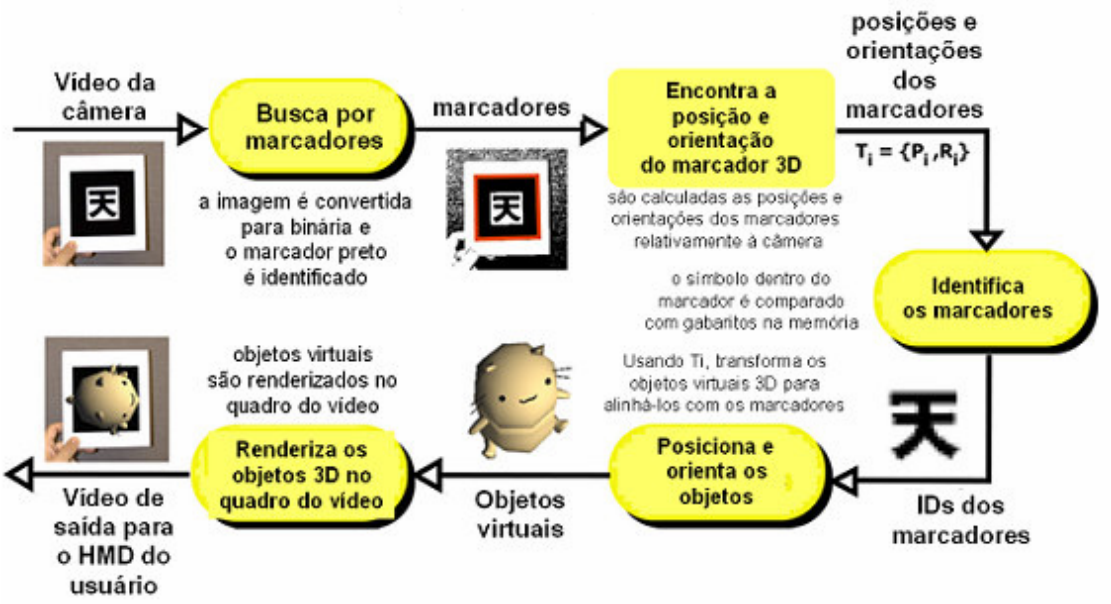


Figura 1 – Representação do processo de reconhecimento dos marcadores e o posicionamento de modelos virtuais (traduzido de Kato et al, 2000).

Os *Displays* Visualmente Acoplados (*Visually Coupled Displays ou Head Mounted Displays*) correspondem a uma classe de sistemas na qual imagens são exibidas diretamente ao usuário, que está olhando em um dispositivo que deve acompanhar os movimentos de sua cabeça. Esses dispositivos geralmente permitem imagens e sons em estéreo e detecção de movimentos da cabeça do usuário, usando essa informação para realimentação da imagem exibida (Azuma *et al*, 1994) (Romano, 2004).

A RV de Mesa (*Desktop VR*) é um subconjunto dos sistemas tradicionais de RV em que, ao invés de *Head Mounted Displays* (HMD), são utilizados grandes monitores ou algum sistema de projeção para apresentação do mundo virtual. Alguns sistemas permitem ao usuário ver imagens tridimensionais no monitor com óculos obturadores, polarizadores ou filtros coloridos. Outros ainda utilizam-se de espelhos e displays horizontais, onde a imagem é retroprojetada em uma mesa translúcida, cujo resultado se assemelha com os hologramas (Cruz-Neira *et al*, 1993) (Burdea *et al*, 1994).

1.2. Imersão, Interação e Envolvimento

A RV também pode ser caracterizada pela coexistência integrada de três idéias básicas: imersão, interação e envolvimento (Morie, 1994). A idéia de imersão está intimamente ligada ao sentimento de fazer parte do ambiente. Normalmente, um sistema imersivo é obtido com o uso de capacete de visualização, cavernas e projeções das cenas nas paredes, teto e piso (Cruz-Neira, 1992). Além do fator visual, dispositivos ligados aos demais sentidos também são importantes para o sentimento de imersão, principalmente o som (Begault, 1994; Gradecki, 1994), além do posicionamento do usuário e dos movimentos da cabeça. A visualização de uma cena 3D em um monitor é considerada não imersiva. Dessa forma, tem-se a conceituação de RV imersiva e não imersiva (Leston, 1996).

De modo geral, do ponto de vista da visualização a RV imersiva utiliza capacete ou cavernas, enquanto a RV não imersiva utiliza monitores. Entretanto, dispositivos baseados nos demais sentidos podem introduzir algum grau de imersão à RV que usa monitores (Robertson, 1993). Ainda assim, os monitores ainda apresentam alguns pontos positivos, como o baixo custo e a facilidade de uso, evitando as limitações técnicas e problemas decorrentes do uso do capacete. Todavia, a tendência deve ser a utilização da RV imersiva, considerando que a imersão, aliada à interação, justificam-se como os grandes propósitos das aplicações em RV.

A interação está ligada à capacidade do computador detectar as entradas do usuário e modificar instantaneamente o mundo virtual em função das ações efetuadas sobre ele (capacidade reativa ou *feedback*). As pessoas sempre procuram uma boa simulação em um sistema de RV, onde que as cenas mudam em resposta aos seus comandos, que é a característica mais marcante dos *vídeo games*. Para que um sistema de RV pareça mais realista, o ambiente virtual deve ser interativo (Araújo, 1996).

A idéia de envolvimento, por sua vez, está ligada ao grau de motivação para o engajamento de uma pessoa em determinada atividade. O envolvimento pode ser passivo, como ler um livro ou assistir televisão, ou ativo, como participar de um jogo com algum parceiro. A RV tem potencial para os dois tipos de envolvimento ao permitir a exploração de um ambiente virtual e propiciar a interação do usuário com o mundo virtual dinâmico.

1.3. Modelagem de Ambientes Virtuais

1.3.1. Modelagem Interativa

1.3.1.1. Seleção de objetos

A seleção de objetos é usada para destacar partes da cena virtual a serem transformadas ou editadas de alguma forma (Hearn e Baker, 1997).

Os dispositivos utilizados para a seleção de objetos são os mesmos utilizados para a seleção em *menu*, ou seja, aqueles baseados em posicionamento de cursor. Com um *mouse*, *joystick* ou luva digital, pode-se posicionar o cursor sobre as primitivas de uma representação exibida e pressionar um botão de seleção. Assim, o posicionamento é então registrado após uma pesquisa sobre diversos níveis na cena para localizar um determinado objeto a ser selecionado.

Primeiramente, a posição do cursor é comparada com as extensões das coordenadas das várias estruturas dentro da cena. Se o retângulo que delimita as fronteiras da estrutura contiver as coordenadas do cursor, a estrutura escolhida será identificada. Entretanto, se duas ou mais áreas de estruturas possuírem as coordenadas do cursor, outras checagens serão necessárias. As coordenadas das extensões de cada aresta das estruturas podem também ser identificadas individualmente, considerando que o cursor esteja sobre as coordenadas de uma única linha. Do contrário, checagens adicionais seriam necessárias para se identificar a aresta mais próxima.

Uma maneira de se identificar a aresta mais próxima à posição do cursor é calcular a distância quadrática das coordenadas do cursor (x, y) para cada segmento de linha dentro do retângulo delimitador que contém tais coordenadas. Para uma aresta com pontos finais (x_1, y_1) e (x_2, y_2), a distância quadrática de (x, y) até a aresta é calculada como:

$$d^2 = \frac{[\Delta x(y - y_1) - \Delta y(x - x_1)]^2}{\Delta x^2 + \Delta y^2} \qquad (1.1)$$

onde $\Delta x = x_2 - x_1$, $\Delta y = y_2 - y_1$ e $\Delta x^2 + \Delta y^2 \neq 0$.

Outro método para se encontrar a aresta mais próxima à posição do cursor é a especificação das dimensões de uma janela de seleção (*Clipping planes*). As coordenadas do cursor são então centradas nesta janela e as arestas candidatas são atreladas à janela de seleção. Através da redução do tamanho da janela, a um tamanho suficientemente pequeno, pode-se assegurar que uma única aresta atravessará a janela. É possível eliminar os cálculos de distância ou a utilização de janelas no processo de seleção de objetos na cena, destacando-se as estruturas sobrepostas pelo cursor. Entretanto, cabe ao usuário o discernimento a respeito da ambigüidade de seleção. A utilização de janelas de seleção bidimensional é ilustrada pela Figura 2.

1.3.1.2. Transformações

Métodos para transformações e modelagens geométricas em três dimensões são estendidos de métodos bidimensionais somados à inclusão de considerações a respeito da coordenada *Z*. Como no caso bidimensional, as transformações geométricas são expressas sob a forma de uma matriz de transformação. Qualquer seqüência de transformações é então representada como uma única matriz, formada pela concatenação de matrizes de transformação individuais em seqüência.

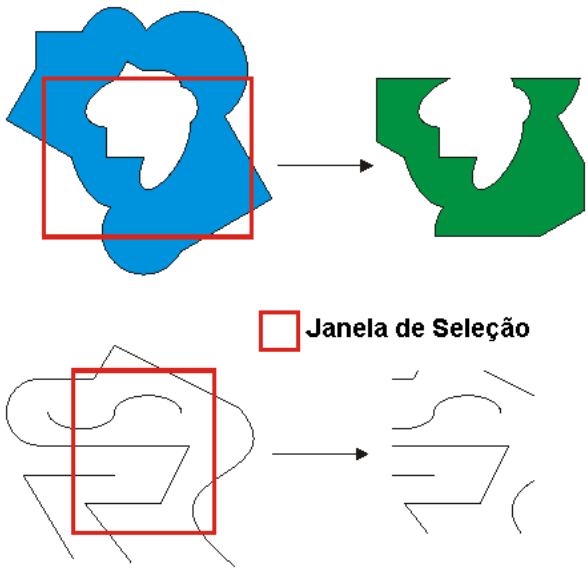


Figura 2- Representação da utilização de janelas de seleção em modelos bidimensionais, onde o plano especificado delimita a área de seleção de arestas através de alterações de sua dimensão.

1.3.1.2.1. Translação

Supondo que se tenha estabelecido um sistema de coordenadas 3-D *C* consistindo de uma origem e três eixos, na qual um ponto *P* tem as coordenadas (*x, y, z*). Os valores dessas coordenadas podem ser interpretados como as distâncias que as mesmas devem viajar ao longo dos eixos partindo da origem até alcançar o ponto *P*. Supondo agora a introdução de um segundo sistema de coordenadas *C′*, no qual suas coordenadas (*x′, y′, z′*) podem ser expressas como funções lineares das coordenadas (*x, y, z*) em *C*. Assim, pode-se notar que:

$$x'(x,y,z) = U_1 x + V_1 y + W_1 z + T_1 \qquad (1.2)$$

$$y'(x,y,z) = U_2 x + V_2 y + W_2 z + T_2 \qquad (1.3)$$

$$z'(x,y,z) = U_3 x + V_3 y + W_3 z + T_3 \qquad (1.4)$$

onde o vetor *T* representa a translação a partir da origem e os vetores *U, V* e *W* representam como a orientação dos eixos de coordenadas é alterada quando transformada de *C* para *C′*.

Em uma representação de coordenadas homogêneas tridimensionais aplicadas sobre suas matrizes identidade, um ponto é transladado da posição *C = (x, y, z)* para a posição *C′ = (x′, y′, z′)* com a matriz de operação:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (1.5)$$

Assumindo-se que as transformações são inversíveis, tem-se que:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \qquad (1.6)$$

ou:

$$P' = T \,.\, P \qquad (1.7)$$

Os parâmetros t_x, t_y e t_z especificam as distâncias de translação para as direções das coordenadas x, y e z, registradas como números reais. A representação matricial da equação (1.5) é equivalente às três equações:

$$x' = x + t_{x'} \qquad (1.8)$$

$$y' = y + t_{y'} \qquad (1.9)$$

$$z' = z + t_{z'} \qquad (1.10)$$

Um objeto é transladado em três dimensões pela transformação de cada ponto definido no objeto. Para um objeto representado por um conjunto de superfícies poligonais, cada vértice de cada superfície é transladado e redesenhado em sua nova posição.

A inversa da matriz de transformação pode ser obtida pela negação das distâncias de translação t_x, t_y e t_z. Isto produz uma translação na direção oposta, e o produto entre a matriz de translação e sua inversa produz a matriz identidade.

1.3.1.2.2. Rotação

Para gerar a transformação de rotação de um objeto, deve-se designar um eixo de rotação (sob qual o objeto será rotacionado) e o valor angular de rotação. Diferentemente de aplicações bidimensionais, onde todas as transformações baseiam-se no plano *x*-*y*, uma rotação tridimensional pode ser especificada em torno de qualquer linha no espaço. Os eixos de rotação mais fáceis de tratar são aqueles paralelos aos eixos de coordenadas cartesianas. Paralelamente, podem ser utilizadas combinações de rotações e translações para se definir uma transformação final. Por convenção, rotações de ângulos positivos produzem rotações em sentido anti-horário

sobre o eixo de coordenadas especificado, concordando com as operações de rotação bidimensionais.

A rotação tridimensional da coordenada *z* é demonstrada como exemplo à partir das equações:

$$x' = x\cos\theta - y\sin\theta \tag{1.11}$$

$$y' = x\sin\theta + y\cos\theta \tag{1.12}$$

$$z' = z \tag{1.13}$$

O parâmetro θ especifica o ângulo de rotação. Em forma de coordenadas homogêneas, a rotação tridimensional do eixo *z* é expressa pela equação:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{1.14}$$

ou

$$P' = R_z(\theta).P \tag{1.15}$$

As equações de transformação para rotações sobre os dois outros eixos de coordenadas podem ser obtidas com uma permutação cíclica das coordenadas dos parâmetros *x, y* e *z* através das substituições:

$$x \longrightarrow y \longrightarrow z \longrightarrow x \tag{1.16}$$

Substituindo as permutações da equação (1.16) nas equações (1.11), (1.12) e (1.13), pode-se obter as equações de rotação para o eixo *x*:

$$y' = y\cos\theta - z\sin\theta \tag{1.17}$$

$$z' = y\sin\theta + z\cos\theta \tag{1.18}$$

$$x' = x \tag{1.19}$$

por coordenadas homogêneas:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{1.20}$$

Ou

$$P' = R_x(\theta).P \tag{1.21}$$

Ciclicamente, permutando as coordenadas das equações (1.17), (1.18) e (1.19), pode-se obter as equações de rotação sobre o eixo *y*

$$z' = z\cos\theta - x\sin\theta \tag{1.22}$$

$$x' = z\sin\theta + x\cos\theta \tag{1.23}$$

$$y' = y \tag{1.24}$$

A representação matricial para a rotação do eixo *y* é dada por:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{1.25}$$

ou

$$P' = R_y(\theta).P \tag{1.26}$$

Uma matriz de rotação inversa é formada pela substituição do ângulo de rotação θ por -θ. Valores negativos para ângulos de rotação geram rotações em sentido horário, assim, a matriz identidade é produzida quando a matriz de rotação é multiplicada pela sua inversa. Considerando-se que apenas os valores da função seno são afetados pela inversão de sinal do ângulo de rotação, a matriz inversa também pode ser obtida intercalando-se linhas e colunas, ou seja, pode-se calcular a inversa de qualquer matriz *R* de rotação pela sua transposta ($R^{-1} = R^T$).

1.3.1.2.2.1. Ângulos de Euler

Até agora, viu-se que a forma mais intuitiva e pratica de se representar orientações em modelos tridimensionais é fornecer suas orientações baseadas nos eixos *x*, *y* e *z* sobre um ângulo determinado, compondo os parâmetros da chamada rotação por ângulos de *Euler*.

A priori, o uso de ângulos de *Euler* parece resolver todo o contexto, entretanto, há uma série de problemas (Gattass *et al*, 2002).

1.3.1.2.2.2. Rotações não comutativas.

Primeiramente é necessário destacar que as rotações, ao contrario das translações, não comutam. Em outras palavras, é possível representar um deslocamento de um modelo como sendo o resultado de somas de vários outros deslocamentos. Já para representar rotações, tal abordagem torna-se errônea, pois ordem em que as rotações são aplicadas sobre os nossos modelos influencia diretamente o resultado final das orientações.

Aparentemente, o uso de ângulos de Euler parece não causar problemas, uma vez que parece razoável pensar que ao estabelecer uma ordem fixa de rotações resolveria o problema. Realmente, se este fosse o único objetivo de um sistema de RV controlado por dispositivos não convencionais, o uso de orientações fixas provavelmente resolveria a questão, entretanto, o problema encontra-se nas orientações mutáveis.

Em diversas ocasiões, não desejamos saltar de uma orientação para outra, e sim alterar pouco a pouco a representação para executar de forma suave uma operação de rotação maior. Sob este novo paradigma, os ângulos de Euler apresentam dois problemas principais.

1.3.1.2.2.3. Gimbal Lock

Existe um fenômeno não muito intuitivo com o qual desenvolvedores de RV se defrontam constantemente ao representar suas entidades e suas respectivas

orientações utilizando ângulos de Euler, no qual graus de liberdade são perdidos ao se executar determinadas combinações de rotações. Tal efeito é chamado Gimbal Lock.

Por exemplo, deseja-se realizar a seguinte combinação de rotações: 90º graus no eixo leste/oeste e outros 90º no eixo norte/sul. Entretanto, a convenção de ângulos de Euler nos diz que rotações no eixo norte/sul são realizadas primeiro.

Desta maneira, ao se executar uma rotação de 90º sobre o eixo leste/oeste, não há qualquer mudança nos outros eixos que possa nos levar a alterações no eixo norte/sul. Isso significa que a posição desejada jamais poderá ser alcançada utilizando ângulos de Euler, assim, enquanto o ângulo de rotação do eixo leste/oeste permanecer em 90º, um grau de liberdade é perdido e algumas orientações não poderão ser alcançadas.

Uma solução para o problema seria simplesmente representar a orientação através de uma matriz de rotação com relação à posição inicial e simplesmente multiplica-la por cada nova rotação aplicada ao modelo representado. Tal abordagem seria falha, pois como são utilizadas matrizes 3x3 para representar três graus de liberdade, dados redundantes são armazenados, assim, sucessivas multiplicações acumulam erros, fazendo com que o resultado final seja errôneo, deixando até mesmo de ser uma rotação. Tal problema poderia ser resolvido normalizando-se as matrizes a cada final de ciclo, entretanto, isso acarretaria mais custo computacional e imprecisões.

Uma outra abordagem é recalcular os três ângulos de Euler a cada pequena rotação. Contudo, esses novos ângulos não se associam com os ângulos antes da rotação, uma vez que para girar em torno de um eixo, em algumas situações é necessário executar uma rotação prévia sobre outro eixo. Desta maneira, próximos aos pontos de Gimbal Lock haveria saltos imprevistos.

Finalmente, a solução definitiva encontra-se em um sistema de representação em que as operações de rotação possam ser executadas de forma natural sob eixos aleatórios.

1.3.1.2.2.4. Interpolação de orientações

Outro grande problema do uso de ângulos de Euler encontra-se na interpolação de orientações, quando é desejado produzir uma seqüência de valores intermediários

entre duas outras rotações dadas. Mesmo que o Gimbal Lock não ocorra, não é nada intuitivo fazer com que um modelo execute um determinado caminho suave entre duas orientações.

Ao trabalhar com ângulos de Euler, a interpolação aplicada a cada um dos ângulos de rotação produzirá rotações independentes em torno desses eixos, ao invés de um movimento suave.

No caso do Gimbal Lock não ocorrer, o produto final poderá até ser o correto, entretanto, as orientações intermediárias não são devidas se o sistema adotado for a interpolação linear, produzindo movimentos estranhos e imprevisíveis.

Dadas duas posições no espaço tridimensional, existe uma infinidade de curvas que as ligam. O modelo 3-D poderia então traçar um caminho aleatório até chegar ao destino, entretanto, no caso de transição entre duas orientações, é indesejável que tal fenômeno aconteça.

A solução para este caso configura-se através de rotação simples ao redor de um único eixo, ou seja, dadas duas orientações, executa-se uma interpolação linear simples no ângulo de rotação em torno de um eixo adotado. Tal eixo não necessariamente existe e a parametrização de Euler não realiza de forma alguma, rotações sobre eixos arbitrários. O ideal é realizar rotações sobre eixos adequados e não seguindo caminhos arbitrários.

Algebricamente, seja um ponto no $\Re^3$ representado por um vetor $\vec{r} = (r_x, r_y, r_z)$ e seja $\rho_{\theta,n}$ uma rotação anti-horária de um ângulo θ em torno de um eixo que intercepta a origem definido por um vetor unitário $\vec{n} = (n_x, n_y, n_z)$. Deseja-se determinar uma expressão para $\rho(\vec{r})$, ou seja, para o vetor que representa o ponto obtido após aplicar em $\vec{r}$ a rotação ρ.

O problema pode ser resolvido decompondo $\vec{r}$ em suas componentes normal $\vec{r}_\perp$ paralela ao vetor $\vec{n}$, basta realizar o produto escalar entre $\vec{n}$ e $\vec{r}$. Assim obtém-se que:

$$\vec{r} = \vec{r}_\perp + \vec{r}_P \tag{1.27}$$

$$\vec{r}_P = (\vec{n} \cdot \vec{r})\vec{n} \tag{1.28}$$

$$\vec{r}_{\perp} = \vec{r} - \vec{r}_{P} = \vec{r} - (\vec{n} \cdot \vec{r})\vec{n} \qquad (1.29)$$

onde $\vec{r}$ é um vetor tridimensional a ser rotacionado, $\vec{r}_{\perp}$ a componente normal, $\vec{r}_{P}$ a componente paralela e $\vec{n}$ um vetor unitário sobre o qual ocorre a rotação.

A componente $\vec{r}_{P}$, naturalmente permanece inalterada por uma rotação em torno do eixo definido por $\vec{n}$, de forma que:

$$\rho(\vec{r}_{P}) = \vec{r}_{P} \qquad (1.30)$$

onde ρ é a rotação e $\vec{r}_{P}$ a componente paralela.

O problema restante é determinar qual resultado da rotação de $\vec{r}_{\perp}$. Sabe-se que, por definição, esta rotação ocorre num plano paralelo a $\vec{r}_{\perp}$, e perpendicular a $\vec{n}$. Logo, define-se o vetor $\vec{v}$ como:

$$\vec{v} = \vec{n} \times \vec{r}_{\perp} = \vec{n} \times (\vec{r} - \vec{r}_{P}) = \vec{n} \times \vec{r} - \vec{n} \times \vec{r}_{P} = \vec{n} \times \vec{r} \qquad (1.31)$$

tem-se que $\vec{n}, \vec{r}_{\perp}, \vec{v}$ formam um triedo direto, e em particular que $\vec{r}_{\perp}$ e $\vec{v}$ são perpendiculares e estão no plano onde ocorre a rotação. Além disso, como $\vec{n}$ é unitário, $\vec{v}$ terá a mesma norma que $\vec{r}_{\perp}$. Portanto, é imediato verificar que:

$$\rho(\vec{r}_{\perp}) = (\cos\theta)\vec{r}_{\perp} + (sen\theta)\vec{v} \qquad (1.32)$$

Somando os componentes temos que:

$$\rho(\vec{r}) = \rho(\vec{r}_{P} + \vec{r}_{\perp}) = (\cos\theta)\vec{r} + (1 - \cos\theta)(\vec{n} \cdot \vec{r})\vec{n} + (sen\theta)(\vec{n} \times \vec{r}) \qquad (1.33)$$

Assim conclui-se que o ponto resultante da realização de uma rotação $\rho(\theta, \vec{n})$ em um ponto $\vec{r}$ é:

$$\rho(\vec{r}) = (\cos\theta)\vec{r} + (1-\cos\theta)(\vec{n}\cdot\vec{r})\vec{n} + (sen\theta)(\vec{n}\times\vec{r}) \quad (1.34)$$

1.3.1.2.2.5. Quatérnios

Um quatérnio é definido por um conjunto de quatro números reais usualmente chamados de *a*, *b*, *c* e *d*. De forma similar à representação dos números complexos, os quatérnios possuem uma parte real e uma imaginária. Entretanto, os mesmos apresentam três componentes diferentes que formam a parte imaginaria (*i*, *j* e *k*).

Assim, um quatérnio é um conjunto da forma q = (*a,b,c,d*) ou,

$$q = a + bi + cj + dk \quad (1.35)$$

Pode-se também expressar um quatérnio de forma condensada:

$$q = (s, \vec{v}) \quad (1.36)$$

onde *s* é um escalar que representa a parte real e $\vec{v} = (v_x, v_y, v_z)$ é um valor de três componentes que representam a parte imaginária.

Análogo às propriedades de i nos números complexos, os valores de *i, j* e *k* constituem-se como generalizações. Assim temos que:

$$i^2 = j^2 = j^2 = -1 \quad (1.37)$$

e ao multiplicar imaginários de natureza diferente temos que:

$$ij = k \quad (1.38)$$

$$ji = -k \quad (1.39)$$

Nota-se que as multiplicações de dois quatérnios, ao contrário dos complexos, não são comutativas, já que se trata de rotações tridimensionais.

A relevância da utilização de quatérnios para aplicar rotações em modelos tridimensionais reside nas operações de multiplicação, onde pode-se notar que o produto de dois quatérnios q_1 e q_2 é representado por:

$$\begin{aligned} q_1 q_2 &= a_1 a_2 - (b_1 b_2 + c_1 c_2 + d_1 d_2) + a_1 (b_2 i + c_2 j + d_2 k) \\ &+ a_2 (b_1 i + c_1 j + d_1 k) + (c_1 d_2 - d_1 c_2) i + (d_1 b_2 - b_1 d_2) j + (b_1 c_2 - c_1 b_2) k \end{aligned} \quad (1.40)$$

e de forma escalar:

$$q_1 q_2 = (s_1, \vec{v}_1)(s_2, \vec{v}_2) = (s_1 s_2 - \vec{v}_1 \cdot \vec{v}_2, s_1 \vec{v}_2 + s_2 \vec{v}_1 + \vec{v}_1 \times \vec{v}_2) \quad (1.41)$$

1.3.1.2.2.6. Representando Rotações utilizando Quatérnios

O ponto $\vec{r} = (r_x, r_y, r_z)$ sobre o qual é aplicada uma rotação é representado pelo quatérnio $p = (0, \vec{r})$ tendo a parte real nula e a rotação aplicada sobre o ponto $\vec{r}$ é representado por um quatérnio $q = (s, \vec{v})$.

Desta maneira, a rotação de p por q pode ser expressa da forma:

$$R_q(p) = qpq^{-1} \quad (1.42)$$

ou

$$R_q(p) = qp\bar{q} \quad (1.43)$$

sendo que *q* é unitário e seu inverso é igual a seu conjugado. Assim temos que:

$$qp\bar{q} = (s, \vec{v})(0, \vec{r})(s, -\vec{v}) = (0, s^2 \vec{r} - (\vec{v} \cdot \vec{v})\vec{r} + 2(\vec{v} \cdot \vec{r})\vec{r} + 2s\vec{v} \times \vec{r}) \quad (1.44)$$

Onde

$$\vec{a} \cdot (b \times \vec{a}) = 0 \quad (1.45)$$

$$\vec{a} \times \vec{b} = -\vec{b} \times \vec{a} \quad (1.46)$$

$$\vec{a}\times(\vec{b}\times\vec{c}) = (\vec{a}\cdot\vec{c})\vec{b} - (\vec{a}\cdot\vec{b})\vec{c} \qquad (1.47)$$

Como $q = (s,\vec{v})$ é unitário, temos que $s^2 + \left|\vec{v}\right|^2 = 1$, ou seja, sempre existe um ângulo θ tal que $s = \cos\theta$ e $\left|\vec{v}\right| = sen\,\theta$. Assim temos que:

$$q = (s,\vec{v}) = (\cos\theta, sen\,\theta\,\vec{n}), \qquad \left|\vec{n}\right| = 1 \qquad (1.48)$$

e

$$\begin{aligned} q\vec{p}\bar{q} &= (0, s^2\vec{r} - (\vec{v}\cdot\vec{v})\vec{r} + 2(\vec{v}\cdot\vec{r})\vec{r} + 2s\vec{v}\times\vec{r}) = \\ &= (0, (\cos 2\theta)\vec{r} + (1-\cos 2\theta)(\vec{n}\cdot\vec{r})\vec{n} + (sen2\theta)\vec{n}\times\vec{r}) \end{aligned} \qquad (1.49)$$

onde

$$\vec{a}\cdot\vec{a} = \left|\vec{a}\right|^2 \qquad (1.50)$$

$$\cos^2\theta - sen^2\,\theta = \cos 2\theta \qquad (1.51)$$

$$2sen^2\theta = (1-\cos 2\theta) \qquad (1.52)$$

$$2\cos\theta sen\theta = sen2\theta \qquad (1.53)$$

Desta maneira, para compor uma combinação de duas rotações representadas por q_1 e q_2 aplica-se:

$$R_{q2}(R_{q1}(p)) = R_{q2}(q_1\bar{p}q_1) = q_2q_1p\bar{q}_1\bar{q}_2 = q_3pq_4 \qquad (1.54)$$

ou então:

$$q_4 = \bar{q}_3 \Rightarrow q_3pq_4 = q_3p\bar{q}_3 = R_{q3}(p) \qquad (1.55)$$

Assim, pode-se concluir que a composição de rotações pode ser realizada de maneira natural pela simples álgebra dos quatérnios, ou seja, com dois

quatérnios unitárioe e seus respectivos ângulos e eixos distintos, pode-se representar uma rotação multiplicando-se os quatérnios.

Aplicando-se os conceitos à rotação anteriormente descrita, temos que a primeira rotação é:

$$\begin{aligned} q_1 &= (\cos(90/2), sen(90/2)(0,1,0) \\ &= (\sqrt{2/2}, (\sqrt{2/2})(0,1,0)) \\ &= (\sqrt{2/2}, (0, \sqrt{2/2}), 0)) \end{aligned} \qquad (1.56)$$

e a segunda:

$$\begin{aligned} q_2 &= (\cos(90/2), sen(90/2)(1,0,0) \\ &= (\sqrt{2/2}, (\sqrt{2/2})(1,0,0)) \\ &= (\sqrt{2/2}, (\sqrt{2/2}), 0, 0)) \end{aligned} \qquad (1.57)$$

e finalmente:

$$q_3 = q_1 q_2 = (1/2, (1/2, 1/2, 1/2)) \qquad (1.58)$$

que correspondente a uma rotação anti-horária de $\theta = 2\arccos(1/2) = 120°$ em torno do eixo definido por $(1/2, 1/2, 1/2)$.

1.3.1.2.3. Escala

A matriz de expressão para as transformações de escala de uma posição *P=(x, y,z)* relativa às coordenadas de origem pode ser descrita como:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (1.59)$$

ou:

$$P' = S \cdot P \tag{1.60}$$

Onde os parâmetros de escala s_x, s_y e s_z recebem qualquer valor positivo. Expressões explícitas para transformações de escala relativas à origem são:

$$x' = x \cdot s_x \tag{1.61}$$

$$y' = y \cdot s_y \tag{1.62}$$

$$z' = z \cdot s_z \tag{1.63}$$

Transformações de escala a partir da matriz (1.27) alteram o tamanho de um objeto e reposiciona o mesmo relativo à origem das coordenadas, preservando a geometria oriinal do objeto, através de escala uniforme ($s_x = s_y = s_z$). Paralelamente, se os parâmetros de escala não forem iguais, dimensões relativas do objeto serão alteradas.

Para mudanças de escala relativas a um ponto fixo da cena virtual (x_f, y_f, z_f), é utilizada a seguinte seqüência de transformações: 1) Transladar o ponto fixo para a origem; 2) Escalar o objeto relativo às coordenadas de origem utilizando a equação (1.27) e 3) Transladar o ponto fixo de volta à posição original.

A matriz de transformação de mudança de escala em relação a um ponto fixo pode ser definida por:

$$T(x_f, y_f, z_f) \cdot S(s_x, s_y, s_z) \cdot T(-x_f, -y_f, -z_f) = \begin{bmatrix} s_x & 0 & 0 & (1-s_x)x_f \\ 0 & s_y & 0 & (1-s_y)y_f \\ 0 & 0 & s_z & (1-s_z)z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.64}$$

1.3.1.2.4. Reflexão

Uma reflexão tridimensional pode ser processada em relação a um determinado eixo de reflexão ou em relação a um plano de reflexão específico.

Similarmente às reflexões bidimensionais, as reflexões relativas a um determinado eixo de coordenadas são equivalentes a rotações de 180 graus sobre o eixo. Reflexões sobre determinados ângulos são equivalentes a rotações de 180 graus em espaço de quatro dimensões (4-D).

Quando o plano de reflexão é um plano de coordenadas (*xy, xz* ou *yz*), a transformação é dita como conversão entre as regras da "mão direita" e "mão esquerda". Esta transformação altera o sinal das coordenadas *z*, deixando as coordenadas *x* e *y* com os valores originais. A matriz que representa a operação de reflexão do plano *xy* é dada por:

$$RF_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.65}$$

Matrizes de transformação para inversão de valores de *x* e *y* são definidas similarmente como reflexões relativas aos planos *yz* e *xz*, respectivamente. Reflexões sobre outros planos podem ser implementadas sob combinações de rotações e reflexões.

1.3.1.2.5. Alongamento e Encurtamento

Transformações de alongamento/encurtamento são utilizadas para modificar a geometria dos objetos. São também utilizadas na visualização tridimensional para simular transformações de projeções. Como exemplo da operação de alongamento/encurtamento tridimensional, representa-se o efeito da mesma sob o eixo *z* através da matriz:

$$SH_z = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.66}$$

Onde *a* e *b* são representados por números reais. O efeito desta matriz de transformação é alterar os valores das coordenadas de *x* e *y* por um montante proporcional ao valor de *z*, enquanto o mesmo permanece com o valor original. Fronteiras de planos perpendiculares ao plano *z* são então deslocadas mediante valores proporcionais ao mesmo.

1.3.1.2.6. Matrizes Ortogonais

Matrizes de transformação no campo de aplicações de Computação Gráfica e Realidade Virtual utilizam, em sua grande maioria, possuem dados ortogonais, onde a transposta da matriz de transformação é igual à sua inversa:

$$M^{-1} = M^{T} \tag{1.67}$$

Matrizes ortogonais possuem a propriedade de preservar comprimentos e ângulos quando são usadas como vetores de transformação. A matriz *M* preserva seu comprimento se para cada vetor *P* tem-se que:

$$\|MP\| = \|P\| \tag{1.68}$$

Para que o ângulo seja também preservado, é necessário que, para dois vetores P_1 e P_2, tem-se que:

$$(MP_1)\cdot(MP_2) = P_1 \cdot P_2 \tag{1.69}$$

1.3.2. Modelagem Imersiva

1.3.2.1. Estereoscopia

A percepção de profundidade é um processo mental sofisticado, que combina indicadores fisiológicos e psicológicos (Vester e Toutin, 1997).

Na estereoscopia, duas cenas de uma mesma área são adquiridas sob diferentes perspectivas, através de variações de pontos de vista e/ou incidências, de modo a permitir a sensação de imersão à medida que as perspectivas se fundem no cérebro. Através da correlação entre as duas cenas, são obtidas as chamadas paralaxes (Paradella *et al*, 2003).

A estereoscopia depende de dois principais princípios: (a) a paralaxe estereoscópica, que se relaciona com a aparente mudança da posição dos objetos na cena quando vistos de diferentes pontos, e (b) o ângulo de intersecção estereoscópica, medido entre as linhas formadas pelos pontos de vista adotados, relativas às visões do olho esquerdo e direito, sobre um determinado ponto na cena. Quanto maior for o ângulo de intersecção, mais próximos de nós os objetos parecem estar (Brito e Coelho, 2002).

Um exemplo simples que pode ser considerado é a observação de diferentes objetos através da janela de um avião. Os objetos que estão distantes aparentam mover-se muito pouco em relação ao referencial, enquanto que os mais próximos da janela aparentam mover-se em distâncias muito maiores, como demonstradas pela Figura 2.

A Figura 3 mostra duas tomadas distintas de um mesmo ponto, onde nota-se a presença de superfícies mais elevadas do que o nível do solo. Assim, é possível observar diferentes distâncias entre picos e bases. A Figura 4 descreve, para dois pontos de uma elevação no terreno, o efeito posicional em duas cenas diferentes.

Medindo as coordenadas dos pontos *a* e *b* nas duas imagens, paralelamente à linha de vôo, pode-se estabelecer uma relação, definida pela primeira equação da paralaxe, denominada paralaxe absoluta:

$$pa = xa - x'a \tag{1.70}$$

onde os deslocamentos ocorrem paralelamente à linha de vôo (no caso, eixo x).

Para um par estereoscópico, a figura será uma linha, materializada por quatro vértices (dois pontos principais e dois conjugados).

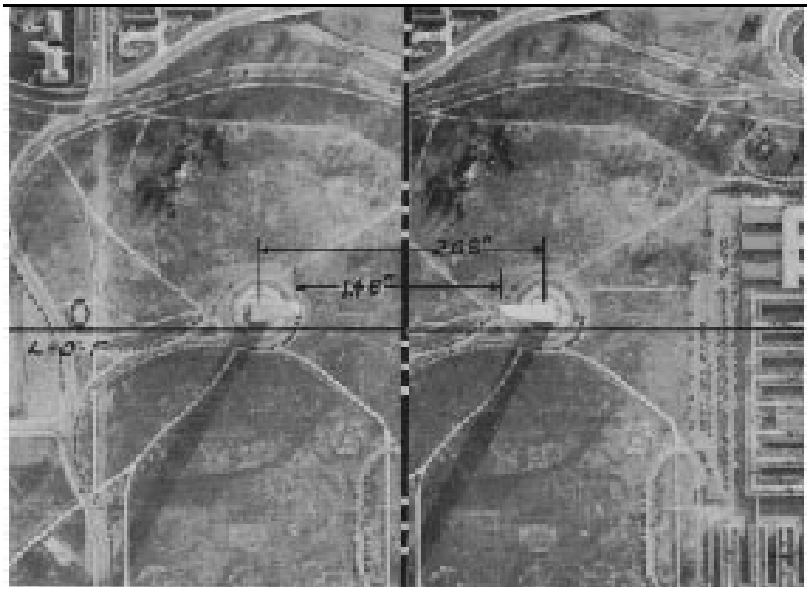


Figura 3 - Foto aérea de monumento em dois instantes diferentes, onde pode-se notar a paralaxe lateral relativa às diferenças de posicionamento dos objetos nas imagens.

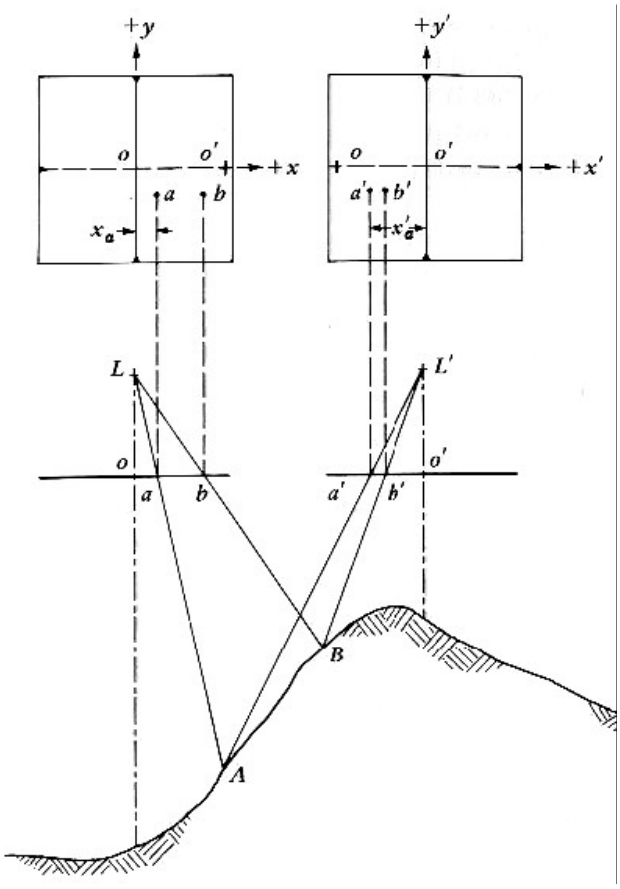


Figura 4 - Representação de deslocamentos devido às diferentes distâncias entre dois pontos, caracterizando a paralaxe (Lillesand e Kiefer, 2000)

A formulação expressa a seguir resume os princípios básicos da medição de paralaxe. A Figura 5(a) mostra a utilidade da paralaxe para a determinação da distância em um ponto. Sabendo-se a altitude da estação *H*, a base aérea, ou *B*, que é a distância entre as estações no sentido da linha de vôo, a distância focal *f* da câmara utilizada e a paralaxe do ponto *a*, pode-se facilmente calcular a altitude de *A*, *hA*. A Figura 5(b) é resultante da superposição de dos triângulos L e L' de modo a mostrar graficamente a paralaxe *pa*.

Por semelhança de triângulos tem-se:

$$\frac{p_a}{f} = \frac{B}{H - h_A} \tag{1.71}$$

onde $f \neq 0$ e $H - h_A \neq 0$

$$H - h_A = \frac{Bf}{p_a} \tag{1.72}$$

e, consequentemente:

$$h_A = H - \frac{Bf}{p_a} \tag{1.73}$$

onde $P_a \neq 0$.

Também por semelhança de triângulos pode-se escrever:

$$\frac{X_A}{H - h_A} = \frac{x_a}{f} \tag{1.74}$$

$$X_A = x_a \frac{H - h_A}{f} \tag{1.75}$$

onde $f \neq 0$ e $H - h_A \neq 0$.

Substituindo-se de (1.42) a (1.43), tem-se:

$$X_A = B\frac{x_a}{p_a} \tag{1.76}$$

Analogamente:

$$Y_A = B\frac{y_a}{p_a} \tag{1.77}$$

onde $P_a \neq 0$.

As equações (1.44) e (1.45), descritas anteriormente, são comumente conhecidas como as equações da paralaxe. Em algumas aplicações, entretanto, deseja-se apenas saber a distância entre dois pontos. Para esses casos, utiliza-se a expressão:

$$\Delta h = \Delta p H^2 / p_a \tag{1.78}$$

onde Δh é a diferença em elevação entre dois pontos cuja diferença de paralaxe é Δp, H é a altura de vôo sobre o ponto de menor altitude e p_a é a paralaxe do ponto mais alto, com $P_a \neq 0$.

Análogo ao processo de fotografias aéreas, o funcionamento dos HMD's (*Head Mounted Displays*) baseia-se nos princípios da estereoscopia, uma vez que os mesmos exibem a cena sob duas telas independentes, representando as visões do olho esquerdo e direito.

Como um par estereoscópico de imagens, as cenas exibidas nas telas de *LCD* podem ser interpretadas como bidimensionais estáticas, possibilitando a inserção de conceitos imersivos aplicáveis às imagens 2-D, como a imersão sob a fusão das perspectivas, baseadas em paralaxe.

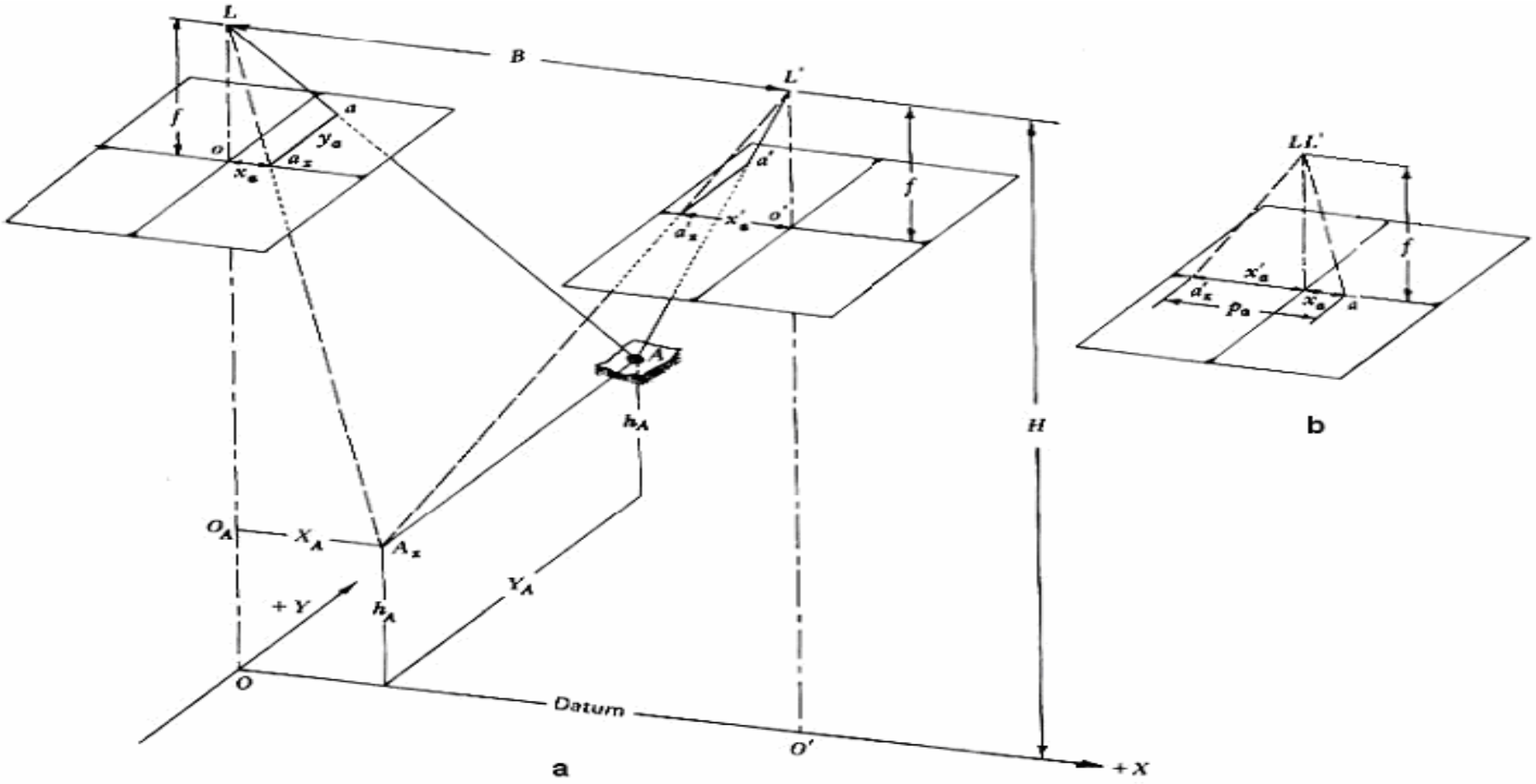


Figura 5 - Relacionamento entre as paralaxes de um ponto A: (a) Utilização da paralaxe para a determinação da distância de um ponto e (b) Resultado da superposição dos triângulos L e L´ (Lillesand e Keifer, 2000)

1.3.2.2. Iluminação, Reflexão e Atenuação

A iluminação das cenas sintéticas tridimensionais, bem como a capacidade de reflexão e atenuação das luzes pelos diversos materiais, trata-se de um fator fundamental no processo de percepção de profundidade dos objetos do ambiente. Sem os recursos de iluminação, os modelos apresentariam a mesma tonalidade sobre todos os pixels, provocando o efeito de bidimensionalidade. Assim, torna-se necessária a utilização de diversas fontes de luz, sob diversos parâmetros, para destacar os verdadeiros formatos das geometrias apresentadas na cena, ressaltando relevos e bordas, contribuindo para o efeito de profundidade.

1.3.2.2.1. Fontes de luz Ambiente

O termo luz ambiente refere-se à luz que foi refletida tantas vezes, sobre tantos objetos, que sua direção original não pode ser determinada. Assim, pode-se dizer que a luz ambiente é não direcional e ilumina igualmente todos os modelos da cena, com um mínimo nível de luz incidente. Pode-se expressar a luz ambiente como uma simples constante global para toda a cena:

$$I_{refletida} = I_{ambiente} \tag{1.79}$$

multiplicando-se pelo fator escalar de Material

$$I_{refletida} = I_{ambiente} \cdot k_{ambiente} \tag{1.80}$$

1.3.2.2.2. Fontes de luz Direcional

Luzes direcionais são utilizadas para representar fontes de luz como o sol, onde os raios são considerados paralelos. Considerando-se que tal fonte não tem posição no espaço, as mesmas têm alcance infinito e sua intensidade não diminui com a distância. Desta maneira pode-se representar a incidência direcional por:

$$I_{direcional} = I_0 \tag{1.81}$$

onde I_0 é a intensidade da luz em candelas (cd).

1.3.2.2.3. Fontes de luz Pontuais

As fontes pontuais de iluminação são aquelas que emitem luz em todas as direções a partir de um único ponto no espaço. Todas as bibliotecas gráficas implementam tal funcionalidade e permitem controlar a intensidade da luz emitida pela fonte através da recíproca dos polinômios quadráticos. Supondo que a fonte de luz seja um ponto *P*, a intensidade *C* atingindo um ponto no espaço *Q*, é dada por:

$$C = \frac{1}{k_c + k_l d + k_q d^2} C_0 \tag{1.82}$$

onde C_0 é a cor da luz, *d* é a distancia entre a fonte *P* e o ponto *Q*, e as constantes $k_c, k_l \, e \, k_q$ são os coeficientes de ateuação constante, linear e quadrática, respectivamente, com $k_c + k_l d + k_q d^2 \neq 0$.

1.3.2.2.4. Fontes de luz *Spot*

Análogo ao processo de iluminação pontual, as fontes *Spot* têm sua intensidade de luz atenuada pela distância, diferindo apenas na presença do chamado efeito *spot*:

$$C = \frac{\max\{-R.L, 0\}^p}{k_c + k_l d + k_q d^2} C_0 \tag{1.83}$$

onde C_0 é a cor da luz, *d* é a distancia entre a fonte *P* e o ponto *Q*, *R* é a direção da luz *spot*, *p* controla a concentração da luz *spot*, as constantes $k_c, k_l \, e \, k_q$ são os coeficientes de ateuação, com $k_c + k_l d + k_q d^2 \neq 0$, e *L* é a unidade de medida para a direção dos raios de luz, onde:

$$L = \frac{P - Q}{\|P - Q\|} \tag{1.84}$$

onde $\|P - Q\| \neq 0$.

1.3.2.2.5. Reflexão Difusa

Pode-se denotar a reflexão difusa *K* como a cor da luz emitida pelo objeto, ou seja, a luz refletida ao observador a partir de um ponto Q na superfície. A fórmula de reflexão difusa é escrita em função da intensidade C_i de cada uma das *n* luzes que iluminam o ponto *Q*. A luz refletida, em todas as direções, é modularizada pela cor de reflexão difusa *D* da superfície. Adicionando as contribuições de *n* fontes de luz e considerando a intensidade do ambiente *A*, pode-se expressar o componente de difusão como:

$$K_{difusa} = DA + D\sum_{i=1}^{n} C_i \max\{N \cdot L_i, 0\} \tag{1.85}$$

onde o vetor unitário L_i, reflete por *Q* a *i*-ésima fonte de luz.

1.3.2.2.6. Reflexão Especular

Em adição à reflexão difusa uniforme, as superfícies tendem a refletir a luz fortemente, em uma única direção, ao longo do trajeto dado pela reflexão dos raios incidentes em seus vetores normais.

Para calcular a contribuição especular com uma única fonte de luz, utiliza-se:

$$K_{especular} = SC \max\{R \cdot V, 0\}^m \text{ com } (N \cdot L > 0) \tag{1.86}$$

onde *S* é a cor de reflexão especular da superfície, *V* é o vetor do observador, *R* é vetor de reflexão direta, *C* é a intensidade da luz incidente, *N* é o vetor normal, *L* é o vetor de direção da luz e *m* é chamado de expoente especular, responsável por

controlar a atenuação da luminosidade especular. A expressão $(N \cdot L_i > 0)$ garante que faces distantes não tenham reflexão especular.

Para uma segunda formulação do índice de reflexão especular, utiliza-se de um vetor *H*, posicionado exatamente entre o vetor de visualização *V* e o vetor de direção da luz *L*. Os pontos especulares mais luminosos ocorrem quando *H* aponta para a direção do vetor normal *N*. Desta maneira, o índice de reflexão especular é denotado por:

$$K_{especular} = S\sum_{i=1}^{n} C_i \max\{N \cdot H_i, 0\}^m \quad \text{com} \quad (N \cdot L_i > 0) \qquad (1.87)$$

onde *H*, sob incidência da i-ésima fonte de luz, é denotado por:

$$H_i = \frac{L_i + V}{\|L_i + V\|} \qquad (1.88)$$

onde $\|L_i + V\| \neq 0$.

1.4. Os Dispositivos de Realidade Virtual

A interface em RV envolve um controle tridimensional altamente interativo de processos computacionais. O usuário entra no espaço virtual das aplicações e visualiza, manipula e explorar os dados da aplicação em tempo real, usando seus sentidos, particularmente os movimentos naturais tridimensionais do corpo. A grande vantagem é que o conhecimento intuitivo do usuário sobre o mundo físico pode ser transportado para o mundo virtual.

Como conseqüência do advento da RV, surgiu a necessidade de se redefinir o paradigma de interface homem-computador. O sistema tradicional mouse-teclado-monitor foi substituído por dispositivos não-convencionais, que permitem maior imersão do usuário no ambiente sintético e o manuseio de todas as potencialidades dessa nova tecnologia (Kirner, 1995).

1.5.1. Dispositivos de saída de dados

A maioria das aplicações de RV é baseada no isolamento dos sentidos, principalmente a visão. Assim, cabe ao *hardware* de RV de saída de dados estimular tais sentidos. A saída dos sistemas de RV, seja ele imersivo (HMD's) ou parcialmente imersivo (monitores), tem como preocupação principal a estereoscopia, seja ela passiva, como a polarização, anaglifos e difração de luz, ou ativa, como os óculos obturadores (Gattas *et al*, 2004).

1.5.1.1. Dispositivos convencionais de saída de dados

Os dispositivos de convencionais de saída de dados são periféricos de comum utilização, destinados à visualização e análise de sistemas de propósito geral. São eles: monitor de vídeo, impressoras, auto-falantes, etc. Envolvendo sistemas de Realidade Virtual, os dispositivos convencionais de saída de dados podem contribuir para o desenvolvimento de ambientes interativos e semi-imersivos, porém, não são capazes de prover realismo e o envolvimento tal qual fornecidos por sistemas virtuais dotados de dispositivos não-convencionais, os quais serão amplamente utilizados no presente projeto e descritos minuciosamente na seção a seguir.

1.5.1.2. Dispositivos não-convencionais de saída de dados

Os dispositivos não-convencionais de saída de dados são responsáveis em fornecer grande parte do efeito imersivo ao sistema de RV. Assim, tais dispositivos, implementados sob *interfaces* intuitivas, são capazes de transpor o usuário à cena sintética, tornando real sua experiência.

1.5.1.2.1. *Head Mounted Displays*

O vídeo-capacete (*Head-Mounted Display*, HMD) é um dos dispositivos de interface para RV mais populares, por tratar-se do dispositivo não-convencional de saída de dados que mais isola o usuário do mundo real. Este dispositivo é constituído basicamente de uma ou duas minúsculas telas e um conjunto de lentes especiais. Com

duas telas, a tecnologia pode ser utilizada para exibir imagens estereoscópicas, apresentando os respectivos pontos de vista de cada olho para cada tela, contribuindo para o efeito imersivo.

As lentes ajudam a focalizar imagens que estão a alguns milímetros dos olhos do usuário, ajudando também a ampliar o campo de visão do vídeo. Os HMD's funcionam também como um dispositivo de entrada de dados, porque contém sensores de rastreamento que medem a posição e orientação da cabeça, transmitindo esses dados para o computador. Consequentemente, o computador gera uma seqüência de imagens por quadro correspondente às ações e perspectivas do usuário (Gradescki, 1994).

Os *HMD*'s são construídos, normalmente, usando três tipos de telas: os monitores de TV (CRT), os monitores de cristal líquido (LCD), os mais usados atualmente, e os de diodo emissores de luz orgânicos (OLED).

1.5.1.2.1.1. *Displays* CRT

Os monitores de TV, em função da avançada tecnologia disponível nesta área, podem exibir imagens de alta resolução (1280x1024 pixels) com uma qualidade de cor excelente, mesmo em pequenas dimensões, sob campo de visão horizontal de até 127 graus. Entretanto, são relativamente pesados, volumosos e colocam altas voltagens muito próximas à cabeça do usuário (> 100v) (Kalawski, 1993). O funcionamento básico de um monitor CRT baseia-se na emissão de um raio de elétrons (raios catódicos) por um canhão de elétrons. O raio passa por um sistema de ajuste de foco (anodo) e deflexão, que direciona o raio para posições específicas de uma tela fosforecente. O fósforo então emite um pequeno ponto de luz em cada posição atingida pelo raio catódico. Devido ao rápido enfraquecimento da luz, a maioria dos HMD's CRT possuem um sistema de *refresh*, responsáveis por redesenhar a imagem repetidas vezes na tela. As cores na tela podem ser representadas através de duas maneiras: por uma dupla camada de fósforo, vermelha e verde, onde as cores exibidas na tela dependem de quanto os raios catódicos penetram nestas camadas e através de máscaras coloridas, posicionadas no caminho do raio. Tais máscaras possuem três pontos de cor em cada posição de pixel (verde, vermelho e azul) e três canhões de elétrons, cada um para um ponto de cor. Quando os raios

atravessam a máscara, eles ativam o triângulo de cores, o qual aparece em um pequeno ponto na tela. Cada cor específica pode ser alcançada por emissões especificas de raios catódicos (Hearn e Baker, 1997).

Os HMD's CRT usam dois Monitores de Raios Catódicos (CRT) que são posicionados nas laterais do HMD. Refletores de imagens são usados para dirigir a cena para o olho do usuário (Lane, 1993) A Figura 6 demonstra a emissão de raios catódicos e a Figura 7 demonstra um HMD com *display* CRT.

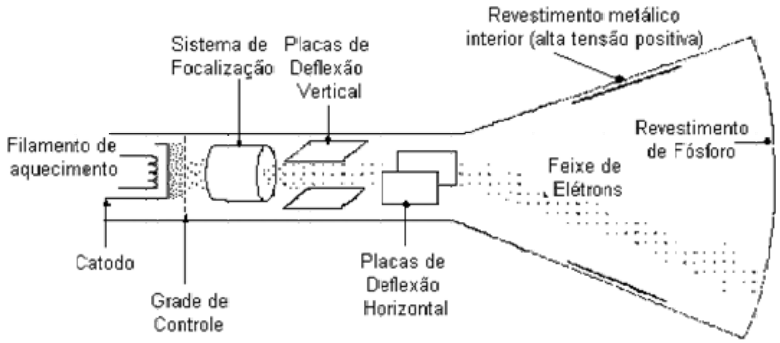


Figura 6 – Representação do funcionamento de um CRT, partindo da emissão de raios catódicos, passando por sistemas de focalização, deflexão e controle, até atingir a tela fosforecente (Mano, 1998)

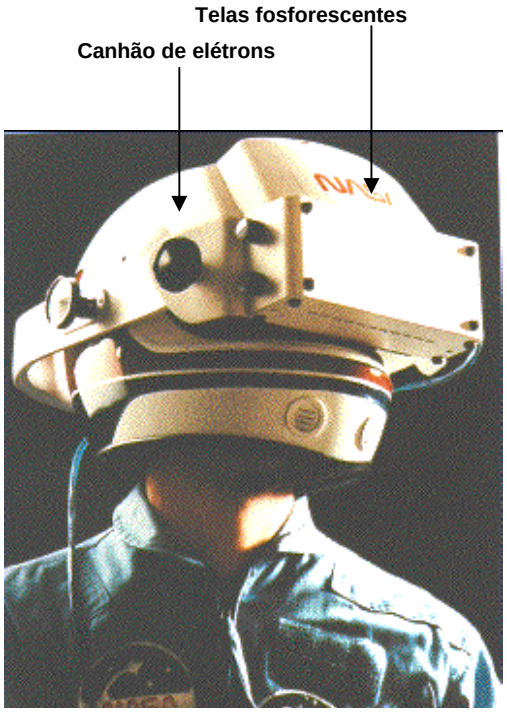


Figura 7 - HMD CRT de 2 telas desenvolvido na NASA, onde o canhão de elétrons é posicionado nas laterais do dispositivo, expondo o usuário a altas voltagens próximas da cabeça (Kalawski, 1993).

1.5.1.2.1.2. *Displays* LCD

Os HMD's LCD's, por sua vez, são leves e podem ser usados com pequenas voltagens (entre 20 e 100v). Sua resolução espacial em monitores pequenos pode variar desde extremamente baixa VGA (640x480 pixels) (Darpa, 1996), até uma amostragem satisfatória SXGA (1280x1024 pixels) (Vrlogic, 2007) com baixo ângulo de visão no campo horizontal (60 graus). Este tipo de HMD usa tecnologia LCD para exibir a cena, através da emissão de luz polarizada por um material de cristal líquido.

O termo cristal líquido atribuído a tal material refere-se ao fato de que os componentes do mesmo possuem uma disposição cristalina de moléculas, fluindo como líquido. O display é construído com cristal líquido nemático, duas superfícies de vidro, contendo polarizadores de luz e separadas pelo cristal líquido. Uma camada horizontal e outra vertical de condutores transparentes são atreladas às superfícies de vidro e a interseção dos condutores define a posição de um pixel. Quando uma luz polarizada ultrapassa todo o material, ela é rotacionada para que ultrapasse a outra superfície polarizadora e ative os pixels. Quando os *pixels* de cristal líquido são ativados, bloqueiam a passagem de luz. Milhares desses *pixels* são localizados em duas matrizes para cada exibição. Depois que o cristal líquido bloqueia a passagem de luz para exibir a cena, a luz deve ser refletida da matriz LCD para os olhos, provendo brilho para a cena. As cores podem ser obtidas em processo semelhante ao CRT, onde são utilizadas máscaras com as cores: vermelho, verde e azul, responsáveis por filtrar as posições específicas de cada pixel. A Figura 8 demonstra o esquema dos componentes dos monitores LCD (Hearn e Baker, 1997) e a Figura 9 demonstra um HMD com *display* LCD.

1.5.1.2.1.3. *Displays* OLED

Já os monitores dos HMD's OLED, os mais recentes do mercado, trabalham de maneira semelhante aos LCD's, entretanto, são baseados em diodos orgânicos emissores de luz e não necessitam de luz traseira para funcionar, acarretando um menor consumo de energia à baixa voltagem. Esta tecnologia possui telas planas muito mais finas, leves e baratas que as atuais telas de LCD. A idéia é usar diodos orgânicos, compostos por moléculas que emitem luz ao receberem uma carga elétrica

através de filamentos metálicos que conduzem os impulsos elétricos a cada célula. Quando uma voltagem é aplicada aos eletrodos, a carga começa a se mover no dispositivo sob influência do campo elétrico, assim, os elétrons deixam o catodo e as cargas positivas deixam o anodo em direções opostas. A combinação das cargas, na camada emissora, leva à criação de um fóton de moléculas emissoras, que sob energia elétrica gera a luz.

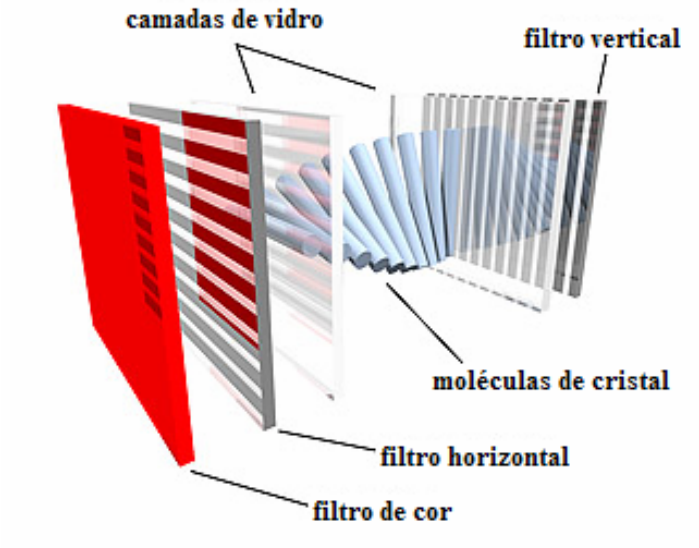


Figura 8- Esquema dos componentes formadores dos monitores LCD, onde a luz polarizada é rotacionada para ativar os pixels e assim bloquear a passagem de luz. Posteriormente, a luz deve ser refletida da matriz LCD para os olhos, provendo brilho para a cena (GeorgiaTech, 2006)

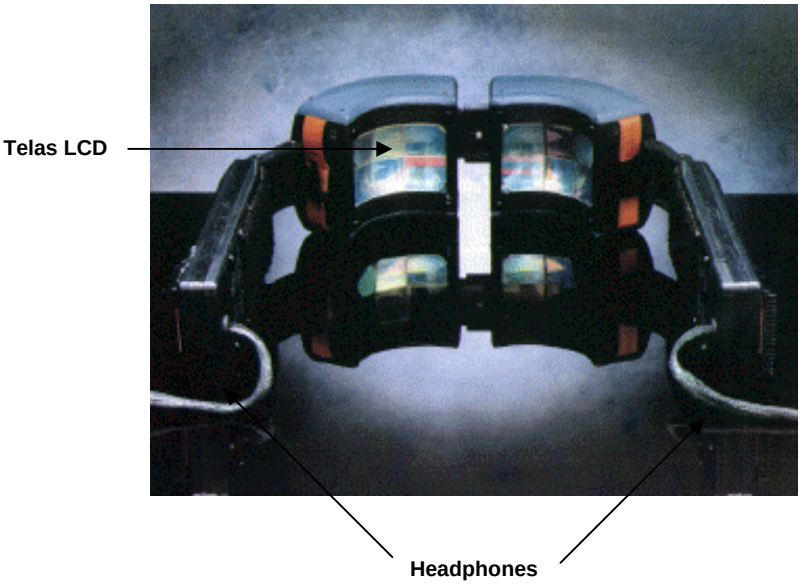


Figura 9 – Exemplo de HMD com duas telas LCD e dois headphones (Darpa, 1996)

A vantagem é que ao contrário dos diodos tradicionais, estas moléculas podem ser diretamente aplicadas sobre a superfície da tela. Como desvantagens apresentam uma rápida degradação do material que compõe as telas, não suportam resolução maior que 800x600 pixels e têm baixo ângulo de visão horizontal (42 graus) (Daeyang, 2005). A Figura 10 demonstra a movimentação de cargas para a formação da luz e a Figura 11 mostra um HMD com *display* OLED.

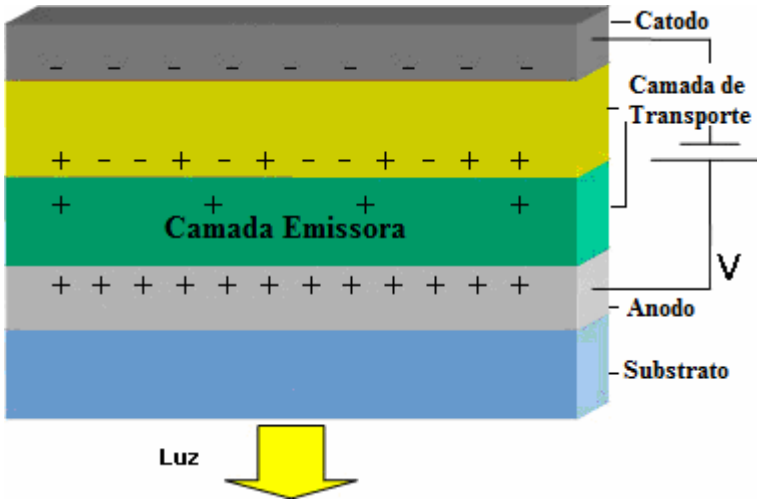


Figura 10 – Representação da movimentação das cargas positivas e negativas para a formação da luz, aplicada sobre a superfície da tela (Nieto, 2006)

Figura 11 – Representação de HMD OLED com duas telas e dois headphones (Vrealities, 2007)

O correto funcionamento de um HMD baseia-se em duas principais diretivas: o posicionamento do plano de imagem e a disparidade focal (VResources, 2007).

Para entender o processo, faz-se necessária compreensão da teoria do stress biológico humano, a qual diz que notamos a presença de um plano de imagem localizado a alguns metros dos olhos do observador. O valor 2 metros representa o melhor ponto de operação pelas seguintes razões. Esta distância é utilizada para minimizar o stress muscular causado pela grande variação entre a distância do plano da imagem e a distância do objeto virtual.

Com o plano da imagem a 2 metros, é possível exibir um objeto a 1 metro de distância e ter 5% de variações do esforço necessário. Além disso, todos os objetos exibidos atrás do plano de imagem ainda estarão próximos do foco necessário desde que a variação do esforço a partir de 2 metros até o infinito seja sempre inferior a 5%. Em modelos mais recentes, o esforço muscular absoluto nunca ultrapassa 10%, sob qualquer circunstancia, não importando a posição do objeto virtual.

Assim, os 2 metros de distância do plano de imagem permitem a visualização de uma imagem relativamente clara, sob uma escala de 1 metro até o infinito, mesmo com um foco óptico fixo, como na maioria dos HMD´s encontrados no mercado. Evidente que mesmo com um esforço mínimo, ainda não é o ideal. Assim, pesquisadores da área buscam desenvolver um sistema óptico adaptativo, capaz de alterar a distância do plano da image dinamicamente.

O Segundo principal ponto relativo aos HMD´s trata-se do campo de visão (FOV), e a sua relação com a mínima distância onde o objeto se apresenta, considerando que as ações devem ocorrer de forma com que a magnitude do campo de visão e os limites humanos devem trabalhar em conjunto.

Dependendo do campo de visão e da distância entre o plano de imagem e o observador, o mínimo ponto de aparição de um objeto virtual será, geometricamente falando, variável. Se o objetivo do uso de HMD´s na aplicação for um nível de imersão razoável, deve assegura-se que o campo de visão seja suficiente, levando em conta a distência do plano de imagem, para permitir a geração de imagens dentro dos limites fisiológicos do ser humano.

Geralmente, pode-se afirmar que o campo de visão de um HMD não consta como fator limitante, considerando a distância mínima de aparição de um objeto virtual na cena. Assim, o limite é imposto pelo máximo stress suportado pelo usuário. Desta maneira, 1 metro ocorrerá no pior caso.

1.5.1.2.2. Projetores

Os projetores desempenham um papel crucial nos sistemas de RV imersivos ou parcialmente imersivos. O brilho e a alta resolução são fatores que melhoram a qualidade da imagem, assim aumentam a sensação de imersão. Alguns projetores possuem a capacidade de gerar imagens para estereoscopia ativa e outros, através da combinação com outro projetor, são capazes de fornecer estereoscopia passiva. Atualmente existem três tipos de projetores: CRT, LCD e DLP (Gattass *et al*, 2004).

Os projetores CRT, os mais antigos, possuem três canhões de elétrons, responsáveis pela geração das três componentes de cores primarias. A imagem é então formada, semelhante ao método descrito na seção anterior, e depois projetada com o auxílio de lentes na tela de projeção.

Os projetores baseados em sistemas LCD também baseiam-se no processo descrito anteriormente, com a diferença de sempre possuírem matriz ativa, a qual permite que uma determinada linha e coluna da tela de cristal líquido seja endereçada e receba uma carga capaz de ser mantida até o próximo ciclo de atualização da tela. Com esta carga é possível controlar a intensidade da luz que irá passar por determinado pixel. Ao fazer isso em pequenos incremento, é possível criar uma escala de cinza.

Já os projetores DLP's utilizam semicondutores óticos para a manipulação digital da luz, dividida em três fases: semicondutor, onde o processo de acender um pixel ocorre pela movimentação de micro-espelhos posicionados em uma malha retangular; imagem cinza, obtida pela freqüência de movimentação dos micro-espelhos ao serem atingidos pela luz, onde quanto maior a freqüência, mais claro será o pixel, dentre 1024 níveis; e adição de cor, obtida pela utilização de máscaras, semelhante aos LCD's. A combinação da freqüência de cada micro-espelho com as componentes da máscara pode produzir 16.7 milhões de cores diferentes.

1.5.1.2.3. Workbenchs e CAVEs

Com o emprego de projetores, múltiplas telas de projeção e espelhos, é possível criar as mais variadas configurações de ambientes de visualização para RV, como as *Responsive Workbenchs* e CAVE's, de maneira que, a projeção de imagens

sobre tais espelho, permite o direcionamento dos raios para superfícies específicas, como telas translúcidas (Gattass *et al*, 2004).

A *Workbench* é um espaço interativo e tridimensional, onde imagens estereoscópicas são projetadas em um topo de mesa horizontal, utilizada como superfície de visualização. A projeção ocorre por um sistema de projetores e espelhos, onde os raios projetados atravessam a superfície translúcida da mesa e são visualizados por meio de óculos 3-D estéro-ativos (óculos oburadores). O movimento da cabeça do usuário é acompanhado utilizando um sistema de rastreamento com seis graus de liberdade (6DOF). Desta maneira, o usuário pode ver o ambiente virtual através do ponto de vista correto, entretanto restrito a apenas uma pessoa. A Figura 12 apresenta um exemplo de *Responsive Workbench*.

Já a CAVE trata-se de uma sala cujas paredes, teto e chão são superfícies de projeção, ou seja, sobre cada superfície semitransparente da sala existe um sistema de câmeras e espelhos responsáveis por transmitir determinada porção da cena virtual. A fusão das partes de cada câmera forma a cena completa e provê ao usuário a sensação de imersão no ambiente. Este dispositivo requer que as projeções estejam sincronizadas e as bordas das imagens sejam combinadas, para que uma junção seja imperceptível. A geração de uma perspectiva do usuário em uma CAVE não é um problema simples e deve ser calculada com base na posição do usuário, a rotação sua cabeça e a tela de projeção. A Figura 13 ilustra o funcionamento de uma CAVE.

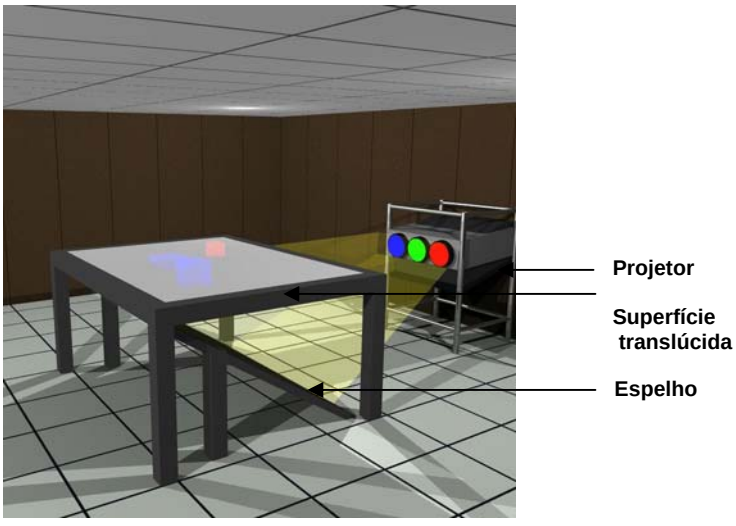


Figura 12 - Representação de uma Responsive Workbench, onde as imagens são projetadas sobre um espelho abaixo da mesa e incididas em sua superfície translúcida (Scgl, 2006)

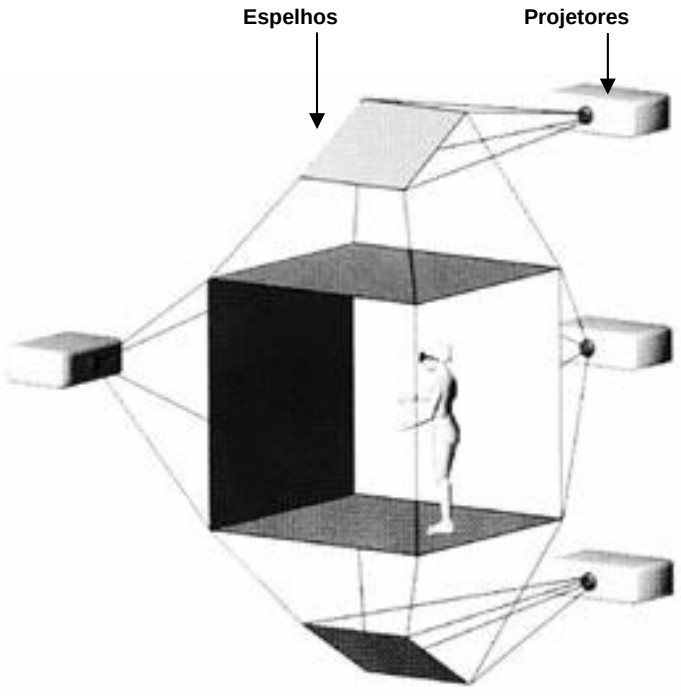


Figura 13 - Representação do funcionamento de uma CAVE, onde o usuário sente-se imerso através das diversas projeções a sua volta, cada qual retratando uma parte da cena
(Buxton e Fitzmaurice, 1998)

1.5.2. Dispositivos de entrada de dados

O usuário de um sistema de RV pode sentir-se dentro do ambiente sintético por intermédio dos dispositivos de saída de dados. Já os dispositivos de entrada, por outro lado, permitem o envolvimento do usuário e sua interação com tal ambiente. Sem um dispositivo de entrada de dados adequado o usuário participa da experiência de RV de forma passiva.

Pimentel (1995), divide os dispositivos de entrada em duas categorias: dispositivos de interação e dispositivos de trajetória. Os dispositivos de interação permitem ao usuário a movimentação e manipulação de objetos no mundo virtual (mouse, teclado, joysticks). Os dispositivos de trajetória, por sua vez, monitoram partes do corpo do usuário, detectando seus movimentos e criando a sensação de presença no mundo virtual (dispositivos de rastreamento).

1.5.2.1. Dispositivos de convencionais de entrada de dados

Os dispositivos de convencionais de entrada são periféricos de usabilidade comum, destinados a usuários comuns e sistemas de propósito geral. São eles: teclado, mouse, *trackballs*, digitalizadores de mesa (*scanners*), canetas digitais e microfones. No âmbito de sistemas de Realidade Virtual, os dispositivos não-convencionais de entrada de dados podem contribuir para o desenvolvimento de ambientes interativos e semi-imersivos, porém, não provêem o realismo e o envolvimento tal qual fornecidos por sistemas virtuais dotados de dispositivos não-convencionais, descritos na próxima sessão e utilizados no presente trabalho.

1.5.2.2. Dispositivos não-convencionais de entrada de dados: Rastreadores e Luvas Digitais

Os dispositivos de rastreamento tratam-se de periféricos não-convencionais de entrada de dados destinados a sistemas de RV. Graças a estes dispositivos, o usuário pode interagir com os objetos da cena virtual através de comportamentos interativos de manipulação direta (*behaviors*), contextualizando-se com a idéia original básica de um sistema de RV, onde se busca a interação natural do usuário com o ambiente virtual (VResources, 2007). O modo como os participantes interagem com o sistema de RV influencia enormemente suas experiências, afetando a facilidade de uso do sistema, a sensação de imersão do usuário e a variedade de ações que podem que o usuário pode tomar dentro do ambiente de RV (Gattass *et al*, 2004).

Já as luvas digitais comportam-se como dispositivos de entrada, devido principalmente a uma característica em comum: seus dispositivos de rastreamento. Assim, as luvas digitais funcionam como suporte aos vários sensores, mecânicos, óticos, acústicos, inerciais e magnéticos, utilizados para capturar dados físicos como a posição e a orientação espacial de um objeto, com complexidade variável.

1.5.2.2.1. Rastreamento por campo magnético

Este modo de detecção permite o rastreamento no espaço através de 6 graus de liberdade (6DOF). O princípio consiste em emitir três impulsos de rádio utilizando

antenas e bobinas, orientadas segundo os eixos de coordenadas cartesianas, as quais, sob corrente elétrica, geram um campo magnético. O usuário tem a mesma configuração de antenas como o emissor, para que possa captar cada onda de radio e assim determinar a intensidade do sinal sob cada eixo. A forca global das três antenas do receptor dará a distância relativa da fonte emissora. A presença de objetos metálicos na área provoca interferências na transmissão, configurando-se um inconveniente ao método. Além disso, o sistema é restrito a pequenos espaços, devido ao alcance do campo magnético de no máximo 3 metros, sem canal de comunicação desobstruído.

Os movimentos são então interpretados por *software*, que podem acompanhar o dispositivo ou ser programado manualmente. Alguns modelos podem ainda fornecer *feedback*, provendo a sensação de tato e categorizando as luvas de dados também como um dispositivo de saída.

Estes dispositivos são bastante precisos, cerca de 1 a 2 mm para posição e 0.1º para orientação, sendo que sua velocidade de captura varia de 100 a 200 leituras por segundo.A Figura 14 apresenta uma luva digital *5DTGlove* com sensores magnéticos.

1.5.2.2.2. Rastreamento por correlação óptica

Este método utiliza-se de análise estereoscópica, correlacionando *pixels* comuns a duas imagens, vistas por 2 câmeras *offset*. Como na transmissão magnética, esta técnica requer uma desobstrução do canal de comunicação, para que as câmeras possam ver os pontos a serem triangularizados (correlação de pixels entre as imagens) em posições tridimensionais, representados por diversos LED´s posicionados no dispositivo. Entretanto, este dispositivo de rastreamento é livre de interferências.

Sua velocidade depende muito do sensor empregado, limitado a amostragem no caso de uma câmera padrão NTSC, a qual consegue capturar imagens a 30 quadros por segundo. Já sua precisão, usualmente suficiente, depende de técnicas de calibração das câmeras e extração de informação da imagem, em geral empregadas sob algoritmos de visão computacional. A Figura 15 apresenta uma *Impulse Glove* com emissores para correlação óptica.

1.5.2.2.3. Rastreamento Mecânico

Este dispositivo baseia-se no princípio de medição de ângulos e distancias entre juntas, onde dada uma posição conhecida, todas as outras podem ser determinadas pela relação entre as juntas. Os rastreadores podem estar presos ao chão ou mesmo anexos ao corpo do usuário, na forma de exoesqueleto. Rotações e distâncias podem ser medidas por engrenagens, potenciômetros ou sensores de dobra. Suas vantagens encontram-se na facilidade de implementação de *feedback* de força, aplicando uma força contrária ao movimento do usuário. Possui alta precisão (0.1º para orientação) e baixo tempo de resposta (200ms). A Figura 16 apresenta uma luva com sensores mecânicos e suas engrenagens.

Figura 14 – 5DTGlove com sensores magnéticos, onde o receptor detecta a intensidade do sinal sob os três eixos, caracterizando o movimento (Dayeang, 2007)

Figura 15 – *Impulse Glove* com rastreamento óptico, que determina a posição por correlação de posições sob diversos emissores LED´s (PhaseSpace, 2008)

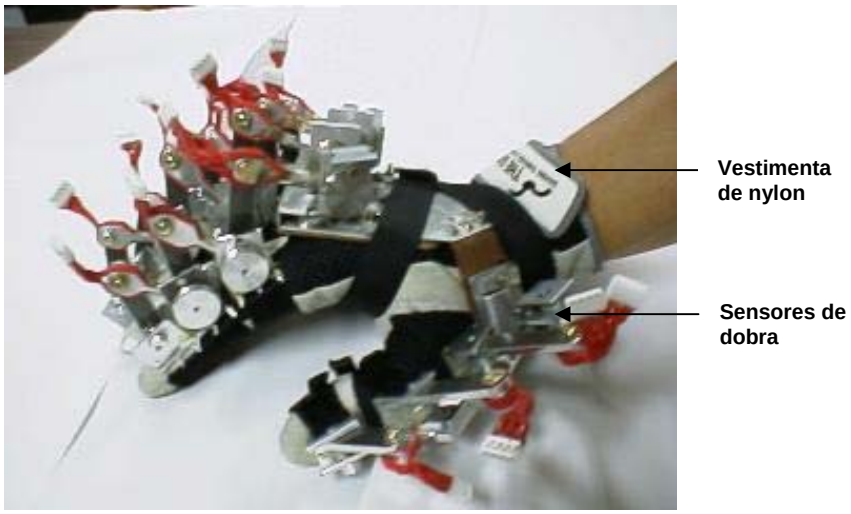


Figura 16 - Representação da luva digital com sensores mecânicos, os quais mapeiam os movimentos na cena através de deslocamentos físicos do exoesqueleto (Marcus, 1991)

1.5.2.2.4. Rastreamento Acústico

Utilizam ondas sonoras ultra-sônicas para medir distância, baseando-se em tempo de vôo e coerência de fase, tendo como objetivo transformar tempo de resposta em distância. Neste método de rastreamento, utiliza-se apenas um par emissor/receptor, o qual fornece a distancia de um objeto em relação a um ponto fixo e três pares fornecem a posição exata do periférico.

Quanto à precisão, o rastreador acústico provê um atraso devido à espera do sinal, dependente da baixa velocidade da velocidade do som. Paralelamente, o desempenho do dispositivo é degradado em ambiente ruidoso e sob caminho obstruído entre os autofalantes e microfones. Devido a estas restrições de interferência, a distancia entre receptor e transmissor não deve ultrapassar 15 metros. A Figura 17 demonstra uma luva digital com rastreamento acústico *Logitech Tracker*.

1.5.2.2.5. Rastreamento Inercial

Os dispositivos de rastreamento inerciais têm seu princípio de funcionamento baseados na segunda lei de Newton onde, $F = ma$ e $M = I\alpha$, sendo assim, o sistema deve integrar a leitura para obter a velocidade e a posição (Gattass *et al*, 2004). Possui três variantes de sensores: os magnetômetros passivos, que medem o campo

magnético do ambiente e fornece medidas angulares, os girômetros, que fornecem apenas medidas angulares e os acelerômetros, que fornecem medidas lineares.

Este método de rastreamento apresenta grande precisão, sendo capaz de alcançar uma resolução angular de até 0.2º em alguns casos. Também é livre de interferências, pois o sistema é autocontido, não havendo a necessidade de um ponto externo para a obtenção de dados, e livre de restrições físicas, limitado apenas pela conexão entre o dispositivo e o computador. Um exemplo de periférico não-convencional que utiliza esse tipo de sensor é a *CyberGlove* (Immersion, 2007), representada na Figura 18.

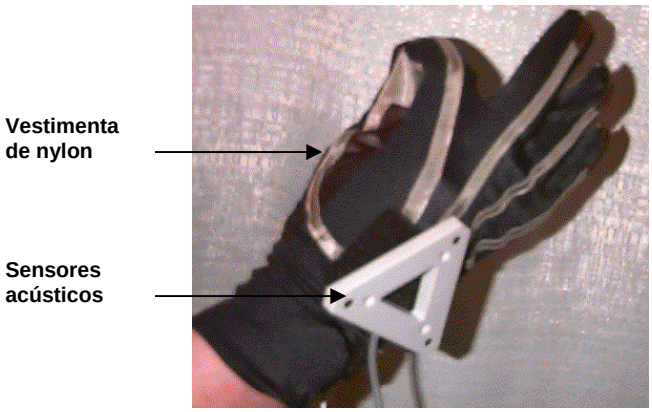


Figura 17 - Luva digital com rastreamento acústico, onde a mesma pode ser localizada no espaço através do tempo de resposta da onda sonora do três receptores, relativos aos três eixos tridimensionais (VRDepot, 2007)

Figura 18 - Representação de uma CyberGlove com sensores inerciais, os quais registram os deslocamentos a partir de uma posição de origem definida pelo sistema (Immersion, 2007).

A seguir, no Capítulo II são apresentados os principais aspectos que envolvem a Tomografia Computadorizada e a Ciência de Solos, contextualizando a área de aplicação do presente trabalho e a origem dos modelos tridimensionais utilizados.

Capítulo II
Tomografia Computadorizada e sua Aplicação na Ciência de Solos

2.1. Histórico da Tomografia Computadorizada

Em 1917, o matemático austríaco Radon foi o primeiro a apresentar uma solução matemática das equações de reconstrução de corpos a partir de projeções, determinando a função de distribuição de densidade da região estudada através de suas projeções, no campo da gravitação (Radon, 1917).

O embrião da tomografia computadorizada (TC) de raios-*x* pode ser encontrado nos trabalhos de Takahashi, onde planos indesejáveis foram eliminados no processo de aquisição, colocando a fonte de raios-*x* e o filme em um mesmo plano (Takahashi, 1957). Posteriormente em 1961, Oldendorf desenvolveu um primeiro aparato rústico para obter imagens através da transmissão de raios γ, promovendo a evolução dos instrumentos para a TC (Oldendorf, 1961).

O desejo de separar estruturas superpostas também levou ao desenvolvimento de uma variedade de técnicas tomográficas analógicas, especialmente a tomografia axial (secções transversais), porém geravam maus resultados. Pesquisadores

reconheceram, então, que um computador seria necessário para realizar o refinamento das imagens.

Nos anos 70, Hounsfield e sua equipe da *EMI Corporation* desenvolveram o primeiro tomógrafo computadorizado, dedicado às aplicações médicas, comercialmente viável, baseado em uma fonte de *Americium (Am)* e um detector de *NaI* (TI), onde a fonte e o detector foram posicionados em sentidos opostos na mesma direção sobre uma mesa motorizada que possibilitava sua rotação e translação (Hounsfield, 1973). Este tomógrafo permitiu pela primeira vez a visualização de estruturas internas do corpo através de secções transversais, e ainda, gerar dados substanciais para reconstruir até mesmo objetos não simétricos, trabalho pelo qual os pesquisadores receberem o prêmio Nobel de Medicina em 1979. Hounsfield dividiu o premio com Allan Cormack, professor de Física da Universidade de Carpe Town, que, paralelamente, desenvolveu uma série de algoritmos de reconstrução de imagens, baseados nos métodos matemáticos previstos por Radon (Cormack, 1973).

Com base em experimentos e observações, Cormack formulou uma matriz de coeficientes para cortes seccionais, que poderia ser obtida pela medida da transmissão de raios-X em vários ângulos através de um corpo (Cormack, 1973).

Aplicações da tomografia computadorizada destinada à agricultura foram discutidas inicialmente em 1981 por físicos e engenheiros eletrônicos da Fundação Educacional de Barretos (FEB), Universidade de Campinas (Unicamp) e Universidade de São Paulo (USP), de onde se originou um projeto multidisciplinar para o desenvolvimento de métodos e equipamentos.

O uso de TC em física do solo também foi discutido por Petrovic (Petrovic,1982), Hainswoth e Aymores (Hainsworth, 1983) e Crestana (Crestana,1986). Petrovic demonstrou a possibilidade de usar um tomógrafo computadorizado de raios X para medir a densidade de volumes de solos, enquanto que Crestana demonstrou que a TC pode solucionar problemas ligados ao estudo da umidade dos solos.

Em 1984, foi criado o Centro Nacional de Pesquisa e Desenvolvimento de Instrumentação Agropecuária (CNPDIA), da Empresa Brasileira de Pesquisa Agropecuária (EMBRAPA), instalada em São Carlos-SP. Assim, tornou-se possível o avanço do desenvolvimento de técnicas de tomografia de raios-x e γ no estudo de solos.

Em 1985 foi criado no Brasil um tomógrafo de raios X e raios γ automatizado e de baixo custo, projetado e desenvolvido inteiramente no Brasil pela mesma equipe de pesquisadores com o objetivo de aplicações multidisciplinares, especialmente na agricultura.

Em 1987, Cruvinel e colaboradores, desenvolveram na Embrapa Instrumentação Agropecuária, um minitomógrafo, baseado em fontes de raios-*x* e raios γ, destinado ao estudo de amostras de solo e plantas. O minitomógrafo também possibilitou experiências com diversas fontes de radiação em diversas energias (Cruvinel, 1987) (Cruvinel, 1990) (Cruvinel, 1996). O desenvolvimento deste minitomógrafo tem influenciado vários projetos com o propósito de melhoramento da visualização e manipulação das imagens adquiridas e dos algoritmos de reconstrução, bem como o aperfeiçoamento do *hardware* do minitomógrafo (Venturini, 1995) (Minatel, 1997) (Granato, 1998) (Cruvinel, 2000) (Pereira, 2001) (Pedrotti *et al*, 2003), (Cruvinel, 2006), (Pereira, 2007), (Pereira *et al*, 2007), (Laia, 2007), (Laia *et al*, 2007) e (Laia *et al*, 2008). A parceria com a Universidade Federal de São Carlos tem contribuído de forma importante para o desenvolvimento em nível computacional do minitomógrafo.

Para fornecer suporte adequado aos diversos sistemas de aquisição, em especial a Tomografia Computadorizada (TC), surgiram diversos algoritmos de reconstrução sob complexas técnicas computacionais que são capazes de reconstruir uma imagem a partir de projeções tomográficas devidamente filtradas, de forma a amenizar ruídos provenientes de várias fontes, como do próprio tomógrafo (Minatel, 1997) (Pereira, 2001) (Pereira, 2007).

2.2. Procedimentos de Tomografia Computadorizada

Em radiologia, uma Tomografia Computadorizada (TC) é uma imagem resultante do tratamento informatizado dos dados obtidos numa série de projeções angulares de raios-*x*, representando uma secção transversal, ou fatia, do corpo ou objeto em estudo. A TC, tal como a radiologia convencional, baseia-se na atenuação dos feixes de raios-*x*, parcialmente absorvidos pelos diversos materiais. Materiais

como plástico e água são facilmente atravessados pelos raios-*x*, ao contrário dos metais (Kak e Slaney, 1988).

O objeto tomografado é então atingido pela radiação e os feixes saem mais fracos do que entraram. Assume-se que os mesmos são tão finos que podem ser tratados matematicamente como linhas retas (unidimensionais sem área).

Há dois fatores que reduzem a intensidade dos feixes: a densidade do material e o comprimento da intersecção do feixe com o material. Supõe-se que cada porção do objeto tenha uma determinada densidade, no sentido de ter maior ou menor capacidade de atenuar os feixes. Essa capacidade de atenuar os feixes que é medida, com base na intensidade emitida pela fonte de raios-X e na intensidade captada por um detector na outra extremidade da reta de propagação. Assim, pode-se determinar o peso da atenuação do objeto, dado essencial para o processo de reconstrução, o qual possibilita deter um mapa de coeficientes lineares da secção transversal do objeto.

Este mapa de coeficientes é representado através de *pixels*, cujos valores são chamados números CT (*Computer Tomography*). Esses números são normalizados em função do coeficiente de atenuação da água (zero). Em outras palavras, definem-se os números CT por:

$$NúmeroCT = \frac{\mu - \mu_{H_2O}}{\mu_{H_2O}} \times 1000 \qquad (2.1)$$

onde μ é o coeficiente de atenuação do corpo analisado. Com esse numero é possível a obtenção de um mapa de coeficientes de atenuação, o que permite uma analise mais detalhada do corpo em estudo.

Basicamente, uma TC indica a quantidade de radiação absorvida, com o valor do sinal expresso em unidades Hounsfield, por cada porção da secção analisada. Tal valor é traduzido em uma escala de cinza e produz uma imagem. Como a capacidade de absorção de raios-*x* de um material está intimamente relacionada com sua densidade, zonas com diferentes densidades terão diferentes intensidades monocromáticas. A principal vantagem da TC é permitir o estudo de secções transversais de matérias sem intrusão, uma melhoria sem paralelo em relação às

outras técnicas de análise do solo, que, em geral, são invasivas e podem destruir características importantes a serem observadas.

Imagens de alta qualidade, provenientes de processos de TC previnem que materiais encontrados no solo não sejam interpretados de maneira errônea. A busca dessa qualidade é alcançada quando a imagem processada corresponde ao material encontrado por uma busca manual intrusiva.

2.3. Algoritmos de Reconstrução

O objetivo do processo de reconstrução de imagens tomográficas é, a partir de um objeto tridimensional *f(x, y, z)*, obter uma fatia representada por *f(x, y, 0),* de forma não destrutiva e não invasiva, a partir de suas projeções tomográficas.

Os dados necessários para a reconstrução de *f(x, y, 0)* consistem em um conjunto de integrais ao longo dos raios que atravessam o objeto. Assim, a linha tracejada, conforme demonstra a Figura 19, parte da fonte para o detector, atravessa o objeto e, à medida que eles se propagam através dos eixos L e L', formam com eixo x um ângulo θ. Obtêm-se então a projeção no ângulo θ, representada por $P_\theta(t)$.

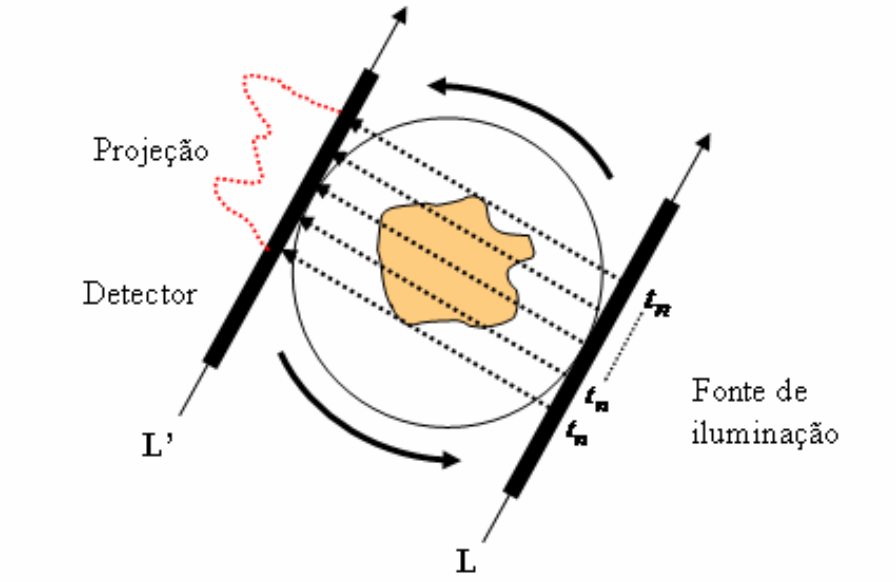


Figura 19 - Ilustração da tomografia de transmissão

A posição do detector é representada pela variável *t*. Com isso a função de integração do objeto ao longo do raio é uma integral de linha. Cada integral de linha

dos feixes paralelos à linha entre a fonte e o detector forma uma projeção paralela. Com as projeções com θ variando no intervalo $0 \leq \theta < \pi$, teremos a Transformada de Radon do objeto e, consequentemente, através da Transformada Inversa de Radon, é possível reconstruir o modelo tomografado. A Figura 20 demonstra a incidência de feixes paralelos sobre o objeto.

Os algoritmos de reconstrução conhecidos têm como entrada os coeficientes de atenuação obtidos pela projeção da radiação sobre o objeto e produzem como saída uma matriz de tons de cinza da imagem estimada, baseando-se nos dados disponíveis. A estimativa da imagem varia de método para método. Contudo a qualidade dos resultados depende de como os dados foram coletados e do objeto que está sendo inspecionado.

Tais algoritmos, do ponto de vista matemático e computacional, determinam como reconstruir um objeto $f(x, y)$, a partir dos dados de suas projeções em diversas direções. Entretanto, as dificuldades matemáticas e computacionais na reconstrução são aumentadas pela presença de ruídos diversos, causados por falhas no ajuste das direções das projeções e posicionamento dos feixes.

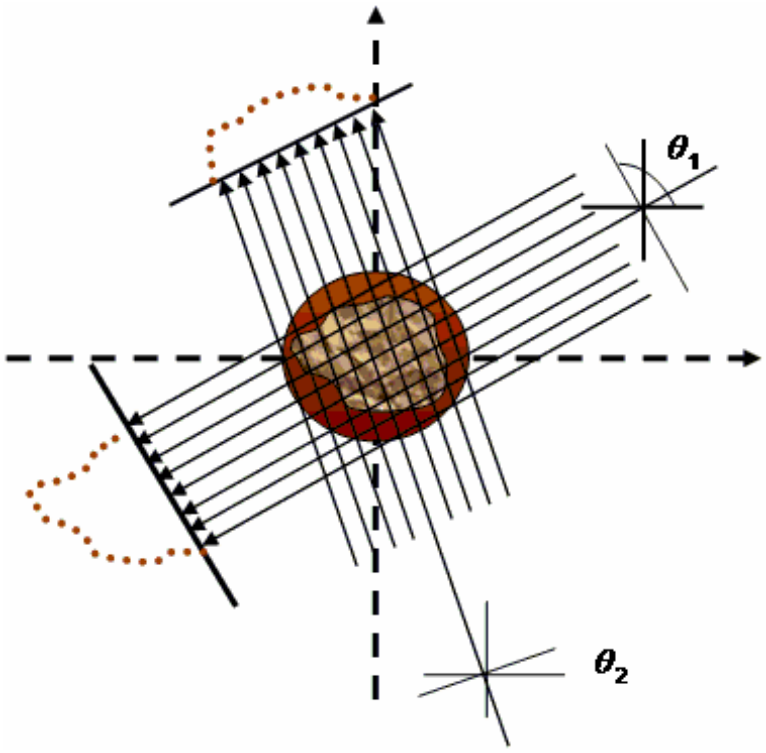


Figura 20 – Representação dos feixes paralelos incidindo sobre f(x, y).

Se uma fonte de raios x é usada, a integral de linha *AB*, é equivalente à integral de linha que possibilita obter os coeficientes de atenuação do objeto, permitindo ao algoritmo de reconstrução, atuar sobre os mesmos, como mostra a Figura 21, esquematizando o procedimento para a reconstrução a partir da Transformada de Radon.

O raio *AB* da Figura 20 no plano *z=0* pode ser expresso matematicamente por:

$$t_1 = x\cos\theta + y\,sen\,\theta \tag{2.2}$$

onde t_1 é a distancia perpendicular da origem até a linha. Com o uso desta equação do raio, a integral do raio é dada por:

$$P_\theta(t_1) = \int_{raioAB} f(x,y,0)dxdy = \int_{-y_m}^{y_m}\int_{-x_m}^{x_m} f(x,y,0)\delta(x\cos\theta + y\,sen\,\theta - t_1)dxdy \tag{2.3}$$

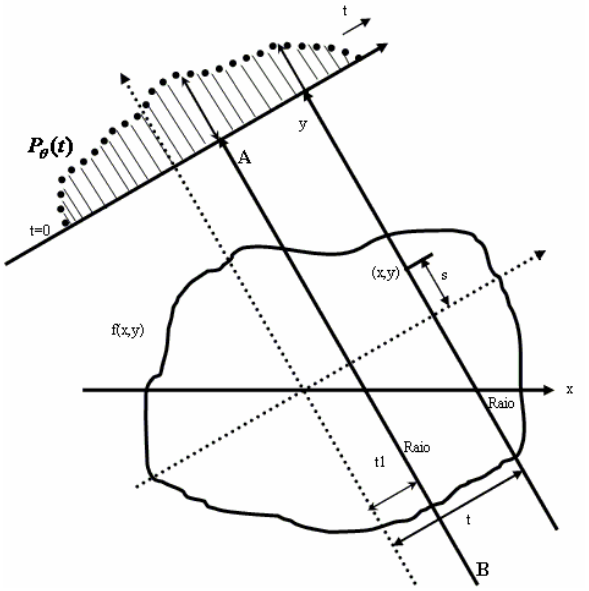


Figura 21 – Projeção paralela de f(x,y) para a Transformada de Radon

Rotacionando-se o eixo das coordenadas x sobre o eixo *z* por um ângulo θ, o eixo *t* e a projeção pode ser escrita por:

$$P_\theta(t_1) = \int_{-s_m}^{s_m} f(t_1, s, 0)ds \tag{2.4}$$

onde:

$$t_1 = x\cos\theta + y\,sen\,\theta \;\; e \;\; s = -x\,sen\,\theta + y\cos\theta \tag{2.5}$$

Com $P_{\theta_i}(t)$ sendo uma função de t representando a projeção paralela com ângulo θ_i. Para θ contínuo, a função $P_{\theta_i}(t)$ é a Transformada de Radon de $f(x, y, 0)$, com as projeções obtidas paralelamente à rotação no eixo x nomeadas por t.

No caso tridimensional, um raio pode ser descrito pela intersecção de dois planos, ou seja, uma vez que $t_1 = x\cos\theta + y\,sen\,\theta$, nas coordenadas (t, s', r) tem-se:

$$r_1 = -(-x\,sen\,\theta + y\cos\theta)sen\,\gamma + z\cos\gamma \tag{2.6}$$

Tal sistema de coordenadas é obtido por duas rotações dos eixos (x, y, z). A primeira rotação, análogo ao caso bidimensional, utilizando-se o ângulo θ com o eixo z, resultando no eixo (t, s, r). A segunda rotação ocorre sob o ângulo γ com o novo eixo t, resultando em (t, s', r). Em formato matricial, as equações podem ser representadas por:

$$\begin{bmatrix} t \\ s' \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & sen\,\gamma \\ 0 & -sen\,\gamma & \cos\gamma \end{bmatrix} = \begin{bmatrix} \cos\theta & sen & 0 \\ -sen\,\theta & \cos & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.7}$$

Sendo uma projeção paralela tridimensional representada por:

$$P_\theta(t_1, r_1) = \int_{-s_m}^{s_m} f(t_1, s', 0)ds' \tag{2.8}$$

A projeção resultante é agora bidimensional e obtida pela integração das linhas paralelas no eixo s'. Assim a integral de linha pode ser matematicamente expressa na forma apresentada.

2.3.1. Reconstrução Bidimensional

A função imagem descreve a distribuição dos coeficientes de atenuação na área da secção transversal em estudo. Após a varredura tomográfica, é montada a

matriz de níveis de cinza (matriz de contagem). A partir dessa matriz é possível reconstruir a imagem tomográfica utilizando um dos métodos de reconstrução que são descritos a seguir.

2.3.1.1. Método Algébrico - ART

ART é um método iterativo, no qual a reconstrução da imagem é estimada a partir de aproximações sucessivas, até que as projeções calculadas se aproximem de determinadas medidas dentro de uma determinada margem de erro. A metodologia utilizada para verificar a correção da imagem, calcula as integrais de linha e compara-as com tais medidas, utilizadas como referência para melhorar a estimativa da reconstrução (Ribeiro, 1994).

Este método, utilizado por Hounsfield no desenvolvimento do primeiro tomógrafo computadorizado, apresenta como vantagem a não necessidade do conhecimento do conjunto complexo dos dados das projeções. Este fato permite minimizar o tempo de varredura e torna possível reconstruir uma amostra que apresenta em seu interior um material opaco, inerente à radiação emitida. Como desvantagens, o método apresenta baixa precisão e maior tempo de processamento em relação aos métodos por transformadas (Brooks, 1976).

2.3.1.2. Teorema das Secções Bidimensionais de Fourier

O Teorema das Secções de Fourier para projeções tomográficas é base para a maioria dos algoritmos de reconstrução. Tal teorema diz que a Transformada de Fourier de uma projeção paralela de uma imagem $g(x, y)$ sob um ângulo θ é equivalente a uma fatia de uma transformada bidimensional de $g(x, y)$, representada por $G(x, y)$. Em outras palavras, a Transformada de Fourier de $P_{\theta_i}(t)$ fornece os valores de $G(\omega 1, \omega 2)$ sobre a linha BB' mostrada na Figura 22.

Neste teorema a Transformada de Fourier unidimensional da função $g(x, y)$ é matematicamente relacionada com sua Transformada de Fourier bidimensional.

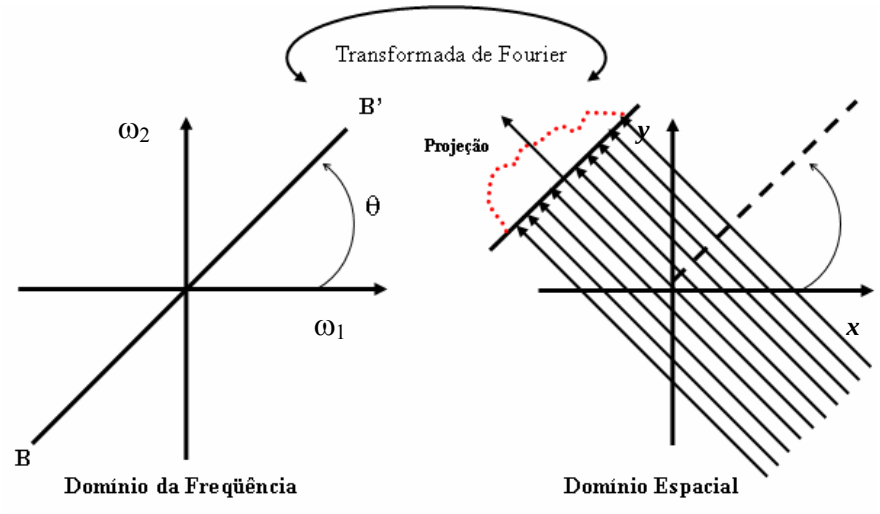


Figura 22 – Teorema das secções de Fourier

O teorema diz que se diferentes projeções são armazenadas em diferentes posições ao redor do objeto, a Transformada de Fourier pode obtida. Se todas as projeções forem armazenadas o objeto pode ser reconstruído diretamente pela inversão da Transformada de Fourier.

O Teorema das secções de Fourier para a secção tomográfica pode ser representado de forma que G(ω1, ω2) seja a Transformada de Fourier da imagem $g(x, y)$, podendo ser implementada por:

$$G(\omega_1, \omega_2) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)e^{-j2\pi(\omega_1 x+\omega_2 y)}dxdy \qquad (2.9)$$

e sua inversa por:

$$g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(\omega_1, \omega_2)e^{j2\pi(\omega_1 x+\omega_2 y)}d\omega_1 d\omega_2 \qquad (2.10)$$

Em seguida, consideram-se os valores de $G(\omega_1, \omega_2)$, com a linha ω_2=0 no plano (ω_1, ω_2). Da equação (2.9), $G(\omega_1, 0)$ denota-se como:

$$G(\omega_1, 0) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)e^{-j\omega_1 x}dxdy = \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} g(x,y)dy\right]e^{-j\omega_1 x}dx$$
$$= \int_{-\infty}^{\infty} P_\theta(t)e^{-j\omega_1 t}dt = S_\theta(\omega_1) \qquad (2.11)$$

A equação acima é resultante de $P_\theta(t) = \left(\int_{-\infty}^{\infty} g(x,y)dy \right)$ com θ =0 e t = x. Pode-se observar que $S_\theta(\omega)$ é a transformada de Fourier da projeção $P_\theta(t)$. O resultado $G(\omega_1, 0) = S_0(\omega)$ implica que a transformada de Fourier da projeção calculada perpendicular ao eixo y, $S_0(\omega_1)$, é igual a transformada de Fourier do objeto sobre o eixo ω_1 . Resultado este que pode ser expandido para obter um resultado similar para θ diferente de zero, rotacionando-se os eixos de coordenadas (x, y) por um ângulo θ para formar o eixo t e s, por intermédio da equação 2.12.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\theta & sen\,\theta \\ -sen\,\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} t \\ s \end{bmatrix} \qquad (2.12)$$

Transcrevendo para as coordenadas (x, y):

$$S_\theta(\omega) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x,y)dse^{-j(x\omega\cos\theta + y\omega sen\theta)}dxdy = G(\omega\cos\theta + \omega sen\theta) \qquad (2.13)$$

Por intermédio de coordenadas polares (ω, θ), pode-se obter $G(\omega, \theta)$:

$$G(\omega, \theta) = S_\theta(\omega) \qquad (2.14)$$

A Transformada de Fourier da projeção $P_\theta(t)$ corresponde aos valores, no domínio da freqüência, de $G(\omega_1, \omega_2)$ sobre a linha *BB'* com ângulo θ.

2.3.1.3. Algoritmo de Retroprojeção Filtrada

Um dos principais algoritmos de reconstrução, baseado em transformadas, é o algoritmo da retroprojeção filtrada, amplamente utilizados em aplicações que baseiam-se em fontes não difrativas. Vários fatores contribuem para a ampla divulgação desse algoritmo, dentre os quais a eficiência, precisão e baixo custo computacional, além de utilizar um hardware simples, o que se torna vantagem sobre outros algoritmos de reconstrução.

Contudo, a precisão do método é limitada e a qualidade da imagem reconstruída e freqüentemente degradada pela presença de diferenças substanciais nos coeficientes de atenuação de regiões externas à área de interesse. Com o propósito de se evitar tal problema, introduziu-se a varredura de seções transversais (*transverse section scanning*), mostrado na Figura 23 com as respectivas projeções geradas, onde apenas as regiões de interesse contribuem no resultado final. Neste tipo de varredura o detector recebe as projeções de uma estrutura em várias direções e em um único plano transversal (Herman, 1973). O princípio desta técnica é o mesmo que o de qualquer outra em tomografia: o coeficiente de atenuação (ou densidade) é estimado pela soma do total das densidades, isto é a soma de todos os raios que atravessam o ponto.

O algoritmo descrito nesta seção trata-se, na realidade, de uma derivação do teorema das secções de Fourier e sua implementação diferente do que o teorema básico sugere. Para iniciar a derivação faz-se necessário o uso de coordenadas polares, (ω_1, θ) no plano (ω_1, ,ω_2), como descrito a seguir:

$$g(x,y)=\int_0^{2\pi}\int_0^{\infty} G(\omega,\theta)e^{j\omega(x\cos\theta+ysen\theta)}\omega d\omega d\theta=\int_0^{\pi}\int_0^{\infty} G(\omega,\theta)e^{j\omega(x\cos\theta+ysen\theta)}\omega d\omega d\theta$$
$$+\int_0^{\pi}\int_0^{\infty} G(\omega,\theta+180°)\cdot e^{j\omega(x\cos(\theta+180°)+ysen(\theta+180°))}\omega d\omega d\theta \qquad (2.15)$$

Usando $G(\omega,\theta+180°)=G(-\omega,\theta)$ e $t = x$ cos θ + y sen θ , pode-se escrever $g(x,y)$ com a ajuda do teorema de secções de Fourier e a expressão para t em termos de x e y, como escrito na equação (2.10) ou seja:

$$g(x,y)=\int_0^{\pi}\left[\int_{-\infty}^{\infty} G(\omega,\theta)|\omega|e^{j\omega t}d\omega\right]d\theta=\int_0^{\pi}\left[\int_{-\infty}^{\infty} S(\omega)|\omega|e^{j\omega t}d\omega\right]d\theta \qquad (2.16)$$

Para construir a equação (2.16) sob a forma de retroprojeção filtrada, é necessário separar a equação em duas partes distintas. A primeira trata-se da filtragem dos dados de projeção para cada ângulo θ, como se segue:

$$Q_\theta(t)=\int_{-\infty}^{\infty} S_\theta(\omega)|\omega|e^{j\omega t}d\omega \qquad (2.17)$$

Depois as projeções filtradas são retroprojetadas sob a forma:

$$g(x,y) = \int_0^{\pi} Q_\theta(x\cos\theta + y\,sen\theta)\,d\theta \qquad (2.18)$$

Para que a filtragem descrita na equação (2.17) tenha o efeito desejado, faz-se necessário que as projeções sejam limitadas em banda com a máxima freqüência de W radianos/segundo. Assim, dada a propriedade do teorema de *Nyquist*, o valor de W será utilizado como freqüência deforma que ω_s como:

$$2W \le \omega_s = 2\pi / \tau \qquad (2.19)$$

Se os dados de projeção descritos na equação (2.19) são amostrados com um intervalo de amostragem t, os mesmos não sofreriam por erros de *aliasing*. Por meio da limitação por banda, a equação acima é expressa como o uso de funções de transferências de filtros $H(\omega)$, como se segue:

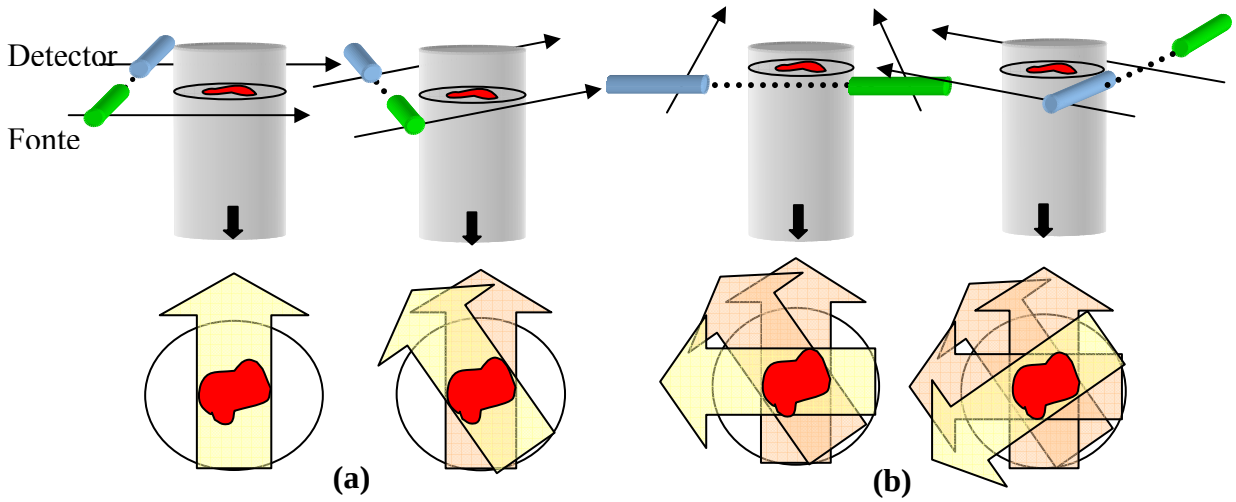


Figura 23 - Procedimento de varredura de um plano de um objeto tridimensional, no método da retroprojeção filtrada – (a) Varredura do plano com θ= 0º ; (b) Varredura do plano com θ= 45º; (c) Varredura com θ= 90º; (d) Varredura com θ = 35º (Pereira, 2001)

$$H(\omega) = \begin{cases} |\omega|, & |\omega| < W \\ 0, & outros \end{cases} \qquad (2.20)$$

A substituição de $H(\omega)$ em (2.17) fornece

$$Q_\theta(t) = \int_{-\infty}^{\infty} S_\theta(\omega) H(\omega) e^{j\omega t} d\omega \qquad (2.21)$$

A resposta impulsiva *h(t)* é obtida pela Transformada Inversa de Fourier, $H(\omega)$:

$$h(t) = \int_{-\infty}^{\infty} H(\omega) e^{j\omega t} d\omega = \frac{1}{2\tau^2}\left[\frac{sen(2\pi t/2\tau)}{2\pi t/2\tau}\right] - \frac{1}{4\tau^2}\left[\frac{sen(\pi t/2\tau)}{\pi t/2\tau}\right]^2 \qquad (2.22)$$

A versão amostrada do impulso resposta, o qual se conhece por filtragem digital de dados amostrados de projeções, é dada por:

$$h(n\tau) = \begin{cases} 1/4\tau^2, & n = par \\ 0, & n = par \\ -1/n^2\pi^2\tau^2, & n = ímpar \end{cases} \qquad (2.23)$$

Pelo teorema da convolução, a equação (2.21) pode ser escrita como:

$$Q_\theta(t) = \int_{-\infty}^{\infty} P_\theta(\zeta) h(t-\zeta) d\zeta \qquad (2.24)$$

a qual pode ser discretizada na forma:

$$Q_\theta(n\tau) = \tau \sum_{k=-\infty}^{\infty} h(n\tau - k\tau) P_\theta(k\tau) \qquad (2.25)$$

Em aplicações reais, a implementação cada projeção é finita, fornecendo uma versão truncada da equação (2.25):

$$Q_\theta(n\tau) = \tau \sum_{k=0}^{N-1} h(n\tau - k\tau) P_\theta(k\tau) \quad n = 0,1,2...N-1 \qquad (2.26)$$

onde $P_\theta(k\tau)$=0 para *k<0* e *k>N-1*. Tal tipo de filtragem de dados de projeção pode ser efetuada pela convolução discreta no domínio do tempo ou como uma multiplicação no domínio da freqüência, como a seguir:

$$Q_\theta(n\tau) = \tau \times IFFT\left\{FFT\left[\ P_\theta(n\tau)\, com\, ZP\ \right] \times FFT\left[\ h(n\tau)\, com\, ZP\ \right]\ \right\} \quad (2.27)$$

onde FFT e IFFT são as Transformadas Rápidas de Fourier e sua Inversa, respectivamente, e ZP é o preenchimento por zero, o que contribui para a suavização do efeito *aliasing*. O passo seguinte da reconstrução é a retroprojeção das projeções filtradas que tem sua aproximação discretizada por:

$$\hat{\hat{g}}(x,y) = \frac{\pi}{K}\sum_{i=1}^{K} Q_\theta(x\cos\theta_i + y\, sen\,\theta_i) \quad (2.28)$$

onde K ângulos θ_i são os valores discretos de θ para cada $P_\theta(t)$ conhecido.

Cada ponto *(x,y)* no processo de retroprojeção que estejam sobre plano de reconstrução na linha *t* possui o valor $Q_{\theta i}(t)$. A adição de cada valor de $Q_{\theta i}(t)$ no ponto *(x,y)* com *i* variando de 1 até *K* constitui o valor do ponto em uma escala de π/K. O valor de $Q_{\theta_i}(t)$ é uma constante sobre a linha *LM* e cada ponto no plano de reconstrução que tem o valor de $Q_{\theta_i}(t)$ é a ele adicionado. Em outras palavras, a imagem da reconstrução é gerada pela soma de todos os valores *t* de $Q_{\theta_i}(t)$, para cada valor θ_i , projetados e multiplicados por π/K. Quando o valor de *t* calculado não corresponde a algum dos valores de *t* na função discretizada $Q_{\theta_i}(t)$, há a necessidade de interpolação. Uma interpolação linear é, em muitos casos, suficiente.

2.3.2. Reconstrução Tridimensional

O grande desafio da reconstrução tridimensional de imagens é reconstituir um corpo a partir simples projeções bidimensionais, estimando dados internos de *voxels*[2] faltantes. Entretanto, os problemas na reconstrução de corpos por meio de projeções

[2] Voxel: Menor unidade representativa de volume em espaço tridimensional.

bidimensionais vêm sendo solucionados através de diferentes algoritmos e plataformas computacionais. Seus principais beneficiários vão desde medicina, passando pelo estudo de solo até a microscopia eletrônica.

A principal diferença entre a aquisição de dados tomográficos tridimensionais e a aquisição de dados bidimensionais é a redundância de dados e a variância espacial. A Figura 24 mostra o processo de aquisição tridimensional.

O vetor unitário $z_r(\beta,\vartheta)$ é perpendicular ao plano de projeção e descreve a orientação do plano. Considerando que $x_r \times z = 0$, as projeções em formação, oriundas do processo de aquisição, vêm de uma função tridimensional $f(x, y, z)$, para uma função 4D $p(x_r, y_r, \beta, \vartheta)$.

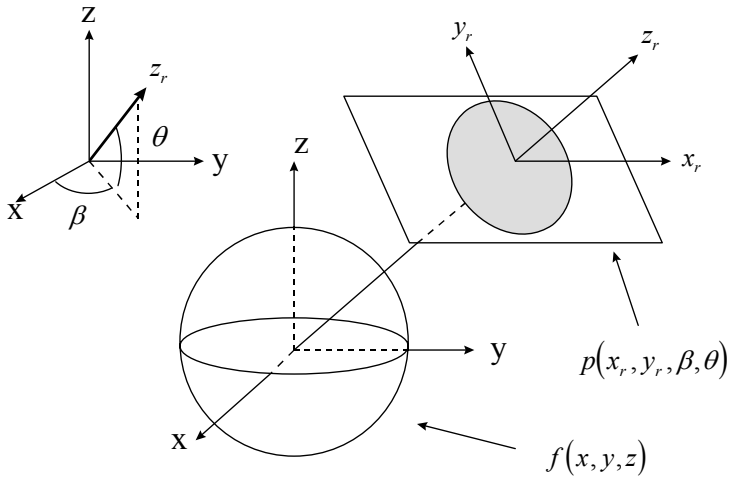


Figura 24 – Projeção bidimensional a partir de aquisição tomográfica tridimensional.

2.3.2.1. Teorema das Secções Tridimensionais de Fourier

Análogo ao teorema das secções bidimensionais de Fourier, a Transformada de Fourier bidimensional das projeções é equivalente aos valores do plano que passa através da origem da Transformada de Fourier tridimensional do objeto no seu ângulo apropriado como ilustra a Figura 25.

A Transformada de Fourier de projeções para $\vartheta = 0$ amostra completamente a Transformada de Fourier do objeto. Isso significa que a Transformada de Fourier de uma única projeção adicional para $\vartheta > 0$ acrescenta informações redundantes quando ela atravessa a Transforma de Fourier do objeto. Na prática, com a discretização de

um conjunto de projeções existem mais dados com $\vartheta > 0$ do que com $\vartheta = 0$, gerando um alto grau de redundância, efeito que pode ser útil ao processo de reconstrução, já que contribui para a redução de ruídos estatísticos.

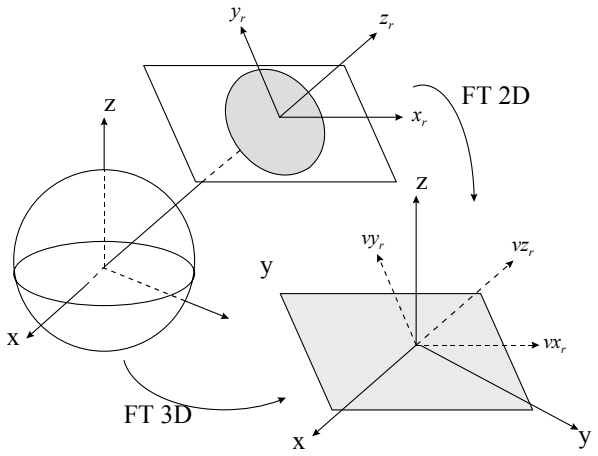


Figura 25 – Ilustração do Teorema das secções de tridimensionais de Fourier

2.3.2.2. Sobreposição de fatias interpoladas

Quando disponível apenas o método de aquisição bidimensional de fatias tomográficas, a reconstrução tridimensional pode ser feita também a partir da sobreposição das mesmas. Tal técnica consiste em montar os planos gerados pelas funções f(x ,y, z_i) para $i = 0, ..., n-1$, onde n é o número de planos reconstruídos.

Conseqüentemente, dados específicos das fatias bidimensionais podem ser interpolados de forma a reconstituir os espaços deixados entre estes planos sobrepostos, reconstruídos bidimensionalmente. O menor elemento formado pela reconstrução tridimensional é comumente chamado de *voxel*. A Figura 26 ilustra a sobreposição dos planos originais e os planos interpolados.

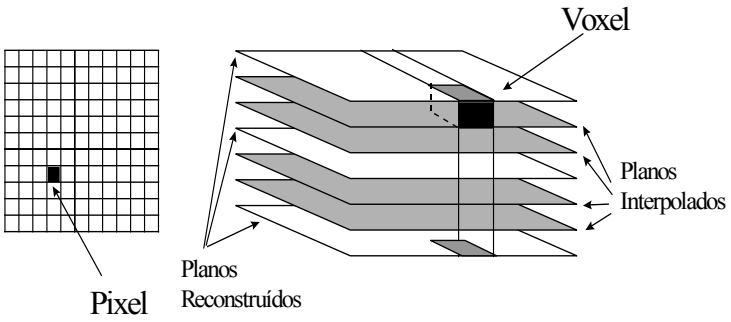


Figura 26 - Técnica de sobreposição e interpolação de planos na reconstrução tridimensional.

2.3.2.3. Interpolação por *B-Splines* (*B-Wavelets*)

Existe uma relação direta entre a funções wavelet ψ_H e a função *B-Spline* de primeira ordem representada por N_1. Esta relação é mostrada na Equação 2.29.

$$\psi_H(x) = N_1(2x) - N_1(2x-1) \tag{2.29}$$

e tem a relação com *B-Spline* de segunda ordem N_2 dada por:

$$\psi_H(x) = N_2(2x) \tag{2.30}$$

melhor dizendo, o valor de N_m pode ser dado pela Equação:

$$N_m(k) = \begin{cases} N_2(k) = \delta_{k,1} \quad k \in \mathrm{Z} \\ N_{n+1}(k) = \dfrac{k}{n} N_n(k) + \dfrac{n-k+1}{n} N_n(k-1) \quad k = 1,\ldots,n \end{cases} \tag{2.31}$$

Toma-se agora função *B-Spline* L_2 dada por $L_2(x) = N_2(x+1)$ onde L é definida pela Equação (2.32), na ordem m e com a seqüência de coeficientes c_k como:

$$L_m(x) = \sum_{k=-\infty}^{\infty} c_k^{(m)} N_m\left(x + \frac{m}{2} - k\right) \tag{2.32}$$

Pode-se definir uma wavelet básica como:

$$\psi_H(x) = L_2(2x-1) \tag{2.33}$$

Seguindo esse desenvolvimento, pode-se conseguir *Wavelets-Splines* de ordens superiores, ou melhor dizendo, pode-se gerar espaços de ordens superiores pela relação:

$$V_{j+1}^{(m)} = V_{2m}^{(m)} \oplus W_j^{(m)}, j \in \mathrm{Z}\,. \qquad (2.34)$$

Onde $\oplus$ simboliza uma soma ortogonal, V pertence a seqüência $\left\{V_j^m\right\}$ gerada pela análise de multiresolução gerada pela *B-Spline* de *m*-ésima ordem e W pertence a $\left\{W_j^m\right\}$ que é a seqüência de espaços wavelets ortogonais complementares.

Na literatura encontra-se o teorema: "Seja *m* um número inteiro positivo e definindo $\psi_{I,m}(x) = L_{2m}^{(m)}(2x-1)$ onde L_{2m} é a spline fundamental de $(2m)^{esima}$ ordem. Então $\psi_{I,m}$ gera os espaços complementares $W_j^m, j \in Z$ da seguinte forma:

$$W_j^m = fecho L^2(\Re)\left\langle 2^{j/2}\psi_{I,m}\left(2^j x - k\right) : k \in \mathrm{Z}\right\rangle, j \in \mathrm{Z} \qquad (2.35)$$

2.3.2.3.1. Algoritmo de interpolação por *B-wavelets*

A interpolação por B-wavelets determina os valores intermediários entre uma seqüência de pontos conhecida. Diferentemente da aproximação, a interpolação não apenas desloca a curva gerada sob a influência dos pontos conhecidos, como também faz com que essa curva passe por esses pontos. A Figura 27 ilustra essa diferença:

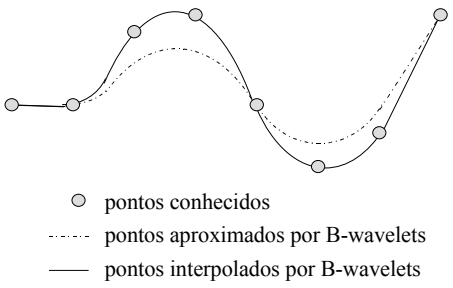


Figura 27 – Exemplo de interpolação e aproximação por B-Wavelets.

Para descrever a interpolação faz-se necessária uma discussão sobre o método de aproximação, seus problemas e suas soluções computacionais.

Seja f uma função de aproximação Spline de ordem m e com passo $\mu = 2^{-j0}\mathrm{Z}$ dada por:

$$f(\mu) = \sum_{i=0}^{N} a_i B(N\mu - i) \tag{2.36}$$

onde N é o número de pontos conhecidos.

De forma a otimizar o processo de cálculo, implementa-se a função $B(x)$, também chamada de função de "*blending*", da seguinte maneira:

$$B(x) = \begin{cases} \frac{1}{6}(2+x)^3 & -2 \le x \le -1 \\ \frac{1}{6}\left(4 - 6x^2 - 2x^3\right) & -1 \le x \le 0 \\ \frac{1}{6}\left(4 - 6x^2 + 2x^3\right) & 0 \le x \le 1 \\ \frac{1}{6}(2-x)^3 & 1 \le x \le 2 \\ 0 & 2 \le |x| \end{cases} \tag{2.37}$$

A implementação direta desta somatória fornece os valores intermediários da aproximação. No entanto, para os pontos inicial e final para m=4 , por exemplo, tem-se os valores:

$$f(0) = \frac{2}{3}a_0 + \frac{1}{6}a_1 \neq a_0 \tag{2.38}$$

e

$$f(1) = \frac{2}{3}a_N + \frac{1}{6}a_{N-1} \neq a_N \tag{2.39}$$

Para solucionar esse problema, adota-se o uso dos chamados "pontos fantasmas". Essa técnica consiste em considerar um ou mais valores antes do ponto inicial e depois do final.

São adicionados dois pontos fantasmas no início e dois no final da seqüência

conhecida. Esses valores são ajustados de forma dinâmica para que a curva a ser gerada pela função *B-wavelet* passasse pelos pontos inicial e final.

Esses pontos tém seus valores obedecendo a regra: $a_{-1} = 2a_0 - a_1$ e $a_{-2} = 2a_{-1} - a_0$ para os pontos iniciais e $a_{N+1} = 2a_N - a_{N-1}$ e $a_{N+2} = 2a_{N+1} - a_N$ para os pontos finais.

A função mostrada na Equação 2.36 fica da seguinte forma:

$$f(\mu) = \sum_{i=-2}^{N+2} a_i B(N\mu - i) \tag{2.40}$$

Para a implementação da interpolação, ao invés da simples aproximação, usa-se uma função muito próxima da função de aproximação. Essa função é dada por:

$$f(\mu) = \sum_{i=-2}^{N+2} A_i N_m(N\mu - i) \tag{2.41}$$

A idéia básica é usar a seqüência de pontos A_i no lugar de a_i. Essa seqüência é dada pela multiplicação das matrizes $A = M^{-1}a$, onde *A* é o vetor (seqüência) a ser encontrado, *v* é o conjunto de pontos conhecidos com os pontos fantasmas e *M* é uma matriz N+3 x N+3 dada por:

$$M = \begin{bmatrix} -\frac{N}{2} & 0 & \frac{N}{2} & 0 & 0 & \Lambda & 0 & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & 0 & \mathrm{K} & 0 & 0 \\ 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & \mathrm{K} & 0 & 0 \\ 0 & \mathrm{K} & \mathrm{K} & \mathrm{K} & \mathrm{K} & \mathrm{K} & 0 & 0 \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} \\ 0 & 0 & 0 & \mathrm{K} & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 \\ 0 & 0 & 0 & \mathrm{K} & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ 0 & 0 & 0 & \mathrm{K} & 0 & -\frac{N}{2} & 0 & \frac{N}{2} \end{bmatrix} \tag{2.42}$$

No Capítulo III é apresentado uma síntese da API *Java3D*, em sua mais recente distribuição, com o propósito de contextualização e justificativa da aplicação da mesma no desenvolvimento de ambientes virtuais interativos e imersivos, demonstrando a implementação de recursos computacionais gráficos incluídos no presente trabalho.

Capítulo III

Introdução à *API Java3D* em Contextualização ao Desenvolvimento de Sistemas de Realidade Virtual

Segundo Manssour (2003), a API Java 3D trata-se de um pacote com classes e métodos hierárquicos ideais para a construção de sistemas gráficos tridimensionais. Tal conjunto de classes possui construtores de alto nível que viabilizam a criação e manipulação de geometrias e importação de objetos, modelados sob diversas ferramentas e formatos, definidos sobre um universo virtual. A API Java 3D também possibilita a criação de ambientes sintéticos com uma grande flexibilidade e portabilidade, onde as cenas são representadas por grafos, seus atributos agrupados e suas transformações efetivadas. Desta maneira, a implementação de um programa Java 3D se resume na modelagem, ou importação de modelos, e na sua atribuição a grafos de cena, organizados em nodos hierárquicos. Os grafos de cena são responsáveis pela especificação do conteúdo do universo virtual e pela forma como este é visualizado.

Java3D foi desenvolvida pela *Sun Microsystems*, em conjunto com a *Apple*, *Intel* e *Silicon Graphics*, e teve seu código disponível a partir de 1998. Com o intuito de fornecer classes e métodos gráficos em uma plataforma independente, o Java 3D surgiu como uma alternativa de desenvolvimento semelhante ao *VRML* (*Virtual*

Reality Modeling Language), porém mais otimizada e baseada em tecnologias como o *OpenGL* e o *DirectX*.

3.1. Universos Virtuais

Para a criação e manipulação de geometrias, os programadores utilizam construtores de alto nível, pois os detalhes para a geração das imagens são abstraídos e instanciados automaticamente. Representações tridimensionais, juntamente com luzes, som e outros elementos integrados, compõem um universo virtual. Neste universo podem existir um ou mais grafos de cena, os quais cuidam da organização dos nodos em uma estrutura do tipo árvore (hierárquica).

3.2. Grafos de Cena

Instâncias de classes Java 3D que definem luz, som, orientação espacial, aparência, geometrias e localização, representam um Grafo de Cena. Tais instâncias são reconhecidas na estrutura pelos nodos (ou vértices) e os seus respectivos relacionamentos, identificados por arcos (ou arestas). As mesmas representam relacionamentos por referência, o qual simplesmente associa um objeto com o Grafo de Cena. Os nodos também são organizados hierarquicamente (pai-filho), onde um nodo de agrupamento pode ter filhos, os quais contêm suas definições e transformações, e apenas um pai. Um "nodo do tipo folha" não pode ter filhos. Em um Grafo de Cena habitual, conforme a Figura 27, os nodos do tipo grupo são identificados graficamente por elipses, e as folhas por triângulos. Assim, uma elipse é considerada a raiz, e os demais são acessados hierarquicamente (Selman, 2002) (Sun, 2007).

Por questão de padronização, cada grafo de cena deve conter um único Universo Virtual, (*VirtualUniverse)*. Tal objeto configura o universo a ser utilizado e possui pelo menos um objeto *Locale*, que demarca um ponto de referência no universo virtual e funciona como ponto base de todos os nodos de agrupamento (*BranchGroup*). A principal finalidade dos nodos de agrupamento é associar outros nodos através de algum atributo comum ou por meio de um conjunto de características.

Os nodos associados a um Grafo de Cena regem duas categorias básicas comportamentais: os nodos que descrevem o conteúdo do universo virtual (*content branch graphs* ou subgrafo de conteúdo), os quais tratam de aparência, geometrias, localizações, sons e iluminação, e os nodos que controlam os parâmetros de controle da visualização da cena (*view branch graphs* ou subgrafo de visualização). Tal divisão de categorias pode também ser observada pela Figura 28.

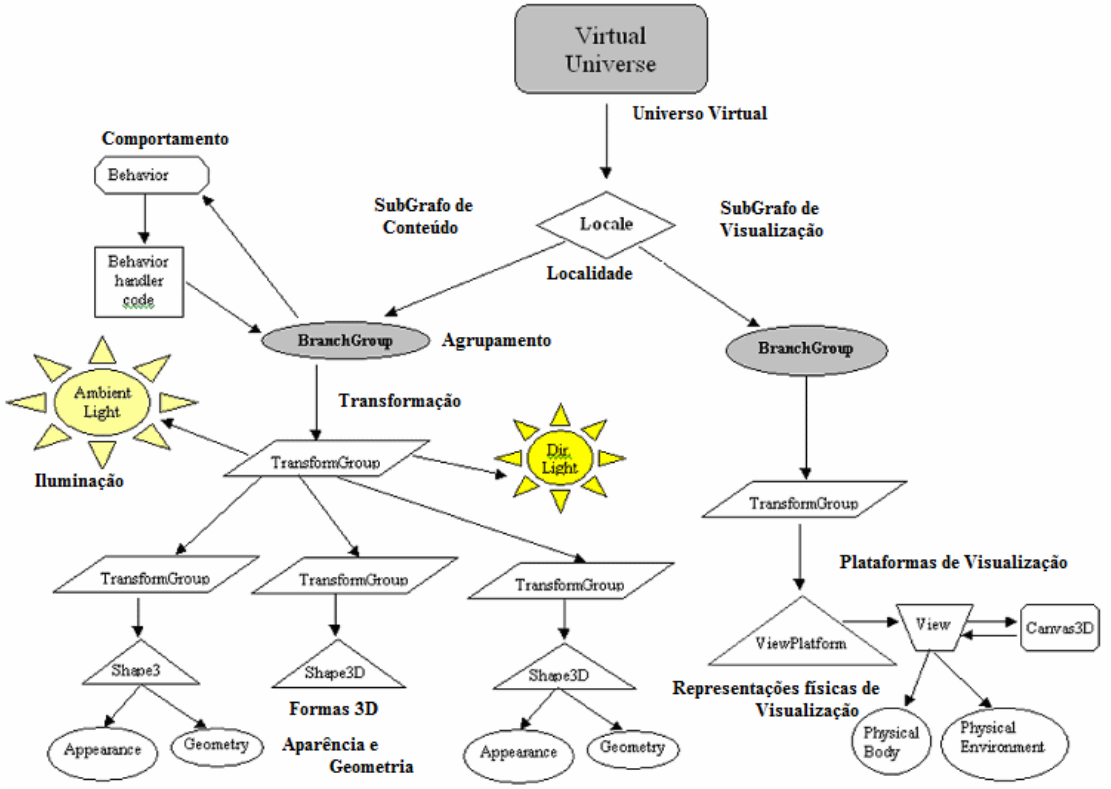


Figura 28-Representação da estrutura hierárquica de um Grafo de Cena habitual, onde os nodos de agrupamento são representados por elipses e os nodos "folha" são representados por triângulos. Pode-se também observar a subdivisão do Grafo em Subgrafo de Conteúdo, compreendendo a metade esquerda, e Subgrafo de Visualização, demonstrado pelo lado direito da estrutura.

3.3. Principais Classes da API Java 3D

A API Java 3D possui um grande número de classes implícitas e pré-programadas para instanciar, posicionar e manipular objetos gráficos modelados ou importados.

Nesta seção são apresentadas classes fundamentais para o desenvolvimento de um sistema Java 3D.

Uma classe de expressiva importância é a *SimpleUniverse*, responsável pela configuração de um universo abstrato, com toda programação embutida, é o topo do Grafo de Cena, onde toda aplicação Java 3D estará atrelada, provendo espaço para o desenvolvimento de uma cena tridimensional. Quando uma instância de *SimpleUniverse* é criada, são instanciados todos os objetos básicos necessários para o subgrafo de visualização, tais como *Locale*, responsável pelo espaço físico na cena, e *ViewingPlatform*, o qual formará os pontos de visualização, referente à *View*, que determina o espaço a ser renderizado. Outras classes necessárias são a *GraphicsConfiguration*, a qual faz parte do pacote *awt*, responsável pela descrição das características do dispositivo gráfico (impressora ou monitor) e a classe *Canvas3D*, que fornece o *Canvas*, ou seja, uma área de desenho ou trabalho, onde é realizada a visualização tridimensional (Sun, 2007).

A classe *BranchGroup* serve como o ponto base de um Grafo de Cena, ou seja, a raiz das transformações. Objetos desta classe são os únicos que podem ser atribuídos e associados em um objeto do tipo *Locale*. Assim, os *BranchGroups* são compilados, e inseridos em um universo virtual em tempo de execução.

Já a classe *Shape3D* é responsável por alocar uma geometria, descrita por pontos e retas, instanciadas ou importadas, de forma que a mesma possa ser manipulada por *Transform3D*, ter sua aparência e material ajustada pelas classes *Appearance* e *Material*, e organizadas pelos nodos *TransformGroup* e *BranchGroup*, e assim compor o subgrafo de conteúdo. Posteriormente, o mesmo *Shape3D* poderá ter seu comportamento modificado mediante *n* fatores ocorrentes na cena, dentre eles, eventos de mouse e teclado. Tal comportamento é instanciado por objetos *Behavior* e gerenciado por um *Handler*.

Transformações geométricas de escala, rotação e translação, são instanciadas através da classe de *Transform3D*, formando uma matriz 4x4 de números reais (*float*), chamada de matriz de transformações. Já os objetos da classe *TransformGroup* especificam uma transformação através de um objeto *Transform3D*, que será herdada a todos os seus filhos. Ao serem aplicadas as transformações, deve-se considerar que os efeitos num grafo de cena são cumulativos.

De forma simplificada, os passos e classes básicas para a criação de um programa Java 3D são: a criaçao de um objeto *GraphicsConfiguration* e de um *Canvas3D*, construção e compilação de pelo menos um subgrafo de conteúdo, criação de um objeto *SimpleUniverse*, que referencia objeto *Canvas3D*, construindo o subgrafo de visualização e a criação dos objetos *VirtualUniverse* e *Locale* para finalmente inserir os subgrafos no universo virtual. A Figura 29 demonstra a utilização das principais classes para a criação de um programa Java 3D, em forma de Grafo de Cena.

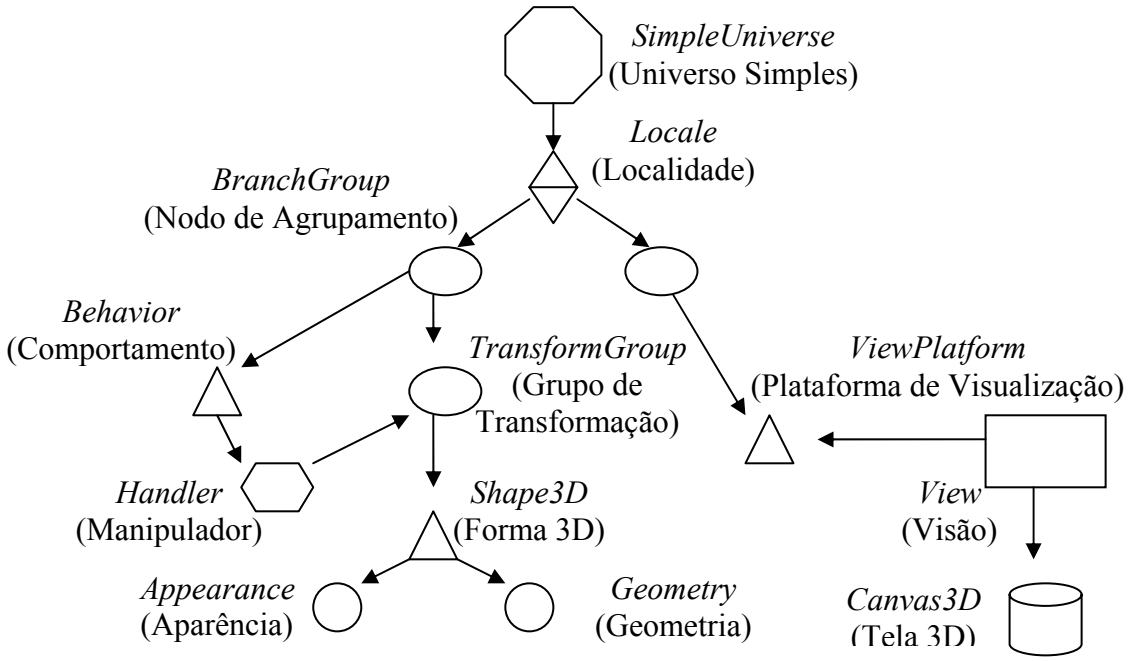


Figura 29 – Representação das principais classes da API Java 3D, imprescindíveis para a configuração de um sistema simples. Esta hierarquia deve conter ao menos um Locale, atrelado ao universo abstrado SimpleUniverse, para delimitar o espaço físico, um nodo de agrupamento BranchGroup para cada lado do grafo, uma ViewPlatform para manipular os modos de visualização sob um Canvas3D, gerenciados por um GraphicsConfiguration, e um Shape3D para armazenar as geometrias da cena.

3.4. Geometrias

Para representar entidades físicas ou fenômenos em Computação Gráfica, são utilizados modelos, os quais são definidos como padrões. Desta maneira a modelagem passa a ser uma etapa muito importante de um projeto, onde o modelo é descrito de forma que possa ser desenhado. Uma representação de um objeto deve ser feita de forma simples que facilite a usabilidade e análise. Atualmente, existem várias técnicas para a representação de modelos tridimensionais. Nesta seção, são apresentadas

formas de se representar um modelo em Java 3D, desde a sua geometria, delimitando sua forma física, até a sua aparência, e especificando as propriedades do material que a compõe, como cor, transparência e textura (Manssour, 2003).

3.5. Representação de Objetos

Para definir estruturas básicas em Java 3D da maneira mais simples possível, utiliza-se primitivas gráficas como *Box*, *Sphere*, *Cylinder* e *Cone*, disponíveis no pacote *com.sun.j3d.utils.geometry*. Sua utilização baseia-se em instâncias de classes, as quais levam os mesmos nomes das primitivas (Sun, 2007).

Uma outra forma de se representar objetos graficamente é utilizando-se listas de vértices, arestas ou faces poligonais. Neste caso, malhas de polígonos representam superfícies compostas por faces planas, que podem ser triângulos (preferencialmente) ou quadrados. Objetos simples e complexos são modelados desta maneira. O nodo *Shape3D* é usado para definir um objeto em Java 3D. Instâncias desta classe referenciam um nodo *Geometry* e um nodo *Appearance*. *Geometry* é uma superclasse abstrata que tem como subclasses, por exemplo, *GeometryArray* e *Text3D*. A classe *GeometryArray* também é uma classe abstrata, e suas subclasses são usadas para especificar pontos, linhas e polígonos preenchidos, tal como um triângulo. Algumas de suas subclasses são: *QuadArray*, *TriangleArray*, *LineArray*, *PointArray* e *GeometryStripArray*, que, por sua vez, tem *LineStripArray*, *TriangleStripArray* e *TriangleFanArray* como subclasses. Cada uma destas classes possui uma lista de vértices que podem ser conectados de diferentes maneiras. Além disso, objetos *GeometryArray* também podem armazenar coordenadas do vetor normal, de cores e de texturas.

Com o Java 3D também pode-se importar dados geométricos, como conjunto de ponto, arestas ou faces, criados por outras aplicações, ou seja, trazer para dentro de um ambiente virtual, modelos desenvolvidos com ferramentas específicas. Neste caso, arquivos de formato *Wavefront* (.obj) podem ser importado através da classe *Loader*, do pacote *com.sun.j3d.loaders*, que fornece subsídios para sua implementação. Outros formatos como o VRML (.wrl) também podem ser trazidos ao ambiente Java 3D, através de pacotes específicos como o *Xj3D* (Xj3D, 2007) e o *VRML97* . Entretanto, suas classes principais de importação devem ser manualmente programadas para que

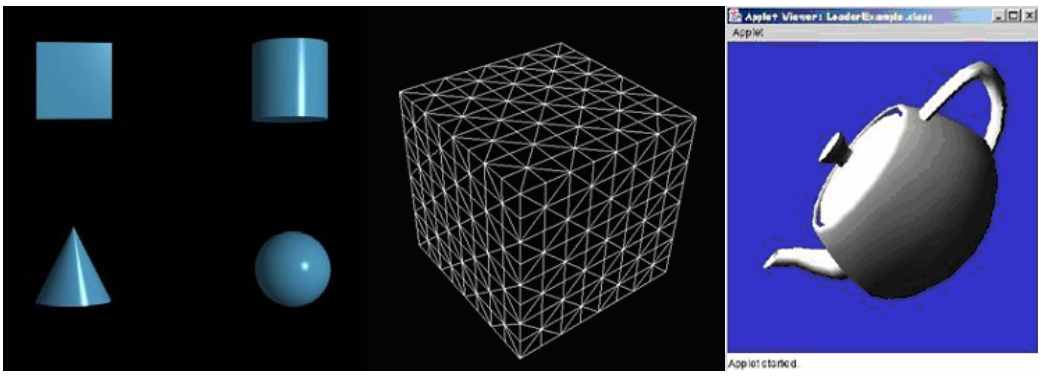
a cena armazenada possa ser representada em código Java 3D. Entre os arquivos que podem ser importados também encontram-se 3D Studio (.3ds) e AutoCAD (.dfx). A Figura 30 apresenta um trecho de código que utiliza um *Loader* de um modelo *Wavefront* (.obj) para importar um objeto modelado por ferramenta externa. A Figura 31(a) mostra exemplos de primitivas, a Figura 31(b) demonstra uma geometria renderizada em *wireframe* e a Figura 31(c) mostra um modelo *Wavefront* importado para o ambiente Java 3D através de classes *Loader*s.

```
ObjectFile f = new ObjectFile(ObjectFile.RESIZE,
(float)(60.0 * Math.PI / 180.0));
Scene s = null;

try
{
s = f.load( new
java.net.URL(getCodeBase().toString() + "./teapot.obj"));
}
catch (FileNotFoundException e) { … }
catch (ParsingErrorException e) { … }
catch (IncorrectFormatException e) { … }
catch (java.net.MalformedURLException ex) { … }

objRaiz.addChild(s.getSceneGroup());
```

Figura 30 – Representação de trecho de código que denota um Loader Wavefront, onde um modelo exportado por ferramentas externas de modelagem é importado e atribuído a um objeto Scene através do método load da classe ObjectFile, para ser posteriormente atribuído a um nodo de agrupamento e assim integrar a metade do Subgrafo de conteúdo. (Sun, 2007)



(a) (b) (c)

Figura 31 – Modelos de representação de objetos (a) Representação das primitivas Java 3D, definidas através de objetos *Sphere, Box, Cylinder e Cone*. (b) Representação de geometria renderizada em wireframe, através de objetos *GeometryArray* (Pavarini et al, 2005). (c) Representação do uso de um modelo Wavefront (.obj) através da classe *Loader* de *ObjectFile* (Manssour, 2003).

3.6. Aparência e Material

Quando são criadas primitivas gráficas, as mesmas não têm cores especificadas. Tal atributo é determinado pela classe *Appearance* e, se o mesmo não for instanciado, a forma geométrica apresentará a cor branca e outras propriedades como transparência, textura e material não poderão ser manipuladas. Alguns de seus métodos são listados na Figura 32, mostrando que esta classe faz referência a vários outros objetos.

```
void setColoringAttributes (ColoringAttributes coloringAttributes)
void setLineAttributes (LineAttributes lineAttributes)
void setTexture (Texture texture)
void setMaterial (Material material)
Material getMaterial ()
```

Figura 32 – Representação de trecho de código que demonstra a declaração de métodos da classe Appearance para definir atributos de cor, linhas, textura e material. Tais métodos são atribuídos a uma instância Appearance e posteriormente a um nodo de agrupamento, geralmente contendo uma geometria associada (Sun,2007)

Já o nodo *Material* é utilizado para definir a aparência de um objeto levando em consideração as diferentes fontes de iluminação. As vertentes a serem tratadas por esta classe são: cor ambiente, refletida da superfície do material, cujo valor *default* é (0.2, 0.2, 0.2); cor difusa, que consiste na cor do material quando iluminado e com valor *default* (1.0, 1.0, 1.0); cor especular do material, que tem branco como *default*; cor emissiva, isto é, a cor da luz que o material emite (preto por *default*); e *shininess*, que indica a concentração do brilho do material, que varia entre 1 e 128, sendo o *default* 64. A Figura 33 mostra a utilização de métodos da classe *Material*.

```
Appearance app = new Appearance();
Material material = new Material(new Color3f(0.8f,0.8f,0.1f),
new Color3f(0.0f,0.0f,0.0f),
new Color3f(0.8f,0.8f,0.1f),
new Color3f(1.0f,1.0f,1.0f), 100.0f);
app.setMaterial(material);
Cone cone = new Cone(0.4f, 0.8f);
cone.setAppearance(app);
```

Figura 33 – Representação de trecho de código que demonstra o uso de métodos da classe Material, onde são definidos os coeficientes especular, difuso, emissivo e brilho, e aplicados sobre uma instancia Appearance. Posteriormente é criado uma primitiva Cone e atribuído à mesma, todos os coeficientes definidos anteriormente em Material através da associação com a instancia Appearance (Sun, 2007)

3.7. Iluminação

Para trabalhar modelos iluminados, deve-se, primeiramente, definir qual a fonte de luz que será utilizada no ambiente, ou seja, o objeto que emite a energia. Os tipos existentes de fonte de luz são: ambiente, pontos de luz, luz direcional e *spot*.

Em Java 3D existe a classe abstrata *Light*, que define um conjunto de parâmetros em comum para todos os tipos de fonte de luz. Tais parâmetros são usados para definir a cor da luz e o grau de interação com sua região de influência. As suas subclasses são *AmbientLight*, *DirectionalLight* e *PointLight* que, por sua vez, tem *SpotLight* como subclasse (Selman, 2002).

A classe *AmbientLight* trata da iluminação proveniente de todas as direções, uniformemente. Seus construtores permitem ativar ou não sua cor. Já a classe *PointLight* é utilizada para especificar uma fonte de luz em um ponto fixo que espalha raios de luz igualmente em todas as direções. A classe *DirectionalLight* determina uma fonte de luz regular e com destino certo com origem no infinito. Estas fontes de luz contribuem para as reflexões difusa e especular. Nos seus construtores é possível definir a cor, a posição e o coeficiente de atenuação da fonte de luz .

Nos construtores da classe *SpotLight*, derivada da *PointLight*, os argumentos são direção, o ângulo de expansão e concentração da luz, sendo que a intensidade da luz é alterada em função do ângulo de expansão. Os trechos de código da Figura 34 demonstram a utilização destas classes para criar diferentes efeitos de iluminação. Em cada método é utilizada uma fonte de luz diferente (direcional, pontual e *spot*) e as reflexões: ambiente, difusa e especular são combinadas.

Posteriormente e não menos importante, é preciso definir o modo com que a luz irá interagir com os modelos virtuais, conforme descrito na seção anterior em *Material*, em termos de superfície e natureza da luz com o principal objetivo de se obter o efeito tridimensional em espaços bidimensionais, aproximado-se da realidade e contribuindo para a sensação de imersão.

3.8. Textura

Tipos diferentes de material possuem identificação distinta. Desta maneira, características próprias como microestruturas que produzem rugosidade na superfície

dos objetos podem ser simuladas com a utilização de imagens digitalizadas específicas.

```
// Exemplo de fonte de luz direcional
 Color3f corLuz = new Color3f(0.9f, 0.9f, 0.9f);
 Vector3f direcaoLuz = new Vector3f(-1.0f, -1.0f, -1.0f);
 Color3f corAmb = new Color3f(0.2f, 0.2f, 0.2f);
 AmbientLight luzAmb = new AmbientLight(corAmb);
 luzAmb.setInfluencingBounds(bounds);
 DirectionalLight luzDir= new DirectionalLight(corLuz,direcaoLuz);
 luzDir.setInfluencingBounds(bounds);
 objRaiz.addChild(luzAmb);
 objRaiz.addChild(luzDir);

 Material material = new Material(new Color3f(0.8f,0.8f,0.1f),
 new Color3f(0.0f,0.0f,0.0f), new Color3f(0.8f,0.8f,0.1f),
 new Color3f(1.0f,1.0f,1.0f), 100.0f);
```

Figura 34 – Representação de trecho de código que ilustra o uso de iluminação, onde são instanciadas uma luzes ambiente, mediante especificação de sua cor através do método Color3f e do tipo AmbientLight, e direcionais, através da definição da direção da luz sob um vetor tridimensional e da instanciação do tipo DirectionalLight. Ao final, os dois objetos de luz são atribuídos ao mesmo TransformGroup (Sun, 2007)

Segundo conceitos de Computação Gráfica, estas imagens da superfície de um objeto são chamadas de textura. Uma técnica de formação de texturas consiste simplesmente no mapeamento de uma imagem (mapa de textura/padrão de textura) para a superfície de um objeto. Para aplicar texturas em modelos desenvolvidos através da API Java 3D é necessário criar uma aparência, armazenar a imagem da textura e fazer a associação entre estes objetos. Também é preciso definir o posicionamento da textura na geometria, bem como os seus atributos (Manssour, 2003).

É importante salientar que as imagens devem ser previamente tratadas com a utilização de um programa externo à API Java3D e deve estar em um formato compatível com o Java3D (JPG, GIF ou PNG)[3]. Além disso, o seu tamanho deve ser múltiplo de dois em cada dimensão. Esta imagem pode estar armazenada em um arquivo local ou em uma URL. Para carregar a textura utiliza-se uma instância da classe *TextureLoader*, que deve ser associada com um objeto *Appearance*. No final,

[3] JPEG (Joint Photographic Experts Group), GIF (Graphic Interchage Format) e PNG (Portable Network Graphics)

devem-se especificar as coordenadas de textura da geometria. Nesta etapa, as posições da textura em uma geometria são determinadas por coordenadas, as quais são definidas por vértices e pontos de textura. Conseqüentemente, uma imagem pode ser esticada, rotacionada ou duplicada.

Já para modelos importados, a aplicação de texturas funciona através de um processo ligeiramente diferente, utilizando-se das classes *TextureLoader e Toolkit*, responsáveis por atribuir uma textura a um objeto do tipo *Appearance*, a qual deve ser instanciada no momento em que o modelo importado, de uma ferramenta de modelagem ou sistema de reconstrução, está sendo carregado, por meio do método *getImage*, utilizando-se de suas instâncias internas (objetos de aparência).

3.9. Interação

Uma interação ocorre quando o usuário recebe uma resposta, física ou meramente visual, às suas ações sobre o sistema, como pressionar uma tecla ou mover o *mouse*, cujo objetivo é alterar o Grafo de Cena. Tais reações são permitidas através da subclasse abstrata *Behavior* que, bem como no tratamento de animações, viabiliza alterações no grafo de cena como, por exemplo, a remoção de modelos ou alguns de seus atributos (Sun, 2007).

A classe *Behavior* é responsável por interpretar as ações do usuário e traduzi-las em alterações para o sistema, ou seja, faz a conexão entre a ação e a reação. Existem dentro desta classe, subclasses muito utilizadas para interação, por exemplo, a *MouseBehavior* e a *Keynavigatorbehavior* , que controlam eventos de *mouse* e teclado, respectivamente sobre o subgrafo de conteúdo.

Uma outra forma de interagir com os objetos em Java 3D é através dos métodos da classe *OrbitBehavior*. Desta maneira, é a *View* do ambiente que é deslocado, ou seja, os modelos permanecem estáticos enquanto é movida apenas a órbita (subgrafo de visualização). Fazem parte desta classe ações como rotação, translação e *zoom*. A Figura 35 apresenta um trecho de código que ilustra a utilização de métodos de interação.

Outras classes e *interfaces*, externas à API Java 3D também podem fornecer métodos de interação do usuário com a cena tridimensional, ainda que façam parte de outros pacotes. É o caso da *interface MouseListener* pertencente ao pacote *awt.event*,

a qual permite a interação do usuário com a cena por intermédio de eventos de *mouse*, como por exemplo o pressionar e o soltar de um botão.

```
BranchGroup scene = criaGrafoDeCena();
universe = new SimpleUniverse(canvas);

ViewingPlatform viewingPlatform = universe.getViewingPlatform();
viewingPlatform.setNominalViewingTransform();

// Adiciona "mouse behaviors" à "viewingPlatform"
OrbitBehavior orbit = new OrbitBehavior(canvas,
OrbitBehavior.REVERSE_ALL);
BoundingSphere bounds = new BoundingSphere
(new Point3d(0.0,0.0,0.0), 100.0);
orbit.setSchedulingBounds(bounds);
viewingPlatform.setViewPlatformBehavior(orbit);
universe.addBranchGraph(scene);
```

Figura 35 – Representação de trecho de código que ilustra métodos de interação, onde um objeto OrbitBehavior, sob as fronteiras de um BoundingSphere, executa a rotação da visão atual, considerando-se que tais instâncias estão associadas com Subgrafo de visualização, representados pela presença de um objeto ViewPlatform, o qual terá o comportamento atrelado ao objeto OrbitBehavior (Sun,2007)

No Capitulo IV é apresentada uma descrição conceitual e metodológica do presente trabalho, envolvendo todos os processos necessários para o desenvolvimento de um ambiente de Realidade Virtual dedicado à inspeção de amostras agrícolas, apresentando os objetivos, justificativa, as técnicas e procedimentos adotados, organizados sob o formato de classes. Posteriormente são descritos os dispositivos não convencionais empregados em fases de manipulação e visualização de tais amostras, apresentando suas características e especificações técnicas.

Capítulo IV
Estruturação Conceitual e Metodológica

4.1. Objetivo

A estruturação conceitual e metodológica aplicada no desenvolvimento do sistema de Realidade Virtual (RV) dedicado à inspeção de amostras tomográficas de solos agrícolas, utiliza dados de imagens reconstruídas com um algoritmo paralelo de reconstrução volumétrica. Através de uma *interface* com o usuário, os dados volumétricos são levados ao ambiente de RV por meio de classes capazes de importar as amostras em formato específico, as quais são fundamentadas em métodos de manipulação e visualização tridimensional. Por intermédio de recursos computacionais gráficos, o sistema busca somar imersão e interação do usuário com as amostras, o que se constituem como elemento essencial para um sistema qualitativo de visualização e análise por RV. Tais recursos computacionais gráficos envolvem predominantemente controle de renderização, iluminação, colorização, extração de atributos e transformações físicas, além da integração de dispositivos não convencionais de entrada e saída de dados, tais como um *Head-Mounted-Display* (vídeo-capacete) e uma luva digital, o sistema busca somar imersão e interação do usuário com as amostras, A Figura 36 mostra uma visão geral do sistema de RV dedicado à inspeção de amostras tomográficas de solos agrícolas, bem como o

caminho dos dados, onde, a partir de dados tomográficos, tais amostras são reconstruídas, importadas e tratadas mediante diversos processos de RV, até a aplicação do mesmo em de ciência de solos.

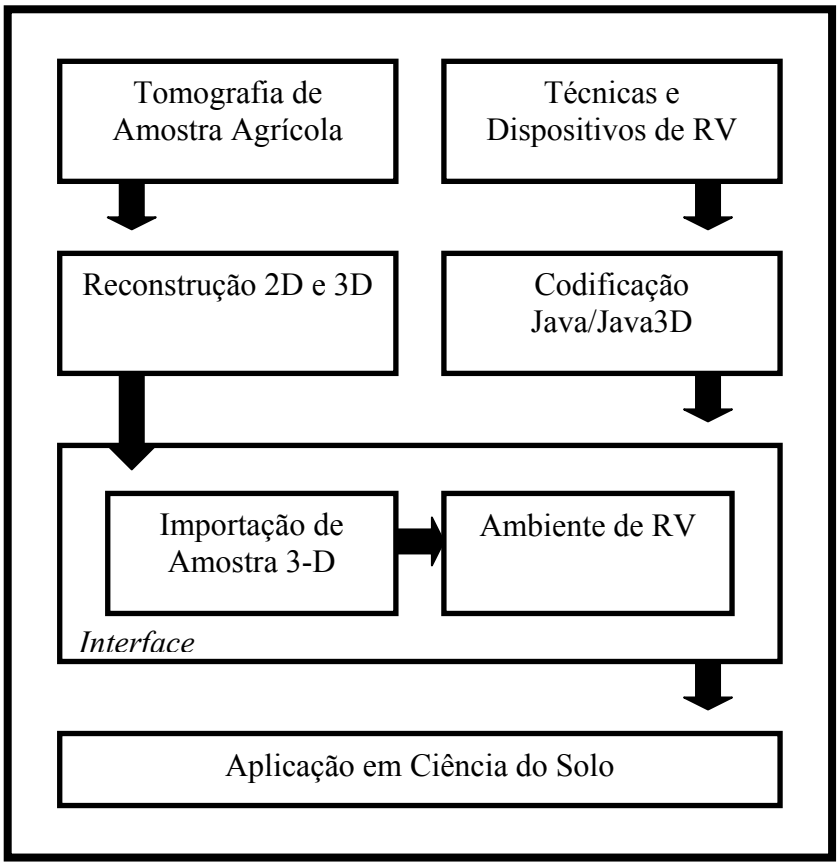


Figura 36 – Visão geral do sistema de RV dedicado à inspeção de amostras tomográficas de solos agrícolas e o caminho dos dados desde a aquisição até a visualização.

4.2. Metodologia

O sistema foi organizado com base nas seguintes classes: Reconstrução, *Loader*, Transformações, Extração de Atributos Polígonos, Filtro, Transparência, Iluminação, Colorização, Colisão Convencional, Colisão Não Convencional, Manipulação Convencional de Modelo, Manipulação Não Convencional de Modelo, Manipulação Convencional de Cena, Manipulação Não Convencional de Cena, Quatérnios, Visualização e Ambiente de RV.

As funcionalidades das classes são descritas a seguir, bem como são representadas pelo diagrama de classes apresentado na Figura 37.

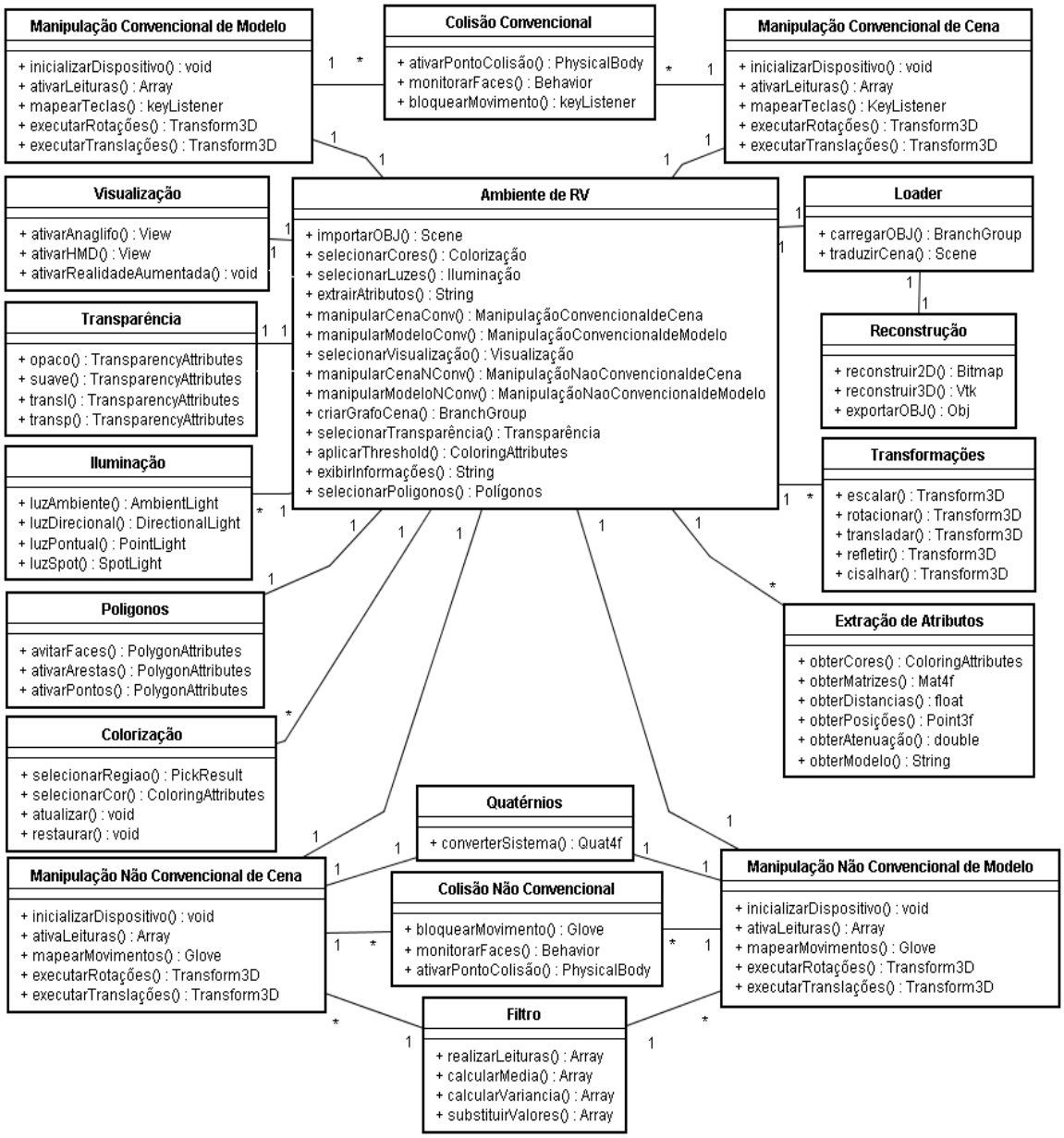


Figura 37 – Diagrama das classes que compõem o trabalho e suas relações.

4.2.1. Reconstrução

Para a obtenção dos dados tomográficos, foi utilizado o minitomógrafo de raios-*x* e γ da Embrapa Instrumentação Agropecuária (EMBRAPA-CNPDIA) (Cruvinel, 1987) para a aquisição das projeções tomográficas tornando possível gerar mapas de coeficientes de atenuação ($\mu = cm^{-1}$), com resolução espacial igual ou maior a 1 mm, com as amostras submetidas ao processo de aquisição sob energia de

60 KeV e janela temporal de 10 segundos de exposição à radiação por amostragem (Figura 38).

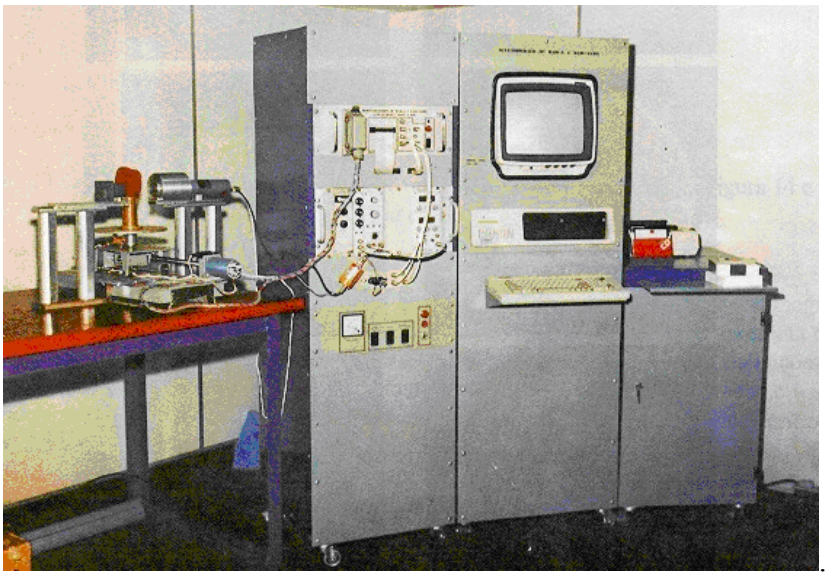


Figura 38 – Minitomógrafo da Embrapa Instrumentação Agropecuária

Para a reconstrução das imagens bidimensionais, foi utilizado o algoritmo de retroprojeção filtrada, juntamente com filtragens no domínio da freqüência, utilizando-se de janelas de *Hamming*, implementadas sob o algoritmo de Transformada Rápida de Fourier Unidimensional (FFT 1-D).

Posteriormente, com as imagens bidimensionais já reconstruídas, foi utilizada a técnica de filtragem bidimensional adaptativa, através da Transformada *Wavelet Daubechies*, com janelas de 76 coeficientes (D76).

Para o desenvolvimento da filtragem bidimensional, com o uso da Transformada *Wavelet*, o algoritmo levou em conta as características da imagem, filtrando determinadas áreas, preservando bordas e detalhes dos objetos representados nas imagens. Assim, o primeiro passo para a filtragem *Wavelet* baseado na decomposição pela aplicação da *Transformada Wavelet Daubechies* (D76) (Daubechies, 1992), sendo que cada função *Wavelet* pode ser aplicada de maneira específica, visando uma melhor eficiência de processamento. Desta maneira, a imagem foi subdividida em quadrantes, onde o quadrante 1 (superior esquerdo), após a primeira decomposição, abrigou os valores de aproximação da imagem decomposta, enquanto que os outros quadrantes armazenavam os coeficientes de detalhes. Assim, a imagem transformada pôde ser restaurada a partir de informações de borda e ruído.

A decomposição é um dos conceitos mais importantes em análise *Wavelet*, chamada também de análise de multiresolução. A mesma consiste em uma seqüência de espaços de aproximação $V_j \subset V$, que são encaixantes, isto é, $V_{j-1} \subset V_j$. Isto possibilita a representação dos dados em multi-nível, em termos da soma direta, na forma:

$$V_j = V_j \oplus W_j \oplus \Lambda \oplus W_{j-1} \tag{4.1}$$

Em que *j* representa o nível menos refinado e W_l contém os detalhes da representação em V_{l+1} que não podem ser representados em V_l.

Os espaços de aproximação V_j são gerados pela base:

$$\{\phi_{jk}(x), k \in Z\} \tag{4.2}$$

definida como:

$$\phi_{jk}(x) = 2^{j/2}\phi(2^j x - k) \tag{4.3}$$

ou seja, são obtidas pela dilatação e translação de uma função base $\phi(x)$.

A função $\phi(x)$ é chamada função de escala e satisfaz a relação:

$$\phi(x) = 2\sum_{k \in Z} h(k).\phi(2x - k) \tag{4.4}$$

O conjunto de coeficientes *h*(*k*) é chamado de filtro e estes são obtidos como raízes de um polinômio que satisfaz certas condições para a existência da relação (Daubechies, 1992).

Uma função $\Psi(x)$ é chamada de *Wavelet*, se o conjunto:

$$\{\Psi_{jk}(x), k \in Z\} \tag{4.5}$$

é uma base para os espaços complementares W_j , em que:

$$\Psi_{jk}(x) = 2^{j/2}\Psi(2^j x - k) \qquad (4.6)$$

Como $\Psi \in W_0 \subset V_1$ existem os coeficientes g_k tais que:

$$\Psi(x) = 2\sum_k g(k).\phi(2x-k) \qquad (4.7)$$

A forma mais difundida para a reconstrução tridimensional de superfícies e volume é a da sobreposição de fatias bidimensionais. Tal método consiste em posicionar as imagens tomográficas 2-D ao longo do eixo *Z* de coordenadas cartesianas. Para otimizar este processo, visando a redução de custos e menor exposição do objeto à radiação, foi adotado o uso de dados interpolados procurando-se compor matematicamente o espaço deixado pelo minitomógrafo entre as imagens. Assim, com apenas algumas fatias, o algoritmo pode estimar os pontos faltantes. O processo de reconstrução implementado é demonstrado pela Figura 39:

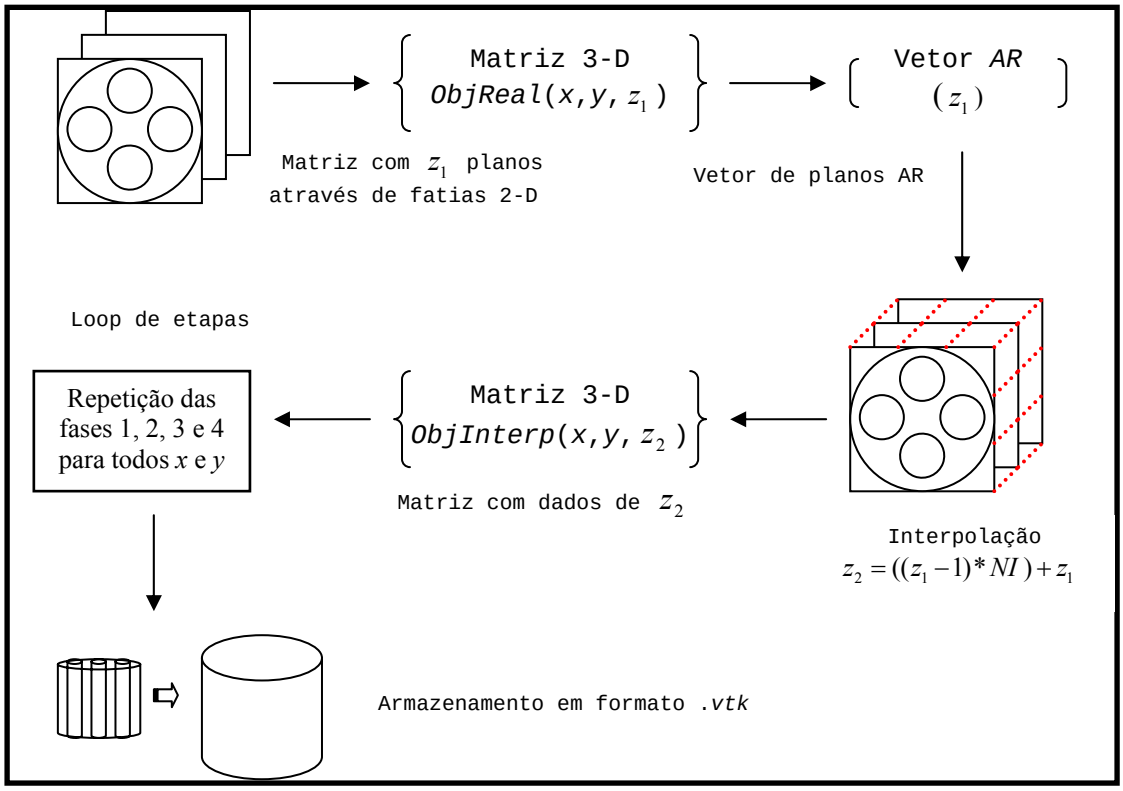


Figura 39 – Representação do processo de reconstrução tridimensional, desde obtenção das fatias bidimensionais e a formação de matrizes tridimensionais, até a fase de interpolação utilizando-se de B-Splines (B-Wavelets), onde ocorre a fase de estimação de pontos e a formação do modelo tridimensional

Com os dados das fatias bidimensionais, obtidas pela reconstrução bidimensional ao longo da coordenada z de um objeto, determinaram-se os valores de uma matriz de dimensão três. Essa matriz, *ObjReal* (x, y, z_1), armazena nas coordenadas x e y os dados dos z_1 planos reconstruídos bidimensionalmente.

Com base na matriz de dados reais *ObjReal* (x, y, z_1), armazenou-se em seguida, um vetor unidimensional os z_1 elementos da coordenada horizontal (x) e vertical (y) fixa, o qual contém os dados conhecidos do processo de interpolação.

Posteriormente, procedeu-se uma interpolação unidimensional dos dados contidos no vetor unidimensional, de forma a se obter um vetor de z_2 elementos contendo os dados interpolados. O número de elementos z_2 foi determinado por:

$$z_2 = ((z_1 - 1) * NI) + z_1 \tag{4.8}$$

Onde *NI* é o total de planos a serem interpolados.

Nesta etapa foi aplicado o algoritmo de interpolação por *B-Wavelets* (ou *B-Splines*), o qual determinou os valores intermediários entre uma seqüência de pontos conhecida. Diferentemente da aproximação, a interpolação não apenas desloca a curva gerada sob a influência dos pontos conhecidos, como também faz com que essa curva passe por esses pontos.

Para descrever a interpolação faz-se necessária uma discussão sobre bases convolucionais. Inicialmente, a idéia de base convolucional deve satisfazer a seguinte equação:

$$\psi_k(x) = \beta(x - k) \tag{4.9}$$

onde β é a função base.

Em outras palavras, a base foi construída a partir de deslocamentos de inteiros de uma função singular β(x), desta maneira, utilizando-se da equação (4.9) tem-se que:

$$f(x) = \sum_{k \in K} c_k \beta(x - k) \tag{4.10}$$

Avaliando a função f (x) na equação (4.10) sob um valor inteiro *n*, foi obtida a equação:

$$f(n) = \sum_{k \in K} c_k \beta(n - k) \tag{4.11}$$

a qual tem exatamente a forma de uma convolução discreta.

A função base β(x) sob valores inteiros, foi digitalmente convolvida com o vetor de coeficientes para produzir valores amostrados da função f (*x*).

Pode-se inverter a equação (4.11) para obter os coeficientes a partir de f (n) por deconvolução.

De acordo com a idéia de bases convolucionais, a interpolação torna-se um procedimento de dois passos. O primeiro passo é a direta inversão da equação (4.11), onde os coeficientes wavelets são descobertos por deconvolução da função amostrada f (n) com o filtro fatorizado β(n). O segundo passo reconstrói a função contínua f (x) de acordo com a equação (4.10). Os dois passos podem ser combinados, entretanto é mais conveniente usar-se da separabilidade.

De forma a otimizar o processo de cálculo, implementa-se a função β(x), também chamada de função *blending*, da seguinte maneira (Minatel, 1997):

$$\beta = \begin{bmatrix} 1/6(2+x)^3 & -2 \le x \le -1 \\ 1/6(4-6x^3-2x^3 & -1 \le x \le 0 \\ 1/6(4-6x^3+2x^3 & 0 \le x \le 1 \\ 1/6(2-x)^3 & 1 \le x \le 2 \\ 0 & 2 \le |x| \end{bmatrix} \tag{4.12}$$

Nesta implementação, adotou-se o uso dos chamados pontos fantasmas. Essa técnica consiste em considerar um ou mais valores antes do ponto inicial e depois do final.

Foram adicionados dois pontos fantasmas no inicio e dois no final da seqüência conhecida. Esses valores foram ajustados de forma dinâmica para que a curva a ser gerada pela função *B-Wavelet* passasse pelos pontos inicial e final.

A seguir, os dados tridimensionais foram armazenados em outra matriz 3D *ObjInterp*(x, y, z_2), as etapas anteriores para todos os valores de x e y repetidas, e os dados de *ObjInterp* ( ) gravados em disco com a extensão *Visualization Toolkit* (*.vtk)*.

Ao final do processo, o modelo tridimensional foi convertido para o formato *Wavefront File Format* (.obj), utilizando-se da classe *vtkOBJExporter* presente no pacote *vtkOBJExporter.h* do *Visualization Toolkit*. A escolha desse formato justifica-se pelo alto desempenho e flexibilidade ao importar tais modelos para ambientes virtuais, onde todos seus atributos podem ser customizados por API's gráficas.

4.2.2. Ambiente de Realidade Virtual

Para o desenvolvimento do Ambiente de RV, foi utilizada a linguagem de programação Java sob versão 1.6.0 e a API Java3D sob versão 1.5.0., possibilitando a construção do grafo de cena do sistema, organizado sob estrutura hierárquica (Sun, 2007).

Desta maneira, uma *interface* principal com o usuário foi desenvolvida como um controle principal do sistema, a qual provê um acesso intuitivo a todos os recursos descritos nesta seção, representados por todas as outras classes do diagrama da Figura 4.2. A Figura 40 representa a forma com que tais recursos são organizados para formar a classe Ambiente de Realidade Virtual.

Assim, o modelo tridimensional proveniente do algoritmo de reconstrução foi importado pela classe *Loader*. Tal modelo é tratado pelo ambiente de RV mediante as ferramentas de computação gráfica disponíveis, como o controle de exibição de polígonos, tratamento de cor por região de influência, seleção de luzes, transformações físicas lineares e controle de exibição por *threshold*.

Desta forma, independente de prévio tratamento, os usuários podem empregar ferramentas de RV para a realização de análise, navegação e exploração das amostras tridimensionais, tais como a extração de atributos das mesmas, seleção do modo de visualização da cena e controle de manipulação da cena e das amostras.

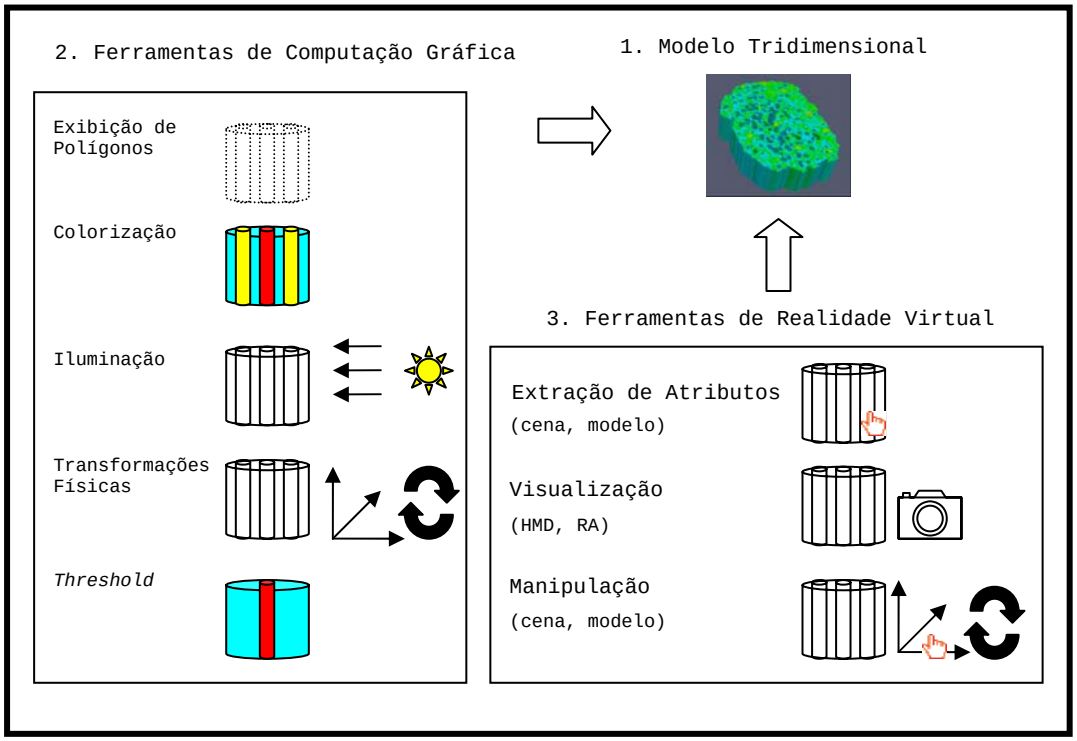


Figura 40 – Visão geral do ambiente de Realidade Virtual, onde o modelo 3-D reconstruído é tratado sob diversas técnicas imersivas e interativas, além de ter suas informações extraídas por técnicas de seleção.

4.2.3. Transformações Físicas Lineares

A classe *Transformações físicas lineares*, consideradas como recursos interativos básicos de um sistema de RV, foram inicialmente incorporadas ao Ambiente. Assim, operações como rotação, translação, escala, alongamento/encurtamento e reflexões foram disponibilizadas através de entradas por teclado para todo o conjunto de dados das amostras, realizadas através de instâncias da classe *Transform3D*, do pacote *javax.media.j3d.Transform3D*, utilizando de uma matriz de transformação 4x4 de números reais (*float*). Tal pacote utiliza-se dos métodos *setRotation*, *setTranslation* e *setScale* para efetivar as operações e posteriormente, nodos de conteúdo da classe *TransformGroup*, derivados do pacote *javax.media.j3d.TransformGroup*, responsáveis por atrelar as transformações de *Transform3D* ao grafo de cena do sistema, garantindo que transformações sejam inerentes a todos os filhos, considerando a hereditariedade e o efeito cumulativo da API. O Pseudocódigo 1 demonstra como foram aplicados os métodos de *Transform3D* e *TransformGroup* nos modelos.

```
Procedimento Transformações Lineares:
Begin
   Novo Agrupamento de Transformações;
   Novo Agrupamento de Geometrias;
   Nova Transformação3D;
   Agrupamento de Transformações.Ajustar Transformação(Transformação3D);
   Agrupamento de Geometrias.Obter Geometrias(Agrupamento de Transformações);
   Transformação3D.Ajustar Escala(Fator de Escala);
   Transformação3D.Ajustar Translação(Novos Pontos);
   Transformação3D.Ajustar Rotação(Eixo Escolhido, Ângulo Escolhido);
   Agrupamento de Transformações Pai.Adicionar Filho(Agrupamento de Geometrias);
End.
```

Pseudocódigo 1 – Aplicação de métodos *Transform3D* de transformações tridimensionais e *TransformGroup* de agrupamento de transformações para a realização de transformações físicas

4.2.4. Polígonos

A classe *Polígonos* é responsável por permitir ao usuário a forma com que os polígonos formadores do modelo tridimensional são representados pelo renderizador da API Java3D (*OpenGL* ou *DirectX*). No presente trabalho, o sistema permite que os usuários escolham entre três formas de renderização dos polígonos: Faces, Arestas e Pontos (*Polygon Fill*, *Wireframe* e *Point Cloud*), viabilizadas pela classe *PolygonAttributes* do pacote *javax.media.j3d.PolygonAttributes*, onde os modelos 3-D são visualizados com todas suas faces, apenas suas arestas ou apenas seus pontos, respectivamente. Todas as opções tem o algoritmo de *Z-Buffer ativado*, com o intúito de omitir do renderizador partes oclusas e aumentar o desempenho. O Pseudocódigo 2 demonstra a alteração do modo de renderização de polígonos de um objeto *Shape3D*.

```
Procedimento Polígonos:
Begin
   Novo Atributo de Polígonos(exibir arestas, sem esconder pontos);
   Nova Aparência;
   Aparência.Ajustar Polígonos(Atributo de Polígonos);
   Novo Atributo de Polígonos(exibir pontos, sem esconder pontos);
   Nova Aparência;
   Aparência.Ajustar Polígonos(Atributo de Polígonos);
End.
```

Pseudocódigo 2 – Aplicação de métodos de renderização de polígonos

4.2.5. Threshold

Esta classe é capaz de acessar visões diferenciadas da amostra importada, permitindo a separação dos componentes formadores de uma amostra, para uma

análise individual, possibilitadas por operações de *threshold* com diversas escalas de cor, aplicadas sobre o algoritmo de reconstrução. Na interface, tais operações determinam a visão completa (sem threshold) e componentes (com threshold). Desta maneira, o usuário é habilitado a visualizar apenas uma região de interesse da amostra, representando níveis específicos de tons de cinza, onde tons mais claros (ou cores quentes, no caso de pseudocores) representam altos coeficientes de atenuação em $cm^{@1}$ e tons mais escuros representam baixos coeficientes de atenuação em $cm^{@1}$, dependendo das características apresentadas pelas amostras agrícolas.

Para a aplicação da classe *threshold* nos modelos tridimensionais, foram adotadas as classes *ColoringAttributes* do pacote *javax.media.j3d.ColoringAtributes*, responsável por controlar as cores intrínsecas dos modelos 3-D, independentes de efeitos de iluminação, e a classe *Material* com suas subclasses de reflexão *AmbientColor*, *DiffuseColor*, *EmissiveColor* e *SpecularColor*.

Para se obter os tons de cinza dos modelos, é necessário conhecer os tons dos canais *RGB (vermelho/verde/azul)*, desta maneira, para obter tais coeficientes, foram utilizados os métodos *getAmbientColor*, *getDiffuseColor*, *getEmissiveColor* e *getSpecularColor*, todos derivados dos métodos *getAppearance().getMaterial()*, e que retratam os índices de reflexão ambiente, difuso, emissivo e especular, respectivamente. Posteriormente, foi possível obter os coeficientes *HSV (matiz/saturação/valor)*, dentre os quais o valor de "*V*" correspondente à luminância (ou brilho), representa os tons de cinza de um determinado ponto ou região. Os coeficientes HSV foram obtidos contendo:

$$H = \begin{cases} 60 \times \dfrac{G-B}{MAX-MIN} + 0, & se\ MAX = R\ e\ G \geq B \\ 60 \times \dfrac{G-B}{MAX-MIN} + 360, & se\ MAX = R\ e\ G < B \\ 60 \times \dfrac{B-R}{MAX-MIN} + 120, & se\ MAX = G \\ 60 \times \dfrac{R-G}{MAX-MIN} + 240, & se\ MAX = B \end{cases} \qquad (4.13)$$

$$S = \frac{MAX - MIN}{MAX} \qquad (4.14)$$

$$V = MAX \qquad (4.15)$$

onde R, G e B representam respectivamente os coeficientes de tons de vermelho, verde e azul (*red, green, blue*), e MAX e MIN os maiores e menores valores de *RGB***.**

A partir destes valores e dos parâmetros fornecidos como limites superior e inferior, a classe *threshold* se encarrega de exibir apenas os coeficientes das classes *ColoringAttributes* e *Material* presentes no intervalo estabelecido, obedecendo uma escala de 0 (zero) a 255 (duzentos e cinqüenta e cinco) tons de cinza.

Assim, os valores que estiverem abaixo ou acima do intervalo, são automaticamente omitidos do processo de renderização do modelo tridimensional. O Pseudocódigo 3 demonstra como foram aplicados os métodos de oclusão dos coeficientes de *ColoringAttributes* e *Material* nos modelos.

```
Procedimento Threshold:
Begin
  Se Luminância > Threshold Superior ou Luminância < Threshold Inferior
    Begin
      Novo Agrupamento de Geometrias;
      Agrupamento de Geometrias.Obter Geometrias(GrupodeTransformações)
      Nova Aparência;
      Aparência.Obter Aparência(Agrupamento de Geometrias);
      Novo Atributos de Cor;
      Atributos de Cor.Nova Cor(Preto);
      Aparência.Ajustar Cor(Atributos de Cor);
      Novo Material;
      Material.Ajustar Reflexão Ambiente(Preto);
      Material.Ajustar Reflexão Especular(Preto);
      Material.Ajustar Reflexão Emissiva(Preto);
      Material.Ajustar Reflexão Difusa(Preto);
      Aparência.Ajustar Material(Material);
    End;
End.
```

Pseudocódigo 3 – Aplicação de métodos de oclusão de instâncias ColoringAttributes e Material

4.2.6. Transparência

A classe *Transparência* responde por fornecer ao usuário a capacidade de controlar em tempo real a porção das faces subjacentes do modelo 3-D que é mostrada pela face que o encobre. No presente trabalho, o sistema permite que os usuários escolham entre quatro níveis de transparência dos polígonos: Opaco, Suave, Translúcido e Transparente (*Opaque*, *Smooth*, *Translucent* e *Transparent*) viabilizadas pela classe *TransparencyAttributes* presente no Java3D, onde faces oclusas dos modelos 3-D podem ser melhor visualizadas em ordem decrescente,

mediante a aplicação do método *setTransparency*. À medida que instâncias da classe *TransparencyAttributes* são criadas, o algoritmo de *Z-Buffer* deixa de executar. O Pseudocódigo 4 demonstra a aplicação de instâncias da classe Transparência sobre um objeto *Shape3D*.

```
Procedimento Transparência:
Begin
  Novo Atributo de Transparência;
  Atributo de Transparência.Ajustar Transparência(Opaco);
  Nova Aparência;
  Aparência.Ajustar Transparência(Atributo de Transparência);
  Novo Atributo de Transparência;
  Atributo de Transparência.Ajustar Transparência(Suave);
  Nova Aparência;
  Aparência.Ajustar Transparência(Atributo de Transparência);
  Novo Atributo de Transparência;
  Atributo de Transparência.Ajustar Transparência(Translúcido);
  Nova Aparência;
  Aparência.Ajustar Transparência(Atributo de Transparência);
  Novo Atributo de Transparência;
  Atributo de Transparência.Ajustar Transparência(Transparente);
  Nova Aparência;
  Aparência.Ajustar Transparência(Atributo de Transparência);
End.
```

Pseudocódigo 4 – Aplicação de métodos de controle de opacidade com *TransparencyAttributes*

4.2.7. Extração de Atributos

A classe de *Extração de Atributos* (também chamada de classe de *picking*) trata da obtenção de dados de *voxels* da amostra, utilizando como entrada dispositivos convencionais, como o mouse, e não convencionais como a luva de dados *P5Glove*, fornecendo aos usuários informação sobre um ponto específico da representação tridimensional.

Inicialmente, é necessário ativar o reconhecimento de eventos de cursor bidimensional na *interface* principal, onde reside o *Canvas3D*, implementando a interface *MouseListener* do pacote *java.awt.event*. Através de seu método *MouseClicked*, toda a interface do sistema, inclusive o objeto *Canvas3D* passa a identificar eventos de clique de *mouse* e a executar as rotinas inseridas em seu contexto.

Posteriormente, dentro do método *MouseClicked*, deve-se instanciar os objetos das classes *PickCanvas* e *PickResult*, os quais são responsáveis por ativar a extração

de dados de um objeto *Canvas3D* e por armazenar tais dados em vetores de resultados de eventos, respectivamente. Desta maneira, utilizando-se dessas instâncias, o usuário decide pela sua região de interesse e extrai os atributos desejados, onde a coordenada *z* é estabilizada no display, permitindo a seleção através de uma *viewport* bidimensional de maneira intuitiva.

Instanciado o objeto da classe *PickResult*, é necessário checar a existência de resultados no vetor através do método *getObject*, armazenando o produto da consulta em um objeto da classe *Node*, o qual armazena instâncias indefinidas ou representações com tipo desconhecido. A seguir, é aplicado sobre tal produto, um recurso de *cast*, ou seja, uma instanciação forçada com o tipo *Shape3D*, o qual tem a capacidade de armazenar diversos dados a respeito de uma determinada geometria. Desta maneira, o modelo escolhido é necessariamente um objeto *Shape3D*, com todas as características que tal tipo provê.

Assim, os dados disponíveis para operações de *picking* sob instâncias de *Shape3D* e seus respectivos métodos são: as fronteiras, com *getBounds*; os grafos de cena, com *getLocale* e *numBranchGraph*; as geometrias, com *getGeometry*; as *ColoringAttributes*, com *getAppearance.getColoringAttributes*; o material sob os formatos *HSL* e *RGB*, com *getAppearance.getMaterial*; a transparência, com *getAppearance.getTransparencyAttributes.getTransparency*; e os polígonos, com *getAppearance.getPolygonAttributes.getPolygonMode*.

Em seguida, um novo objeto é instanciado, pertencente à classe *PickIntersection*, também do pacote *com.sun.j3d.utils.picking*, responsável por abrigar o ponto de colisão entre uma entidade/nodo e o cursor bidimensional. Desta maneira, esta instância armazena em seu conteúdo o produto da interseção entre uma entidade de *PickResult* com o ponto escolhido do *Canvas3D*, passado ao método *getClosestIntersection* como parâmetro. Assim, a instância da classe *PickIntersection* pode fornecer mediante seus eventos: a distância entre o ponto e o observador, com o método *getDistance*; as coordenadas do vértice mais próximo, com o método *getClosestVertexCoordinates*; as coordenadas do ponto, com o método *getCoordinates*; a reta normal do ponto, com o método *getNormal*; e as matrizes de transformação com o método *getMatrix*.

Alem das instancias das classes *PickIntersection* e de *PickResult*, a classe *Extração de Atributos* conta com a obtenção do coeficiente de atenuação linear. Neste

contexto, essa medida e obtida através do nível dos tons de cinza, representada aqui pela luminância, índice “L” do padrão HSL, obtido pelo método *getMaterial*. O Pseudocódigo 5 demonstra a aplicação de métodos de extração de atributos.

```
Procedimento Extração de Atributos:
Begin
  Novo Nodo;
  Nodo:= Vetor de Resultados.Obter Objeto Clicado;
  Novo Agrupamento de Geometrias;
  Agrupamento de Geometrias.Extrair Geometrias(Nodo);
  Novo Ponto de Intersecção;
  Ponto de Intersecção:=Vetor de Resultados.Obter Intersecção Mais Próxima(x,y,0);
  Agrupamento de Geometrias.Exibir(Agrupamento de Geometrias.Fronteiras, Agrupamento
de Geometrias.Geometria, Agrupamento de Geometrias.Grafo de Cena, Ponto de
Intersecção.Distância, Ponto de Interseccção.Vértice Mais Próximo, Ponto de
Intersecção.Coordenadas do Ponto, Ponto de Intersecção.Reta Normal);
  Agrupamento de Geometrias.Exibir(Agrupamento de Geometrias.Atributos de Cor,
Agrupamento de Geometrias.Material, Agrupamento de Geometrias.Atributos de
Transparência, Agrupamento de Geometrias.Atributos de Polígonos, Luminância, Saturação,
Coeficiente de Atenuação Linear, Elemento Químico);
End.
```

Pseudocódigo 5 – Aplicação dos métodos de extração de atributos

4.2.8. Manipulação Convencional de Cena

A classe de *Manipulação Convencional de Cena* representa a troca dinâmica do posicionamento das câmeras do ambiente de visualização em tempo real, ou seja, os modelos tridimensionais permanecem fixos em uma posição determinada enquanto a cena virtual tem seus pontos de vista alterados. Tal exercício na presente classe utiliza-se de dispositivos convencionais de entrada de dados, como o teclado, o qual permite a manipulação da cena de forma intuitiva e simples.

Inicialmente, é necessário definir logo no início da criação do Grafo de Cena, um objeto da classe *ViewingPlatform*, responsável por realizar a conexão com a *View*, ou seja, a implementação de qualquer operação sobre a plataforma de visualização do ambiente, tais como rotações e translações das câmeras do mesmo, também realizadas através de matrizes de transformação 4x4, de forma análoga aos objetos das classes *Transform3D* e *TransformGroup* descritos anteriormente.

Posteriormente, foi implementada uma classe auxiliar estendida de das classes *ViewPlatformBehavior* e *KeyBehavior*, ambas estendidas da classe *Behavior*, responsáveis por determinar o comportamento da plataforma de visualização à medida

que determinadas teclas são acionadas, identificando-as e exercendo alguma influência sobre a cena.

A implementação de uma classe estendida de *KeyBehavior* sugere o uso de quatro métodos fundamentais: *Initialize*, *Process Stimulus*, *KeyPressed* e *Process KeyEvent*. O método *Initialize* determina o modo com que a classe é inicializada, no caso ao se pressionar uma determinada tecla. Neste método, através de critérios de início do tipo *WakeupCriterion*, dois novos critérios de eventos foram definidos: KEY_PRESSED e KEY_RELEASED, responsáveis por inicializar os mesmos ao pressionar e ao liberar uma tecla, através da instanciação de objetos da classe *WakeupOnAWTEvent*.

O método *Process Stimulus* trata da continuidade das ações, ou seja, o que fazer quando a próxima tecla é pressionada, organizando os eventos através de instâncias *Enumeration*, passado ao método como parâmetro, definindo os próximos passos do processo. Também, o mesmo é responsável por chamar o método *processKeyEvent*, o qual finalmente efetiva a realização dos movimentos da cena, onde instâncias *Transform3D* e *TransformGroup* são utilizadas para as operações, baseando-se nas entradas do dispositivo como parâmetros de métodos *setTranslation* e *setRotation*, de forma semelhante à classe de Transformações.

Já o método *KeyPressed*, não menos importante, é utilizada para compor eventos adicionais ao se pressionar uma determinada tecla, como a criação de um grafo de cena adicional ou até mesmo a inserção de novos componentes em um grafo pré-programado No presente trabalho, o método *KeyPressed* é utilizado para implementar um dos estudos de caso em ciência de solos, onde o caminho percorrido pelas câmeras é armazenado e demarcado com instância *Shape3D*, anexadas ao grafo de cena previamente implementado pela classe *Ambiente de RV*.

Finalmente, é necessário delimitar a região de influência onde os movimentos da cena irão atuar, ou seja, até quais coordenadas tridimensionais as câmeras podem ser movimentadas. Tal região de influencia é instanciada pela classe *BoundingSphere*, do pacote *javax.media.j3d.BoundingSphere*, onde as fronteiras são delimitadas pelo raio de uma esfera centrada na origem das coordenadas. Posteriormente, a instancia *BoundingSphere* é adicionada à cena por meio da chamada do método *setSchedulingBounds* e a instanciação de um objeto da classe *KeyBehavior* conectada ao objeto de *ViewingPlatform* através do método *setViewingPlatform*. O

Pseudocódigo 6 demonstra a utilização dos métodos da classe de *Manipulação Convencional de Cena* e a Figura 41 apresenta a aplicação prática da classe.

```
Procedimento Manipulação
Convencional de Cena:
Begin
Procedimento Estímulo:
  Begin
    Novo Evento;
    Enquanto Durar o Evento
      Begin
        Se o Evento for Tecla
Pressionada
          Processa Evento(Evento);

      End;
  End;
Procedimento Mover Para Frente:
  Begin
    Mova(Novo Vetor de
Posição(0,0,Taxa de Translação);
  End;
Procedimento Rotacionar X:
  Begin
    Rotacione(Taxa de Rotação);
  End;

Procedimento Mova:
  Begin
    Nova Transformação3D;
    Grupo de Transformações.Obter Transformação(Transformação3D);
    Transformação3D.Translação(Vetor de Posição);
    Atualizar Matriz de Transformação;
    Fronteira Esférica.Ajustar Parâmetros(Raio, Origem);
    Nova Plataforma de Visualização;
    Plataforma de Visualização.Ajustar Comportamento de Cena( Manipulação
Convencional de Cena);
  End;

Procedimento Rotacione:
  Begin
    Nova Transformação3D;
    Grupo de Transformações.Obter Transformação(Transformação3D);
     Transformação3D.Rotação X(Taxa de Rotação);
     Atualizar Matriz de Transformação;
     Nova Fronteira Esférica.Ajustar Parâmetros(Raio, Origem);
     Nova Plataforma de Visualização;
     Plataforma de Visualização.Ajustar Comportamento de Cena(Manipulação
Convencional de Cena);
  End;
End.
```

```
Procedimento Inicializar:
  Begin
    Novos Critério de Disparo;
      Critério de Disparo.Tecla Pressionada;
      Critério de Disparo.Tecla Liberada;
  End;
```

```
Procedimento Processa Evento:
  Begin
    Novo Código;
    Código.Obter Código do Evento;
    Se Código for “Frente”
      Mover Para Frente;
    Se Código for “Frente + Shift”
      Rotacionar X;
  End;
```

Pseudocódigo 6 – Utilização dos métodos de manipulação convencional de cena.

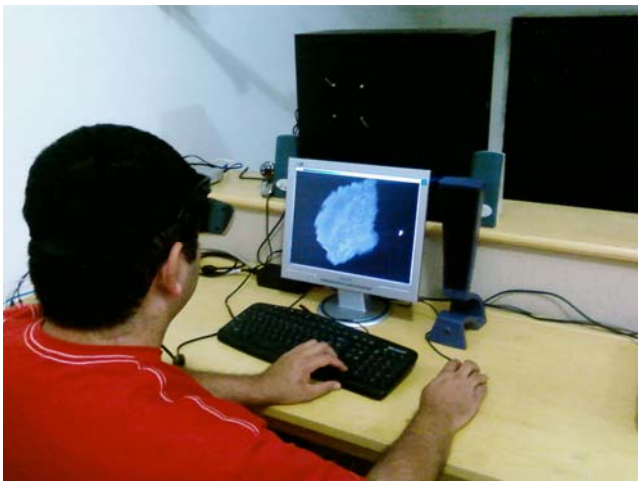


Figura 41 – Representação da Manipulação Convencional

4.2.9. Manipulação Convencional de Modelo

A classe de *Manipulação Convencional de Modelo* representa a troca dinâmica e em tempo real do posicionamento dos modelos no ambiente de visualização, ou seja, as câmeras da cena permanecem fixas em uma posição determinada enquanto os modelos têm sua posição e orientação alterada. Tal exercício, na presente classe, também se utiliza de dispositivos convencionais de entrada de dados, como o teclado e o mouse, os quais permitem a manipulação do modelo de forma intuitiva e simples.

Para a efetiva implementação de tais eventos de manipulação, a API Java3D dispõe de três classes previamente programadas denomidadas *KeyNavigatorBehavior*, *MouseTranslate* e *MouseRotate*, responsáveis por translações e rotações de teclado, translações por mouse e rotações por mouse respectivamente.

A classe *KeyNavigatorBehavior*, análoga à classe descrita anteriormente, é estendida de *KeyBehavior*, sendo capaz de ler, interpretar e executar alguma ação mediante eventos do teclado. Desta maneira, semelhante à movimentação de câmeras, o pressionar das teclas definem rotações e translações dos modelos tridimensionais por intermédio de métodos semelhantes ao *Initialize*, *Process Stimulus*, *processKeyEvent* e *KeyPressed*, diferindo apenas no alvo das transformações, uma vez que as ações ocorrem sobre instâncias da classe *TransformGroup* que abrigam representações tridimensionais e não mais plataformas de visualização.

A partir da classe *MouseTranslate*, estendida da classe *Behavior*, o nodo de transformação que abriga o modelo tridimensional no grafo de cena, pode ser alterado em tempo de execução por meio de movimentos do mouse, especificamente com o botão direito pressionado, resultando em translações da representação tridimensional por toda a cena sintética. Da mesma forma que na classe *KeyNavigatorBehavior*, a classe *MouseTranslate* funciona com base nos quatro métodos anteriormente citados, os quais podem ser customizados para a realizar a movimentação desejada.

Análogo ao funcionamento de *MouseTranslate*, a classe *MouseRotate* do pacote *com.sun.j3d.utils.behaviors.mouse.MouseRotate* tem como base os mesmos métodos, diferindo apenas no modo com que a mesma atua sobre o grafo de cena, onde os nodos de transformação tem suas matrizes multiplicadas por matrizes de rotação, alterando a orientação dos modelos em tempo de execução do grafo através de movimentos do mouse aliados ao botão esquerdo.

Depois de instanciados, os objetos das classes citadas têm imediatamente suas fronteiras de atuação delimitas pelo método *setSchedulingBounds*, cujo parâmetro é uma instância da classe *BoundingSphere*, sugerindo que as fronteiras de atuação da classe de Manipulação Convencional de Modelo seja esférica e com raio embutido em tal parâmetro. Posteriormente, os objetos das classes de manipulação são associados ao nodo *TransformGroup*, o qual mantêm ligação direta com o objeto *Shape3D* que armazena a representação tridimensional que será manipulada. A implementação da presente classe e de suas bases seguem a mesma idéia da *Manipulação Convencional de Cena*, diferindo apenas no alvo de aplicação, uma vez que o produto das mesmas é aplicado sobre o nodo *TransformGroup* que abriga o modelo tridimensional.

4.2.10. Manipulação não Convencional de Cena

A classe de *Manipulação não Convencional de Cena* configura-se como uma das grandes responsáveis pela interatividade e imersão do usuário com o Ambiente de Realidade Virtual, uma vez que permite que o utilizador navegue pela cena sintética em todas as direções, se aproxime e adentre as estruturas reconstruídas utilizando-se da luva de dados *P5Glove*.

Para a realização de tais eventos, as classes de manipulação, tanto de cena quanto de modelo, baseiam-se em outra classe auxiliar denominada *FPSGlove*

presente no pacote *com.essentialreality*, fornecido pelo fabricante do dispositivo e melhorada por desenvolvedores (Kenner, 2007), provendo uma interface programável entre o desenvolvedor e o *hardware* da luva de dados.

A classe *FPSGlove* é a responsável por abrigar todos os parâmetros referentes ao dispositivo não convencional, no que diz respeito ao posicionamento, orientação e dobra dos dedos, tornando possível detectar a proximidade e a inclinação no mesmo, e assim disparar uma série de eventos customizáveis. Desta maneira, em princípio são declaradas *flags* constantes que indicam se a luva está próxima ou distante da torre de recepção. Posteriormente, novas *flags*[4] registram o posicionamento vertical e horizontal do dispositivo.

De maneira análoga, novas constantes delimitam os *triggers* responsáveis por iniciarem determinados eventos à medida que seus valores são ultrapassados. Por exemplo, quando a luva ultrapassa uma determinada altura, a câmera começa a se mover para cima e assim por diante. Tais *triggers* existem também para inclinação da mão e dobra dos dedos, ambos determinados por ângulos mínimos.

Posteriormente, já no método construtor da classe, outros parâmetros de igual importância são ativados, tais como *P5_Init*, *P5_setForward*, *P5_setMouseState*, P5*_setFilterAmount* e *P5_setRequiredAccuracy*, responsáveis por inicializar, determinar a direção positiva, desligar o mouse, filtrar o sinal e determinar a precisão de movimentos, respectivamente.Em seguida, são declarados os métodos responsáveis por detectar a posição da luva no ambiente real. São eles o *getXPosition*, *getYPosition* e *getZPosition*, os quais mapeiam as *triggers* antes citadas para disparar um tipo de evento, ou seja, monitora os valores recebidos pela luva através de instâncias da classe *P5State*, classe responsável por determinar o estado atual da luva. Assim, por intermédio do método *filterPos* de *P5State*, a exata posição do dispositivo é obtida e então atribuída aos métodos para checar se os limites foram ultrapassados.

De forma análoga aos métodos de detecção de posicionamento, ainda na classe *FPSGlove*, são descritos os métodos *getYaw*, *getPitch* e *getRoll*, os quais respondem por detectar a inclinação do dispositivo nos eixos Y, X e Z, determinando se os limites estabelecidos pelas *flags* foram atingidos.

[4] *Flag* (Bandeira): mecanismo lógico que funciona como semáforo; ativada se a característica associada a essa *flag* estiver presente.

Na classe *FPSGlove*, são ainda definidos os métodos: *isClenched*, para determinar se as dobras do dispositivos estão flexionadas, também regido por *flags* que determinam os limites, e *isAPressed*, *isBPressed* e *isCPressed*, os quais resolvem o pressionar do botão A, B e C da luva de dados, respectivamente.

Depois de implementados os monitores e *triggers* de eventos com a classe auxiliar, a *Manipulação não Convencional de Cena* deve agora ter seus eventos descritos em seu escopo. Para tal, a classe deve primeiramente ser estendida da classe *ViewingPlatform*, ou seja, ter suas instâncias interpretadas como eventos sobre a cena virtual. Em seguida, dois parâmetros específicos são incluídos, ou seja, o passo de translação e o passo de rotação, responsáveis por definirem o quanto os modelos virtuais serão deslocados ou inclinados a cada movimento do dispositivo no mundo real, depois de reconhecidos pela classe *FPSGlove*. Os próximos passos configuram-se em implementar os métodos *Initialize* e *ProcessStimulus*, de forma semelhante aos métodos de manipulação convencional de cena e modelo, onde os eventos de despertar das ações e as próprias são descritas, respectivamente. O Pseudocódigo 7 demonstra a implementação de *Manipulação Não Convencional de Cena* e a Figura 42 apresenta a utilização prática da classe.

Finalizado o processo, a instancia da classe de *Manipulação Não Convencional de Cena* devem ser atrelada ao objeto da classe *ViewingPlatform* do atual objeto *Canvas3D*, para que todos os movimentos afetem a cena e não o modelo trdimensional, através do método *setViewingPlatform*.

Para a efetiva realização de uma translação de cena, uma instância de *FPSGlove* é criada para a chamada dos métodos *getXPosition*, *getYPosition* e *getZPosition*, armazenando a real posição do dispositivo no mundo real e movimentando a cena virtual na direção apontada pelo mesmo. Para tal ação, a classe utiliza-se dos mesmos nodos *Java3D* dos processos de manipulação convencional ou mesmo de transformações lineares não interativas: o nodo de agrupamento de transformações da classe *TransformGroup* e as efetivas transformações da classe *Transform3D*, por intermédios dos métodos de *setTranslation* e *setRotation*, utilizando como parâmetro os ângulos e posições reconhecidos pelo objeto de *FPSGlove*, movimentando a cena na direção real desejada. De forma análoga à *Manipulação Convencional de Cena*, os caminhos percorridos podem ser demarcados e armazenados para posterior renderização.

```
Procedimento Manipulação Não Convencional De Cena:
Begin

Mínimo para Rotacionar Z Esquerda;
Mínimo para Rotacionar Z Direita;
Mínimo para Considerar Perto;
Mínimo para Considerar Longe;

Procedimento Obter Posição De Z:
Begin
  Posição Z = Estado Da Luva.Posição Filtrada De Z;
  Se Posição Z < Mínimo Para Considerar Perto
    Retorna Perto;
   Se Posição Z > Mínimo Para Considerar Longe
     Retorna Longe;
   Senão
     Retorna Meio;
End;

Procedimento Obter Rotação De Z:
Begin
  Rotação Z= Estado Da Luva.Orientação Filtrada De Z;
  Se Rotação Z<Mínimo Para Rotacionar Z Esquerda
    Retorna Rotacionar Z Esquerda;
  Se Rotação Z>Mínimo Para Rotacionar Z Direita
    Retorna Rotacionar Z Direita;
  Senao Retorna Meio;
End;

Procedimento Obter A Pressionado:
Begin
  Novo Vetor De Botões Pressionados;
  Vetor De Botões Pressionados=Estado Da Luva.Botão Pressionado;
  Retorna Botões Pressionados[Posição Zero];
End;

  Novo Vetor Frente=(0,0,-Passo De Translação);
  Novo Vetor Tras=(0,0,Passo De Translação);
  Novo Vetor Cima=(0,Passo De Translação,0);
  Novo Vetor Baixo=(0,-Passo De Translação, 0);

Procedimento Inicializar:
Begin
  Nova Condição De Gatilho;
  Condição De Gatilho.Tempo Decorrido;
  Ativar Condição;
End;

  Nova Fronteira Esférica;
  Fronteira Esférica.Ajustar Parâmetros(Raio, Origem);
  Nova Plataforma de Visualização;
  Plataforma de Visualização.Ajustar Comportamento de Cena(Manipulação Não
Convencional de Cena)
End.
```

```
Constantes
  Perto;
  Longe;
  Meio;
  Rotacionar_Z_Esquerda;
  Rotacionar_Z_Direita;
  Passo De Rotação:=3 Graus;
  Passo De Translação:=0.05 Metros;
  Nova Luva;
  Luva.Direcao De Z(Para Trás);
  Luva.Estado Do Mouse(Desligado);
  Luva.Ajustar Filtro(Médias);
```

```
Procedimento Estimulo
Begin
  Atualizar Luva;
  Nova Posição Da Mão;
  Posição Da Mão:=Luva.Obter Posição De Z;
  Se Posição Da Mão=Perto
    Movimentar(Frente);
  Se Posição Da Mão=Longe
    Movimentar(Trás);
  Senão Meio;
  Nova Orientação Da Mão;
  Orientação Da Mão:=Luva.Obter Rotação De Z;
  Se Orientação Da Mão=Rotacionar Z;
    Rotacionar Z(-Passo De Rotação);
  Se Orientação Da Mão=Rotacionar Z Direita
    Rotacionar Z(Passo De Rotação)
  Senão Meio
End;
```

Pseudocódigo 7 – Pseudocódigo da Manipulação Não Convencional de Cena

Figura 42 – Representação da Manipulação Não Convencional

4.2.11. Manipulação Não Convencional de Modelo

A implementação de *Manipulação Não Convencional de Cena* é a classe responsável por realizar a movimentação das representações tridimensionais através de movimentos reais da luva de dados *P5Glove*, onde o usuário pode alterar o posicionamento e orientação dos modelos em tempo real sob todas as direções e ângulos, contribuindo diretamente para a interatividade do ambiente virtual, em seis graus de liberdade.

Para viabilizar o processo, análogo à *Manipulação não Convencional de Cena*, a presente implementação utiliza a classe de apoio *FPSGlove*, descrita anteriormente como a detentora dos *flags* que reconhecem a posição e orientação do dispositivo não convencional. Todos os eventos que aqui se utilizem de *FPSGlove* são necessariamente idênticos à classe anterior, diferindo apenas do alvo ao qual se aplicam as transformações.

Assim, a *Manipulação não Convencional de Modelo* inicia-se estendida da classe *Behavior*, denominando-se uma classe que descreve comportamentos, customizados para reações aos movimentos do dispositivo. Desta maneira, logo são descritos os passos de translação e de rotação, determinando o quanto o modelo deve se movimentar a cada deslocamento da luva.

Também de forma semelhante à classe anterior, os comportamentos da *Manipulação não Convencional de Modelo* são regidos por dois métodos básicos de

comportamento: *Initialize* e *processStimulus*, detectando a posição e a orientação do dispositivo exatamente da mesma maneira, entretanto, com alguns pontos adicionais.

Após atribuir os métodos *getXPosition*, *getYPosition, getZPosition* para obter o posicionamento, e os métodos *getYaw*, *getPitch* e *getRoll*, para obter as orientações, sob instâncias da classe *FPSGlove*, é chamado um novo método, denominado *rotateQuaternion*, responsável por converter rotações realizadas sob ângulos de *Euler* em coordenadas de *Quatérnios*, estabelecendo rotações com números complexos e eixos imaginários, a fim de contribuir com a precisão de movimentos.

O método *rotateQuaternion* passa à sua classe os parâmetros de eixo e ângulo, em coordenadas de *Euler* e retorna a descrição de um quatérnio, a set utilizado no método construtor *Quat4f*, construindo um quatérnio de *float* e posteriormente em *setRotation*, efetivando uma rotação com instâncias da classe *Transform3D*.

Ao final do processo, o produto da classe de *Manipulação não Convencional de Modelo* é encapsulado sob a forma de um objeto *BranchGroup* e atribuído ao grupo de transformação *TransformGroup* que rege os movimentos da representação tridimensional, de forma distinta à classe anterior, fazendo com que toda a detecção de movimento e a efetiva mudança de posicionamento e orientação das entidades surta efeito sobre os atuais modelos no objeto *Canvas3D*. A implementação da presente classe e de suas bases seguem a mesma idéia da *Manipulação Não Convencional de Cena*, diferindo-se apenas no alvo de aplicação, uma vez que o produto das mesmas é aplicado sobre o nodo *TransformGroup* que abriga o modelo tridimensional.

4.2.12. Colisão Convencional

A classe denominada *Colisão Convencional* trata da implementação de um algoritmo de detecção de colisão agregado à classe de *Manipulação Convencional de Cena*, com seu funcionamento restrito a eventos que utilizem dispositivos convencionais de entrada de dados, especificamente o teclado. Assim, por meio de tal algoritmo, os usuários são impedidos de atravessar as faces dos modelos tridimensionais durante o processo de navegação na cena sintética, permitindo apenas a transposição das câmeras por dentre os espaços vazios entre tais faces, simulando processos físicos reais.

Para a implementação da classe em questão, foi adotada mais uma vez como extensão a interface *Behavior*, a qual fornece genericamente através de dois dos métodos anteriormente descritos, *Initialize* e *ProcessStimulus*, os subsídios para o reconhecimento e interpretação das teclas pressionadas para a identificação das ações que culminam na colisão entre uma determinada face do modelo tridimensional e a estrutura que representa o posicionamento do usuário na cena.

Para o método *Initialize*, são definidos três instancias da classe *WakeupCriterion*, determinando as condições de inicialização para a classe de colisão. São elas: objeto de *WakeupOnCollisionEntry*, a qual sinaliza quando uma colisão é iniciada; objeto de *WakeupOnCollisionExit*, que prevê a finalização de um processo de colisão, e objeto de *WakeupOnCollisonMovement*, a qual garante que a colisão continuará inicializada se a posição de choque entre as representações for mantida.

O método *processStimulus* de *Colisão Convencional* é o responsável por identificar uma série de teclas pressionadas, através de um parâmetro da classe *Enumeration*, análogo aos processos anteriores. A seguir, o método *processStimulus* interpreta tal série como instâncias de critérios de estímulo para sua execução, ou seja, traduz cada tecla verificando se a mesma é objeto *WakeupCollisionEntry*, *WakeupOnCollisionExit* ou *WakeupOnCollisonMovement*. Identificada a condição da colisão, o método *processStimulus* bloqueará os movimentos convencionais da cena na direção onde ocorre a colisão, ou seja, se a cena se move para frente, ao colidir com uma face de um modelo tridimensional, os movimentos das câmeras, provocados por tal tecla são neutralizados, até que qualquer tecla que leve a câmera pra outra direção, e que não conduza a outra colisão, seja pressionada. O mesmo ocorre com todas as outras teclas que movimentam as câmeras (setas do teclado), sobre todas as faces dos modelos. Tal bloqueio ocorre através do acesso aos objetos da classe de *KeyBehavior*, mencionada anteriormente. Esta classe armazena em quatro objetos individuais, as teclas pressionadas, representando as direções de movimentação da cena, usadas tanto para posicionamento quanto para orientação. Desta maneira, a classe de *Colisão Convencional*, por intermédio de chamadas publicas, instancia tais objetos com parâmetros nulos, fazendo com que as mesmas deixem de influenciar os movimentos da cena.

Entretanto, para que tal bloqueio de movimentos seja concretizado, é necessário posicionar alguma entidade na cena com a qual as faces dos modelos

tridimensionais se colidem. Desta maneira, faz-se uso de duas instâncias *Shape3D*, munidas de qualquer geometria, no caso esferas azuis de raio 0,5. Tais esferas são, a cada inicialização do sistema, associadas a um objeto da classe denominada *PhysicalBody*, que em *Java3D* delimita o posicionamento físico do observador da cena virtual. Assim, as esferas azuis simbolizam os olhos do observador, que, em choque com as faces do modelo, são impedidas de prosseguir, independente da direção. O Pseudocódigo 8 demonstra a instanciação de métodos de colisão.

```
Procedimento Colisão Convencional:
Begin
Procedimento Inicializar:
  Begin
    Novo Critério De Inicialização;
    Critério De Inicialização.Ao Encostar Na Face;
    Critério De Inicialização.Ao Desencostar Da Face;
    Critério De Inicialização.Ao Permanecer Encostado;
    Adicionar Lista De Critérios Ao Sistema;
  End;

 Senão Se Critério De Estimulo = Ao Desencostar Da Face
       Begin
         Se Tecla Travada = Seta Para Cima
            Destravar Seta Para Cima;
         Se Tecla Travada = Seta Para Baixo
            Destravar Seta Para Baixo;
         Se Tecla Travada = Seta Para Esquerda
            Destravar Seta Para Esquerda;
          Se Tecla Travada = Seta Para Direita
            Destravar Seta Para Direita;
          Exibir "Deixando Estado De Colisão"
       End;

    Senao Se Criterio De Estimulo = Encostado
      Exibir "Estado De Colisao Em Curso";
    End;
End.

Procedimento Estímulo:
  Begin
    Novo Critério De Inicialização=Próximo Elemento;
    Se Critério De Estimulo = Ao Encostar Na Face
      Begin
        Critério.Obter Objeto;
          Se Ultima Tecla Pressionada = Seta Para Cima
            Movimento Para Frente = Travado;
          Se Ultima Tecla Pressionada = Seta Para Baixo
            Movimento Para Baixo = Travado;
          Se Ultima Tecla Pressionada = Seta Para Esquerda
            Movimento Para Esquerda = Travado;
          Se Ultima Tecla Pressionada = Seta Para Direita
            Movimento Para Direita = Travado;
          Exibir "Estado De Colisão Iniciado";
      End;
```

Pseudocódigo 8 – Pseudocódigo da instanciação dos métodos de *Colisão Convencional*

4.2.13. Colisão Não Convencional

A classe denominada *Colisão Não Convencional* trata da implementação de um algoritmo de detecção de colisão agregado à classe de *Manipulação Não Convencional de Cena*, com seu funcionamento restrito a eventos que utilizem dispositivos não convencionais de entrada de dados, especificamente a luva de dados *P5Glove*. Desta maneira, por meio de tal algoritmo, os usuários são também

impedidos de atravessar as faces das representações tridimensionais durante o processo de navegação na cena sintética, permitindo apenas a transposição das câmeras por dentre os espaços vazios entre tais faces, simulando processos físicos reais.

De forma idêntica à *Colisão Convencional*, a presente classe é estendida da interface *Behavior* e baseia-se também em inicialização e estímulo. De forma idêntica à *Colisão Convencional*, o método *Initialize* instancia três critérios de inicialização: *WakeupOnCollisionEntry*, *WakeupOnCollisionExit* e *WakeupOnCollisionMovement*, definindo as condições em que a colisão não convencional se inicia.

Já o *processStimulus* apresenta algumas diferenças em relação à classe anterior. Ao processar e analisar a série de critérios passados como parâmetros, o método passa agora a interpretar os movimentos da luva, tendo como foco uma possível colisão. Novamente utilizando-se de instâncias da classe de Manipulação não Convencional de Cena, extensão de *Behavior*, cada posição espacial da luva é testada quanto à atual instância da mesma, ou seja, para qual direção a luva está se movimentando no atual momento, onde as possíveis alternativas são: esquerda, direita, para cima, para baixo, para frente e para trás.

Identificado o posicionamento da luva no momento de uma suposta colisão, a classe de *Colisão Não Convencional* realiza o bloqueio dos movimentos do dispositivo, de forma análoga à detecção da classe anterior. Assim, a ultima movimentação da luva ao colidir é imediatamente impedida de continuar, embora a luva de dados possa ser movimentada livremente no ambiente real. Tal ação é provocada por uma nova instanciação das atuais posições luva, atribuindo às mesmas, vetores zerados, ou seja, inicializados na origem, provocando a parada imediata dos movimentos do dispositivo.

Paralelamente, ao realizar qualquer outro movimento, que não os que levem a uma continuação do bloqueio, a classe os interpreta e permite continuar a série válida de movimentos, através da nova instanciação das posições mapeadas da luva, utilizando como parâmetro a posição onde a colisão foi iniciada e o passo linear adotado pela classe, um fator multiplicativo que representa a unidade cartesiana percorrida. Ao final do processo, de forma idêntica à *Colisão Convencional*, um *Shape3D* é adicionado ao *PhysicalBody* para realizar a detecção no processo de navegação do usuário na cena, permitindo que a visão do mesmo se choque com as

faces da representação tridimensional e desencadeie o algoritmo da classe descrita, impedindo que a navegação prossiga por entre a cena e permitindo a continuação à medida que qualquer outra direção de e/ou sentido de movimentação da luva seja acionado. O Pseudocódigo 9 demonstra a utilização da classe de *Colisão Não Convencional*.

```
Procedimento Colisão Não Convencional:
Begin
  Procedimento Inicializar:
    Begin
      Novo Critério De Inicialização;
      Critério De Inicialização.Ao Encostar Na Face;
      Critério De Inicialização.Ao Desencostar Da Face;
      Critério De Inicialização.Ao Permanecer Encostado;
    End;
  Procedimento Estímulo:
    Begin
       Se Critério De Estimulo = Ao Encostar Na Face
              Begin
          Critério.Obter Objeto;
          Se Ultimo Movimento Da Luva =  Para Cima
                Movimento Para Cima = Travado
          Se Ultimo Movimento Da Luva =  Para Baixo
                Movimento Para Baixo = Travado
          Se Ultimo Movimento Da Luva =  Para Frente
                Movimento Para Frente = Travado
          Se Ultimo Movimento Da Luva =  Para Trás
                Movimento Para Trás = Travado
          Se Ultimo Movimento Da Luva =  Para Esquerda
                Movimento Para Esquerda = Travado
          Se Ultimo Movimento Da Luva =  Para Direita
                Movimento Para Direita = Travado
          Exibir "Estado De Colisão Iniciado"
        End;
     Senão Se Critério De Estimulo = Ao Desencostar Da Face
              Begin
                Se Movimento Travado = Para Cima
                  Destravar Movimento Para Cima
                Se Movimento Travado = Para Baixo
                  Destravar Movimento Para Baixo
                Se Movimento Travado = Para Frente
                  Destravar Movimento Para Frente
                Se Movimento Travado = Para Trás
                  Destravar Movimento Para Trás
                Se Movimento Travado = Para Esquerda
                  Destravar Movimento Para Esquerda
                Se Movimento Travado = Para Direita
                  Destravar Movimento Para Direita
          Exibir "Deixando Estado De Colisão"
              End;
     Senão Se Critério De Estimulo = Permanecer Encostado
               Exibir "Estado De Colisão Em Curso"
  End;
End.
```

Pseudocódigo 9 – Pseudocódigo que demonstra os métodos de Colisão Não Convencional

4.1.14. Quatérnios

Conforme citado anteriormente no Capítulo 1, a descrição de rotações por ângulos de Euler configura-se como um sério problema para o sistema quando se espera realizar mudanças de orientação de cena ou de modelo em eixos não convencionais, ou seja, diferente dos padrões do sistema cartesiano.

Desta maneira, a classe de *Quatérnios* implementa um algoritmo de conversão para que o sistema deixe de utilizar apenas rotações sobre o eixos x, y e z , passando a realizar orientações sobre eixos intermediários, definidos por um vetor que passa pela origem e atinge um ponto no espaço, o qual é representado por uma coordenada específica do dispositivo real, como por exemplo as coordenadas cartesianas (*x, y, z*) de um dos oito *LED*'s da luva. Para realizar tal operação, a técnica utiliza-se de bases imaginárias e números complexos, fornecendo assim uma alternativa de parâmetro para o método *setRotation* da classe *Transform3D*, o qual permite a utilização de um quatérnio como argumento.

Primeiramente, deve-se compor a parte vetorial do vetor quatérnio através de:

$$s = sen\left(\frac{angulo}{2}\right) \tag{4.17}$$

$$v[i] = s \cdot axis[i] \tag{4.18}$$

onde a variável s é composta pelo seno da metade do $angulo$, parâmetro multiplicativo de inclinação da luva em uma direção aleatória. Já a variável v representa a parte vetorial propriamente dita, composta de três partes contadas por i, multiplicando o outro parâmetro *axis*, representado pelos eixos de *Euler* recebidos pela classe de conversão.

Em seguida, deve-se estipular a quarta parte (escalar) do quatérnio:

$$v[3] = \cos\left(\frac{angulo}{2}\right) \tag{4.19}$$

onde o valor escalar do quatérnio, ou quarta parte, é igual o cosseno da metade do ângulo de Euler recebido.

Desta maneira, ao chamar uma instância de *Quatérnios* ao se realizar uma rotação com o dispositivo não convencional *P5Glove*, a orientação da luva é interpretada pela classe *FPSGlove* e traduzida pela *Manipulação não Convencional de Cena* ou de *Modelo*, é convertida do sistema de Euler para a base de *Quatérnios*, retornando ao sistema novas coordenadas de orientação, a serem efetivadas pelo método *Quat4f* de *Transform3D*, a qual encapsula todo o funcionamento de um quatérnio descrito anteriormente. O Pseudocódigo 10 mostra a conversão de bases de Euler para *Quatérnios*.

```
Procedimento Quatérnios:
Begin
  Procedimento Conversão:
    Begin
      Novo Angulo De Euler;
      Novo Vetor De Eixos;
      Novo Fator De Rotação;
      Novo Flutuante;
      Flutuante = Seno (Angulo De Euler/Fator De Rotação);
      Novo Vetor De Quatérnios;
      Vetor De Quatérnios[Posição 0]=Flutuante*Vetor De Eixos[Posição 0]
      Vetor De Quatérnios[Posição 1]=Flutuante*Vetor De Eixos[Posição 1]
      Vetor De Quatérnios[Posição 2]=Flutuante*Vetor De Eixos[Posição 2]
      Vetor De Quatérnios[Posição 3]=Cosseno (Angulo De Euler/Fator De
Rotação)
     End;
End.
```

Pseudocódigo 10 – Pseudocódigo que demonstra a conversão do sistema de Euler para Quatérnios

4.2.15. Filtro

A luva digital *P5Glove*, conforme sua especificação possui um sistema óptico de transmissão de dados, por meio de emissão, recepção e decodificação de sinais infravermelhos, onde os 8 *LED's* os enviam de acordo com a visibilidade do dispositivo pela torre de recepção. Já na torre tais sinais são recebidos por um par de sensores fotossensíveis, capazes de determinar a posição/distancia da luva pela correlação dos raios que chegam a ela.

Como qualquer sistema óptico, a transferência de dados entre a luva *P5Glove* e a torre receptora, é suscetível a ruídos, principalmente o ruído branco. Desta maneira, o sinal recebido pela torre apresenta uma intermitente inconstância de

valores, os quais representam as coordenadas cartesianas (x, y, z) de cada *LED* emissor de luz.

A posição final da luva e de seus *LED's* nunca serão as reais posições no mundo real, bem como a posição das entidades que compõem o mundo sintético. Paralelamente, a representações tridimensionais e as câmeras da cena apresentam uma ligeira trepidação ao serem manipuladas e/ou exploradas utilizando-se do dispositivo não convencional, resultado da mesma inconstância de leitura dos dados infravermelhos.

Para solucionar o problema, foi implementada a classe *Filtro*, a qual se responsabiliza por receber os valores imprecisos provenientes do sinal original ruidoso e atualiza-los em valores precisos e constantes. Para tal atividade, foi adotada como apoio, a mesma classe que dá suporte às classes que controlam o dispositivo não convencional, denominada *FPSGlove*, a qual detêm toda a informação a respeito dos sensores sob os aspectos de posicionamento, orientação e dobras dos dedos.

Inicialmente, para dar início aos cálculos das novas posições e orientações dos *LED's* emissores, é necessário obter os parâmetros a serem utilizados para tal atividade, no caso, as primeiras coordenadas dos *LED's* da luva, as quais podem ser obtidas a partir da instância da classe *FPSGlove* através do método *Update*, responsável por inicializar todos os parâmetros da luva, executado em um laço de repetição de cinco vezes, a fim de capturar vinte amostras de posicionamento tridimensional para cada um dos oito *LED´s* emissores, ou seja, vinte amostras de cada coordenada cartesiana (x, y, z).

Posteriormente essas cinco amostras são armazenadas em um vetor a fim de organizar os dados, para que em seguida seja calculada a média aritmética dos valores dos valores através da formula:

$$\overline{x} = \left(\frac{x_1 + x_2 + x_3 + x_4 + \ldots + x_n}{N} \right) \tag{4.20}$$

onde x_n são as amostras e N o total de amostras

Em seguida, o valor da média é utilizado para realizar o cálculo da variância dos dados, com o propósito de descobrir o quanto os valores recebidos pelo sensor se distanciam da média calculada. A variância é obtida através da formula:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2 \qquad (4.21)$$

onde $\overline{x}$ é a média, *n* as amostras e x_i a amostra atual.

Finalmente, os valores iniciais das amostras para cada *LED* são comparados com o valor da variância e assim, o valor que mais se aproxima da mesma é adotado como o novo valor de posicionamento espacial do *LED* em questão. Desta maneira, cada valor de cada coordenada tem sua posição reavaliada e substituída por um valor único que mais se aproxima da posição real do *LED* emissor, corrigindo eventuais trepidações e erros de cálculo. O Pseudocódigo 11 demonstra a implementação da classe *Filtro*, responsável por tomar amostras de leituras de dados da luva, calcular novos valores e os substituir, evitando movimentos indesejados.

4.2.16. Visualização

A *Classe de Visualização* é responsável por fornecer métodos secundários de exibição de todo o contexto tridimensional que o sistema envolve, provendo soluções alternativas de visualização de cena e de entidades tridimensionais, como visualização imersiva, Realidade Aumentada e visualização semi-imersiva, através de novos dispositivos não convencionais de entrada e saída de dados, tais como um *HMD* (*Head Mounted Display*), uma câmera de vídeo e um óculos de anaglifo, respectivamente..

Para que tal evento tenha efeito, primeiramente, a interface do sistema prepara a amostra tridimensional para se adequar à visualização pelo dispositivo, ajustando-o ao modelo 3-D para que o mesmo ocupe toda a extensão do objeto *Canvas3D*, maximizando a instância *viewport* e movendo o objeto *viewingplatform* para o local mais próximo possível da amostra. Tal atividade faz com que todos os espaços sejam aproveitados, contribuindo para a qualidade da visualização, uma vez que a resolução das telas do *HMD* são inferiores aos monitores *LCD* e *CRT* convencionais.

A API *Java3D* fornece em seu conjunto de classes, métodos essenciais para a programação de um *HMD*, dentre os quais: *setViewPolicy* e *int getViewPolicy*, derivados do pacote *javax.media.j3d.View* e responsáveis por definir e visualizar a política atual para a visão do usuário, sob o parâmetro, HMD_VIEW, o qual

especifica que o *Java3D* deve computar novos *viewpoints* usando a seqüência de transformação apropriada ao ambiente *Head Mounted Display*; *setScreenScalePolicy* e *getScreenScalePolicy*, utilizados para definir e visualizar a política de escala de visão. Tais métodos encapsulam toda a programação da estereoscopia, fornecendo todos os subsídios para a o funcionamento do dispositivo, de forma que a geração das imagens separadas por campo de visão fica a critério da API, bem como toda a métrica que envolve a geração de uma imagem binocular.

Para a geração de Realidade Aumentada, foram utilizados os algoritmos propostos por Kirner (Kirner, *et al* 2007), codificados sob a linguagem C e utilizando-se da biblioteca *ARToolkit*, onde a representação tridimensional é inserida no ambiente real de forma que a mesma pode ser explorada e analisada como se a entidade estivesse na mão no utilizador, utilizando-se de dispositivos simples e acessíveis.

Para a realização de tal evento de visualização, em um primeiro momento são impressos marcadores de papel com inscrições pré-definidas, as quais servem de ponto de referência no ambiente real para a inserção de modelos virtuais. Tais marcadores têm suas imagens capturadas por uma *Webcam* de 1.3*Mpixels* de resolução de foco.

Assim, através de algoritmos de visão computacional executados em tempo real, tais como o reconhecimento de bordas e limiarização de imagens, o desenho do papel é então identificado.

Posteriormente, tais desenhos são comparados a uma base de inscrições definidas pelo desenvolvedor, onde apenas as válidas prosseguem o curso do sistema, ou seja, se a imagem reconhecida for igual a alguma das imagens da base. Em seguida, o marcador que possuir tal desenho receberá um objeto virtual, também armazenado junto a uma base de modelos. Desta maneira, da mesma maneira que os modelos tridimensionais são manipulados com os dispositivos convencionais (teclado e mouse) e não convencionais (luva de dados), os mesmos podem ser transladados e rotacionados à medida que o usuário movimenta os marcadores, permitindo igual exploração das entidades reconstruídas.

Ao final do processo, o resultado pode ser visualizado tanto no monitor convencional, quanto no *HMD* (*Head Mounted Display*).

Já a visualização por Anaglifo constitui-se como o ultimo processo de visualização diferenciado. Tal processo caracteriza-se como semi-imersivo e de mais baixo custo, uma vez que proporciona uma sensação de profundidade do usuário na cena sintética utilizando-se apenas de um óculos estereoscópicos confeccionado com papel e plástico, em conjunto com um monitor convencional.

Para conseguir o efeito imersivo, o sistema prove ao usuário o mesmo modelo tridimensional visualizado sob dois diferentes ângulos, os quais representam o modelo visto pelo olho direito e esquerdo, renderizados concomitantemente no mesmo objeto *Canvas3D*. Posteriormente, as visões do olho esquerdo e direito são coloridas de vermelho e azul respectivamente, mesmas tonalidades das lentes de cada lado dos óculos. Desta maneira, através da lente vermelha, o usuário visualiza apenas o modelo que representa a visão do olho esquerdo e, através da lente azul, apenas o modelo visto pelo olho direito. O lado em que cada modelo se posiciona no objeto *Canvas3D* é independente, pois os óculos sempre permitirão visualizar apenas um modelo por lente, em acordo com a teoria sobre estereoscopia e paralaxe. Assim, ao fundir as imagens vistas através dos óculos, o cérebro humano as funde em uma única imagem acinzentada, dotada de profundidade mesmo que inerte na cena virtual. O Pseudocódigo 12 demonstra a implementação da visualização por Anaglifo.

4.2.17. *Loader*

A classe *Loader* configura-se como a responsável por identificar, carregar e renderizar os modelos tridimensionais sob o formato adotado pelo presente trabalho, tornando-se um elo de integração entre o algoritmo de reconstrução e o ambiente de Realidade Virtual, possibilitando a aplicação de todos os processos anteriormente descritos.

Com a API *Java3D* é possível importar pacotes de dados geométricos externos à programação nativa (cubos, cones, cilindros e esferas), como conjunto de pontos, arestas ou faces, modelados sob diversas ferramentas e/ou algoritmos computacionais, ou seja, trazer para dentro de um ambiente virtual Java, modelos desenvolvidos com ferramentas específicas, através de classes importadoras, as quais armazenam a cena, representada em código *Java3D*. Entre os tipos de arquivos que podem ser importados

estão *3D Studio* (.3ds), *Wavefront* (.obj), *VRML* (.wrl), AutoCAD (.dfx), dentre outros.

Inicialmente, para que as atividades desta fase possam ser iniciadas, torna-se necessária a utilização de métodos do pacote do *Visualization Toolkit* (vtk), descritos anteriormente, as quais são capazes de converter os arquivos que possuem as representações dos objetos tridimensionais, em formatos praticáveis em ambiente *Java3D*, possibilitando a implementação da classe importadora. Desta maneira, o objeto reconstruído tridimensionalmente e salvo no formato *Visualization Toolkit* (.vtk) (Kitware, 2007), é escrito sob o formato *Wavefront File Format* (.obj) (Wavefront, 2007), o qual representará a visualização dos modelos, cujas regiões são filtradas pela classe *Threshold*, utilizada para representar os componentes que formam o objeto reconstruído.

Posteriormente, com os arquivos já preparados, uma nova classe será implementada, derivada da classe *ObjectFile* (Sun, 2007), a qual provê subsídios para trazer todas as características do modelo ao ambiente Java, utilizando-se de métodos pré-implementados, portáveis para a reutilização. Esta nova classe será capaz de importar o modelo completo, reconstruído e exportado sob o formato *Wavefront*, para o ambiente virtual, mantendo todas as suas características originais, tais como aparência (*Appearance*), material (*Material*) e geometria (*Geometry*). Paralelamente, o objeto importado poderá ser tratado mediante todos os recursos de RV previamente descritos na seção anterior.

A escolha pelo sistema de representação de modelos em *Wavefront*, ocorreu justamente por tratar-se de um formato que representa modelagens com minuciosa obediência às exigências de performance de um sistema de RV. Também se destaca por armazenar seus atributos de aparência e material em um arquivo distinto, denominado *Material Library* (.mtl), de onde se encontram as informações de pontos, vértices e arestas, o que facilita seu uso e aplicação.

O processo de importação dos dados geométricos, aparência e material, ocorrem sob os passos descritos a seguir. Inicialmente, o arquivo *Wavefront*, arquivo texto por definição, são seguidos linha por linhas através de ponteiros e convertidos por um *parser* para a codificação na qual a linguagem Java possa interpretar. Posteriormente, todos os dados são mantidos em estruturas similares às estruturas *Java3D*, compondo os protótipos, a partir de nodos de uma estrutura típica do arquivo

de entrada, como nodos *Wavefront*. Estas estruturas são agora interpretadas por um gerenciador de Grafo de Cena, o qual realizará a estrutura hierárquica, provendo dados visualizáveis para o ambiente. A Figura 43 demonstra o fluxo dos dados no processo de importação de arquivos *Wavefront* para o ambiente de Realidade Virtual e o Pseudocódigo 13 demonstra a implementação da classe *Loader*.

```
Procedimento Filtro:
Begin
  Procedimento Atualiza Leds:
    Begin
      Enquanto Contador De Leds < Numero Total De Leds
        Begin
          Enquanto Amostra Atual < Numero Total De Amostras
            Begin
              Média De Amostras X[Contador De Leds][Contador De Amostras] =
Coordenada X[Contador De Leds][Contador De Amostras]/Numero De Amostras
              Média De Amostras Y[Contador De Leds][Contador De Amostras] =
Coordenada Y[Contador De Leds][Contador De Amostras]/Numero De Amostras
              Média De Amostras Z[Contador De Leds][Contador De Amostras] =
Coordenada Z[Contador De Leds][Contador De Amostras]/Numero De Amostras
              Somatório X [Contador De Leds][Contador De Amostras]=Somatório
X[Contador De Leds][Contador De Amostras]* (Coordenada X - Media De Amostras X
[Contador De Leds][Contador De Amostras])
              Somatório Y [Contador De Leds][Contador De Amostras]=Somatório
Y[Contador De Leds][Contador De Amostras]* (Coordenada Y - Media De Amostras Y
[Contador De Leds][Contador De Amostras])
              Somatório Z [Contador De Leds][Contador De Amostras]=Somatório
Z[Contador De Leds][Contador De Amostras]* (Coordenada Z - Media De Amostras Z
[Contador De Leds][Contador De Amostras])

              Variância X [Contador De Leds][Contador De Amostras] = 0.5* Somatório
X[Contador De Leds][Contador De Amostras];
              Variância Y [Contador De Leds][Contador De Amostras] = 0.5* Somatório
Y[Contador De Leds][Contador De Amostras];
              Variância Z [Contador De Leds][Contador De Amostras] = 0.5* Somatório
Z[Contador De Leds][Contador De Amostras];

              Valor Trocado De X [Contador De Leds][Contador De Amostras]= Menor
Diferença (Variância X[Contador De Leds][Contador De Amostras], Coordenada X[Contador
De Leds][Contador De Amostras])
              Valor Trocado De Y [Contador De Leds][Contador De Amostras]= Menor
Diferença (Variância Y[Contador De Leds][Contador De Amostras], Coordenada Y[Contador
De Leds][Contador De Amostras])
              Valor Trocado De Z [Contador De Leds][Contador De Amostras]= Menor
Diferença (Variância Z[Contador De Leds][Contador De Amostras], Coordenada Z[Contador
De Leds][Contador De Amostras])
              End;
            End;
        End;
End.
```

Novo Media De Amostras X;
Novo Media De Amostras Y;
Novo Media De Amostras Z;
Novo Somatório X;
Novo Somatório Y;
Novo Somatório Z;

Nova Variância X;
Nova Variância Y;
Nova Variância Z;

Novo Valor Trocado X;
Novo Valor Trocado Y;
Novo Valor Trocado Z;

Pseudocódigo 11 – Pseudocódigo que realiza o cálculo da variância e substituicao dos novos valores de leitura para os dados da luva

```
Procedimento Anaglifo
Begin
   Nova Cena;
   Cena = Loader.Carregar Modelo;
   Nova Transformação3D;
   Transformação3D.Translação(+Offset Olho, -Offset Olho,0)
   Novo Grupo De Transformação;
   Grupo De Transformação.Adicionar Transformação (Transformação3D)
   Nova Transformação3D;
   Transformação3D.Translação(-Offset Olho, +Offset Olho,0);
   Novo Grupo De Transformação;
   Grupo De Transformação.Adicionar Transformação (Transformação3D);
   Novo Agrupamento De Geometrias
   Agrupamento De Geometrias.Obter Geometrias Da Cena;

   Nova Cor Azul;
   Nova Cor Vermelha;
   Nova Aparência
   Aparência.Ajustar Material(Cor Azul);
   Nova Aparência;
   Aparência.Ajustar Material(Cor Vermelha);
   Novo Atributo De Transparência;
   Novo Coeficiente De Transparência;
   Coeficiente De Renderização.Suave;
   Atributo De Transparência.Ajustar Transparência(Coeficiente De Transparência);
      Aparência.Ajustar Atributos De Transparência(Atriubutos De Transparência);
      Aparência.Ajustar Atributos De Transparência(Atriubutos De Transparência);
   Novo Atributo De Renderização;
   Novo Coeficiente De Renderização.Desativar ZBuffer;
   Coeficiente De Renderização;
   Atributo De Renderização.Ajustar Atributo De Renderização(Coeficiente De
Renderizacao);

   Aparência.Ajustar Atributo De Renderização(Atributo De Renderização);
   Aparência.Ajustar Atributo De Renderização(Atributo De Renderização);
   Grupo De Transformação.Adicionar Filho(Agrupamento De Geometrias);
End.
```

Pseudocódigo 12 – Pseudocódigo para a implementação de visualização por Anaglifo

```
Procedimento Loader:
Begin
      Procedimento Carregar:
      Begin
         <----------------------
      End
End
```

```
Nova Cena:
Novo Objeto Padrão
Objeto Padrão.Ajustar Ao Canvas;
Cena.Importar(Modelo Tridimensional);
Novo Nodo de Agrupamento;
Nodo de Agrupamento:= Cena.Obter Grupo de Modelos;
Retornar Nodo de Agrupamento;
```

Pseudocódigo 13. Pseudocódigo da implementação da classe *Loader*

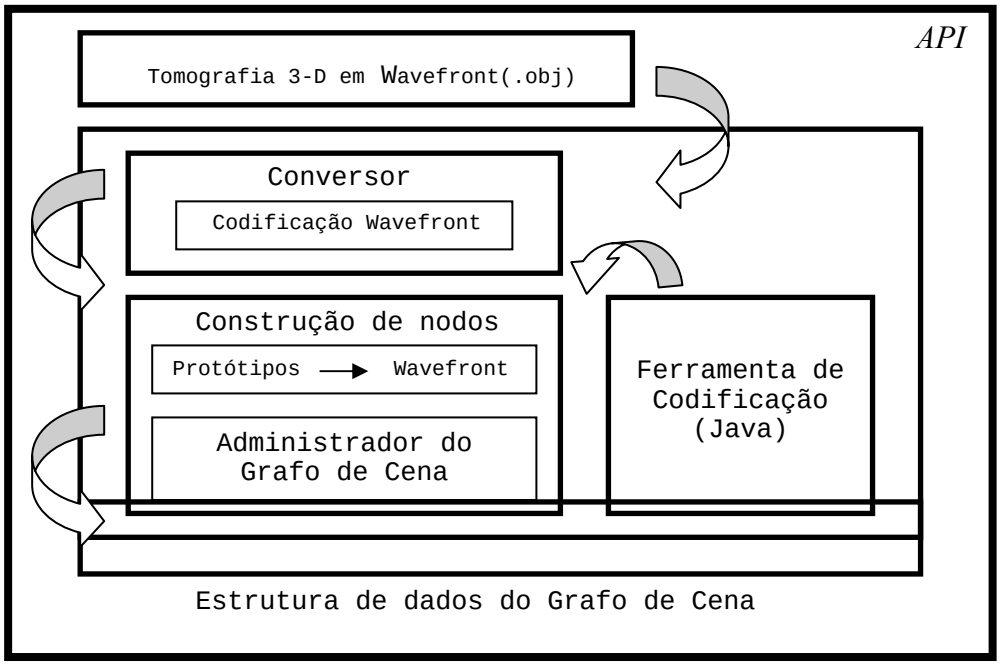


Figura 43 – Representação do fluxo dos dados durante o processo de importação de arquivos Wavefront na classe Loader

4.3. Dispositivos Não Convencionais Empregados

No presente trabalho, para a efetiva implementação dos recursos imersivos e interativos anteriormente descritos, fez-se necessário o uso de dispositivos não convencionais, responsáveis por caracterizar o sistema, juntamente com as classes programadas, um modelo de visualização e análise de RV.

Desta maneira, nesta seção são apresentados os dispositivos não convencionais empregados em cada classe, bem como suas especificações técnicas, os quais justificam sua utilização.

4.3.1. Dispositivos de Entrada de Dados:

4.3.1.1. Luva de Dados: *P5Glove*

Entre os dispositivos de entrada de dados empregados está a luva de dados *P5Glove* do fabricante *Essential Reality*. Tal dispositivo responde por ser o principal *hardware* responsável por realizar a implementação de todas as classes cujos principais objetivos é a manipulação de cenas e amostras tridimensionais, tais como:

Manipulação Não Convencional de Cena, *Manipulação Não Convencional de Modelo*, *Quatérnios*, *Filtro*, *Colisão Não Convencional* e *Extração de Atributos*.

A luva de dados *P5Glove* pesa 128 gramas e possui compatibilidade *mouse-like*, quando utilizada sem qualquer programação envolvida, apresentando a mesma interface (USB 1.1) de um *mouse* comum sob os sistemas operacionais *Windows/Linux/Macintosh*, o que lhe isenta do uso de cabos de energia.

A *P5Glove* também apresenta em sua estrutura, sensores de dobra localizados nos dedos, responsáveis por identificar os movimentos de clique, bem como ações de segurar uma amostra no ambiente sintético, a fim de prover a total exploração da mesma. Tais sensores podem ter seus parâmetros customizados através da *API Dualmode*, uma vez que o movimento de dobras dos dedos é variável para cada usuário.

Seu funcionamento baseia-se em um sistema de rastreamento óptico com dois receptores fotossensíveis em sua torre, os quais realizam a correlação dos raios de luz enviados pelos oito LED's emissores presentes na luva, para determinar a posição e orientação do dispositivo no espaço real de coordenadas, possibilitando a conversão para o espaço virtual. Tais LED's emissores encontram-se em sua totalidade na parte posterior da luva, o que se configura como uma das limitações do hardware, uma vez que são necessários no mínimo uma leitura de três LED's distintos para a identificação do correto posicionamento e orientação do dispositivo no espaço, visto que o usuário pode apontar a palma da mão utilizada para os receptores de luz. Tal limitação é combatida pela implementação da classe *Quatérnios*, a qual realiza a estimação de eixos de rotação intermediários, impedindo a falta de parâmetros das rotações de Euler e da falta de dados.

Outra limitação de hardware existente na P5Glove refere-se às leituras realizadas pela torre, no sentido de que os receptores fotossensíveis são suscetíveis a ruídos. Como o funcionamento do dispositivo baseia-se em emissão, recepção e correlação de raios infravermelhos, qualquer equipamento que opere dentro da faixa de freqüência de 300GHz e 300THz, pode interferir na leitura dos dados, causando uma inconsistência de posicionamento e orientação, fazendo com que os valores oscilem em uma margem de cinco unidades para mais ou para menos. Essa limitação justifica a implementação da Classe filtro, presente no sistema.

A recepção dos dados efetua-se sob uma taxa de atualização de 60 Hz e seis graus de liberdade, sendo:

- Três graus de Translação (*X/Y/Z)*, representados pelos movimentos de translação, relativos aos movimentos esquerda/direita, cima/baixo e frente/trás. Apresenta resolução e precisão de 0,003175 metros, além de alcance de 1,2192 metros do receptor e alcance ilimitado do monitor.

- Três graus de Rotação (*yaw/pitch/row*), representados pelos movimentos de rotação dos eixos *x, y* e *z* de coordenadas cartesianas. Apresenta resolução e precisão de 1º.A Figura 44 apresenta a luva de dados P5Glove utilizada no presente trabalho.

Figura 44 – Luva de dados P5Glove da EssentialReality

4.3.1.2. *Webcam* Clone

Outro dispositivo não convencional utilizado para a entrada de dados no sistema foi uma *webcam* do fabricante Clone, modelo 11090. Tal dispositivo configura-se no trabalho como o *hardware* responsável por realizar a captura dos dados a serem processados pelo módulo de visualização através de técnicas de Realidade Aumentada, presente na classe Visualização.

Este dispositivo suporta uma resolução máxima de 1,3 megapixels, capaz de capturar imagens a 1280x960 *pixels*, suficiente para identificar corretamente os marcadores, impedindo com que as bordas dos mesmos sejam confundidas com artefatos ou *pixels* interpolados da imagem. A *webcam* é alimentada por porta USB. Possui brilho automático e foco de 30mm ~ infinito, o que também garantem a confiabilidade ao se capturar as imagens dos marcadores. A Figura 45 apresenta a *webcam* Clone utilizada no presente trabalho.

Figura 45 – Webcam Clone 11090

4.3.2. Dispositivo de Saída de Dados

4.3.2.1. *Head Mounted Display* Innovatek

O dispositivo não convencional de saída de dados empregado foi um *HMD* (*head mounted display*) do fabricante *Innovatek* e responde por ser o responsável em produzir o efeito imersivo de visualização ao utilizar o sistema, baseando-se nos princípios de estereoscopia, descritos também pela classe *Visualização*.

A imagem é fornecida pelo dispositivo através de duas telas de LCD de 0.6cm x 0.6cm cada, as quais apresentam sinal de entrada de vídeo NTSC/PAL colorido e 180.000 *pixels* (800x225) de resolução espacial, essencial para uma boa qualidade de análise, o que assegura a ausência do efeito de *Aliasing*[5].

O dispositivo também apresenta ângulo de visão diagonal de 26º, o que assegura a independência ótica do usuário, uma vez que lhe permite visualizar diferentes pontos da tela, mantendo o efeito estereoscópico, o qual simula, além de profundidade das amostras visualizadas, um tamanho de imagem de uma tela de 86,4 cm a 2 metros de distância.

Tal dispositivo pesa 200 gramas, favorecendo a utilização do mesmo por um grande período de tempo. Depende de alimentação de 4,5VDC a 12,0VDC estável. Possui também saída do adaptador de energia de 9,0V e bateria de lítio: LP900 de 7,2V/900mAh, sob consumo de 1,0W.

Finalmente, o *HMD* também apresenta terminal de entrada de áudio/video quadrupolo de $\varnothing$3.5mm, imagem de 1 Vértice por pixel sob resistência de 72Ω (sem

[5] *Aliasing* (discontinuidade): Efeito em imagem proveniente de baixa amostragem de freqüência (taxa de Nyquist), o que provoca o efeito de escada nas bordas das mesmas.

movimento), som de 200mVrms sob resistência de 20KΩ e terminal de energia com distância de ∅4.0mm. A Figura 46 apresenta o *HMD Innovatek* utilizado no presente trabalho.

Figura 46 – Head Mounted Display Innovatek V-490

No Capitulo V são apresentados os resultados da implementação de cada uma das classes descritas, bem como um estudo de caso para aplicações em Ciência de Solos, onde é abordado o tema da porosidade de amostras agrícolas, envolvendo a formação de caminhos preferenciais de fluxo de água e solutos.

Capítulo V
Resultados e Conclusões

5.1. Reconstrução

Para a avaliação dos resultados descritos nas seções deste capítulo, um conjunto de diversas amostras foi utilizado. Tal conjunto constitui-se de: porções de latosolo degradado, submetidos a um mesmo processo de aquisição tomográfico sob 58 KeV de energia, 15mm de translação total, 0,083mm de passo linear, 180º de rotação total, 1º de passo angular e 4 segundos de tempo de contagem por amostra de projeção; argila, sob 56 KeV de energia, 1mm de translação total, 0,0611 de passo linear, 180º de rotação total, 1º de passo angular e 7 segundos de tempo de contagem por amostra de projeção; adubo orgânico, sob 58,5 KeV de energia, 15mm de translação total, 0,0833mm de passo linear, 180º de rotação total, 1º de passo angular e 4 segundos de tempo de contagem por amostra de projeção; areia, sob 56 KeV de energia, 0,0486mm de passo linear, 180º de rotação total, 1,2º de passo angular e 10 segundos de tempo de contagem por amostra de projeção; vidro, sob 56 KeV de energia, 4mm de translação total, 0,040 de passo linear, 180º de rotação total, 1,8º de passo angular e 14 segundos de tempo de contagem por amostra de projeção; e cimento, sob 56 KeV, 10mm de translação total, 0,0556 de passo linear, 180º de rotação total, 1º de passo angular e 8 segundos de tempo de contagem por amostra de projeção.

Considerando a classe de *Reconstrução*, os algoritmos descritos por Pereira e Cruvinel (Pereira e Cruvinel, 2001) foram reutilizados, os quais foram escritos sob a linguagem de programação C++. Assim a partir das projeções tomográficas obtidas no minitomógrafo, dados bidimensionais foram reconstruídos por Retroprojeção Filtrada, entretanto, com presença de algum ruído, amplificado por um filtro rampa, presente no algoritmo descrito. A Figura 47 apresenta as imagens obtidas com o uso do algoritmos de reconstrução bidimensional de imagens tomográficas, aplicado sob os diversos tipos de amostras ensaiadas onde: (a), (b), (c) e (d) são imagens de amostras de solo degradado; (e) imagem de um torrão de latosolo; (f) e (g) são imagens de amostras de argila; (h) imagem de uma amostra de areia; (i) imagem de uma amostra de adubo; (j) e (k) são imagens de amostras de cimento, e (l) imagem de uma amostra composta por esferas de vidro.

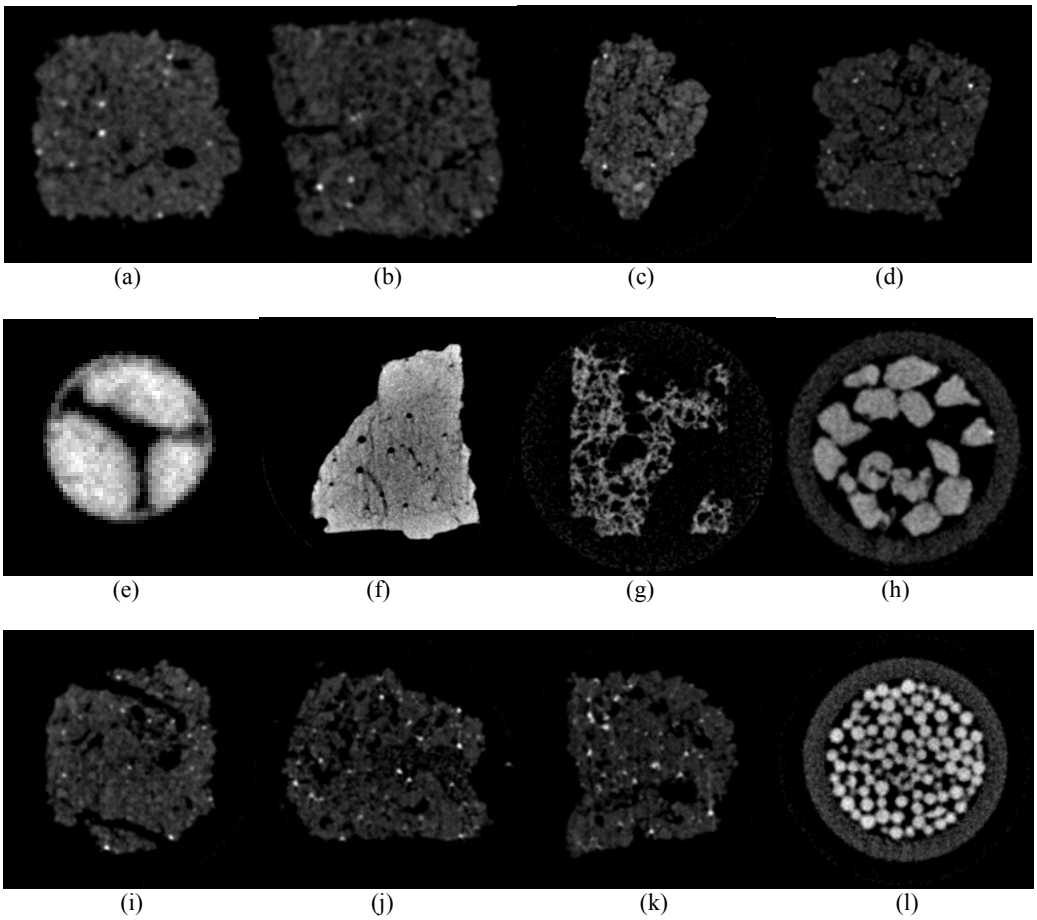


Figura 47 – Imagens das amostras reconstruídas em 2-D com o algoritmo de Retroprojeção Filtrada e adotadas para o desenvolvimento do presente trabalho: solo degradado (a, b, c, d), latosolo (e), argila (f, g), areia (h), adubo (i), cimento (j, k) e vidro (l).

Após a reconstrução das imagens, de forma a se obter melhoria, aplicou-se uma filtragem bidimensional, utilizando-se da *Transformada Wavelet Daubechies* com uma janela de 72 coeficientes. A nova filtragem bidimensional demonstrou bons resultados quando combinada com a única filtragem *a priori* por meio de janelas de *Hamming*, utilizada na implementação original, conforme se pode observar na Figura 48 para as mesmas imagens processadas.

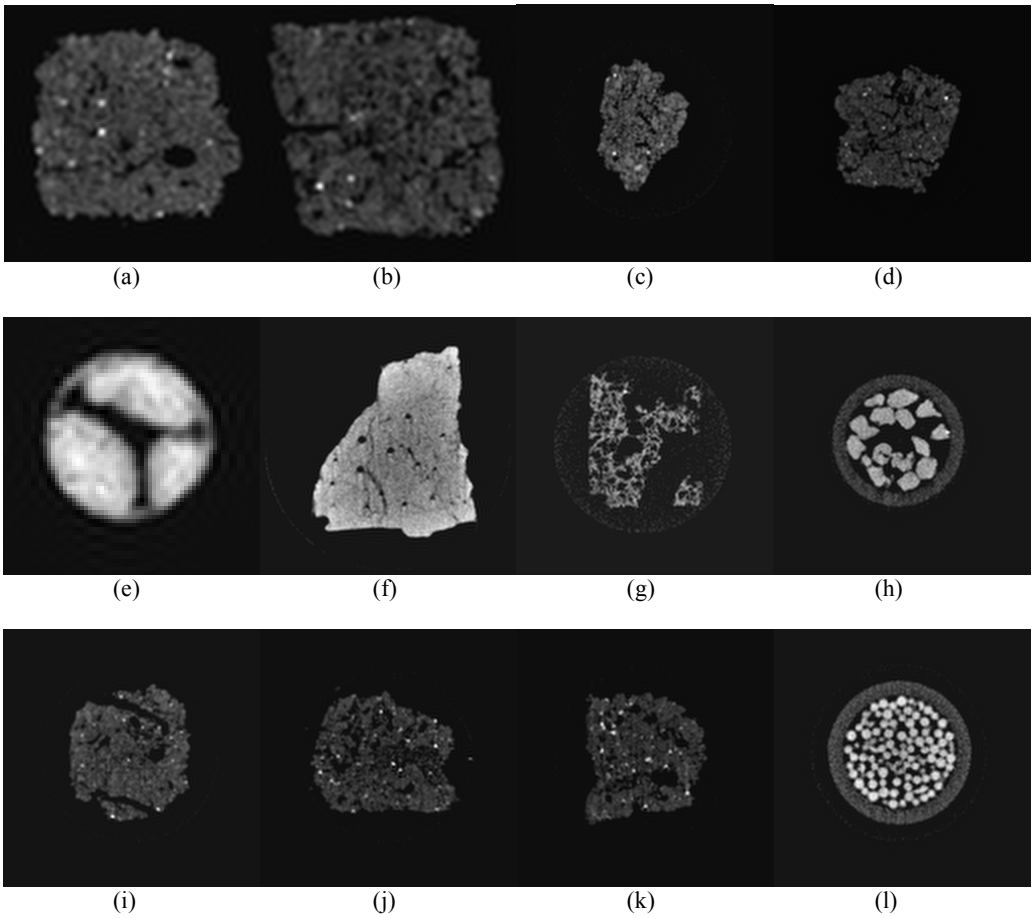
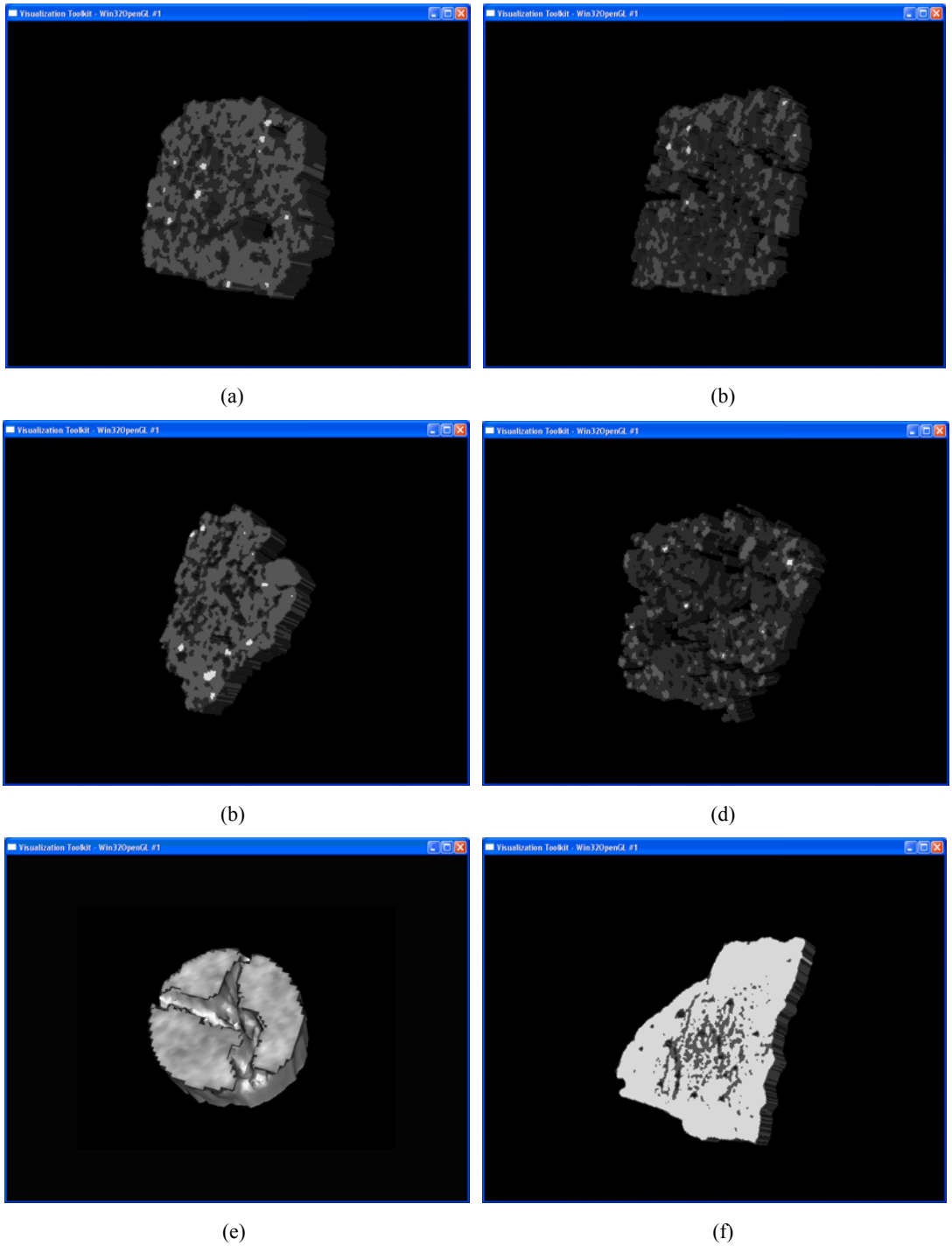


Figura 48 – Imagens filtradas com o algoritmo adaptativo através da Transformada Wavelet Daubechies com uma janela de 72 coeficientes: solo degradado (a, b, c, d), latosolo (e), argila (f, g), areia (h), adubo (i), cimento (j, k) e vidro (l).

Pode-se observar que as imagens tomográficas obtidas com o algoritmo de retroprojeção apresentaram-se com elevado ruído. Já as imagens filtradas com coeficientes *Daubechies*, foram suavizadas nas porções ruidosas e ainda assim preservaram os detalhes da imagem reconstruída pela retroprojeção.

Já a fase tridimensional da reconstrução forneceu modelos sintéticos fiéis aos modelos reais, convertidos sob o formato *Wavefront* e de passível aproveitamento

a todas as classes descritas no capitulo anterior, uma vez que carrega consigo todas as propriedades físicas originais, tais como formas, coeficientes de atenuação e volume, conforme ilustrado nas Figura 49 e 50.

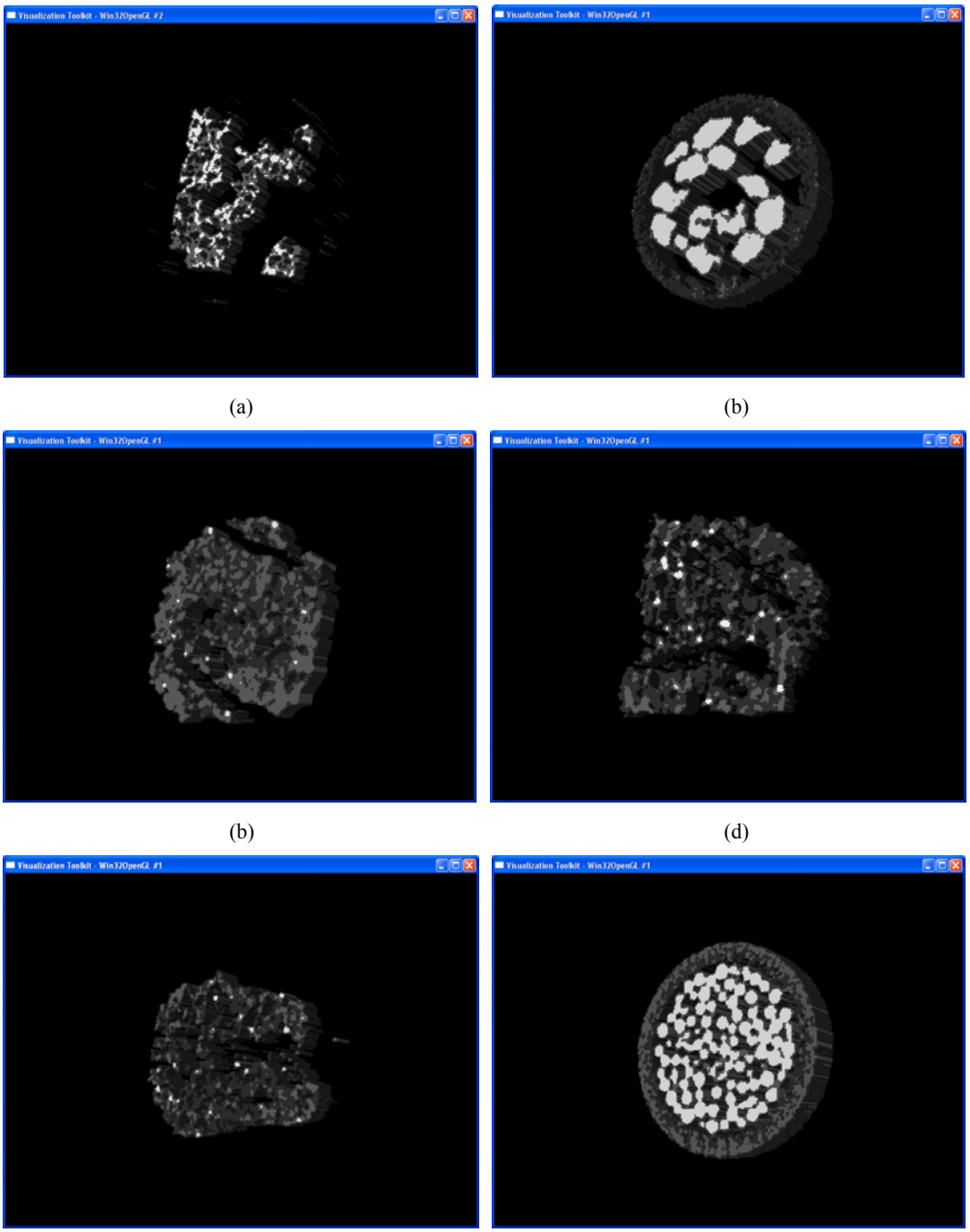


(a) (b)

(b) (d)

(e) (f)

Figura 49 – Imagens das amostras reconstruídas em 2-D com o algoritmo de Retroprojeção Filtrada e adotadas para o desenvolvimento do presente trabalho: solo degradado (a, b, c, d), latosolo (e) e argila (f).



(a) (b)

(b) (d)

(e) (f)

Figura 50 – Imagens das amostras reconstruídas em 3-D com o algoritmo de *B-Wavelets* e adotadas para o desenvolvimento do presente trabalho: argila (a), areia (b), adubo (c), cimento (d, e) e vidro (f).

5.2. Ambiente de Realidade Virtual

A interface que representa a integração de todas as ferramentas de Realidade Virtual, como anteriormente descritas, foi desenvolvida para possibilitar o acesso a todos os recursos necessários para um sistema de RV, utilizando a linguagem de

programação *Java*, sob sua *API Java3D*. Assim, toda estrutura do Grafo de Cena foi programada, de acordo com as classes descritas neste trabalho, suportadas pela *API*, resultando em uma *interface* planejada, intuitiva e de fácil usabilidade, de forma a se obter melhor acomodação do usuário com a mesma . A Figura 51 ilustra a *interface* e seus respectivos controles, acessos e janelas de exibição de resultados.

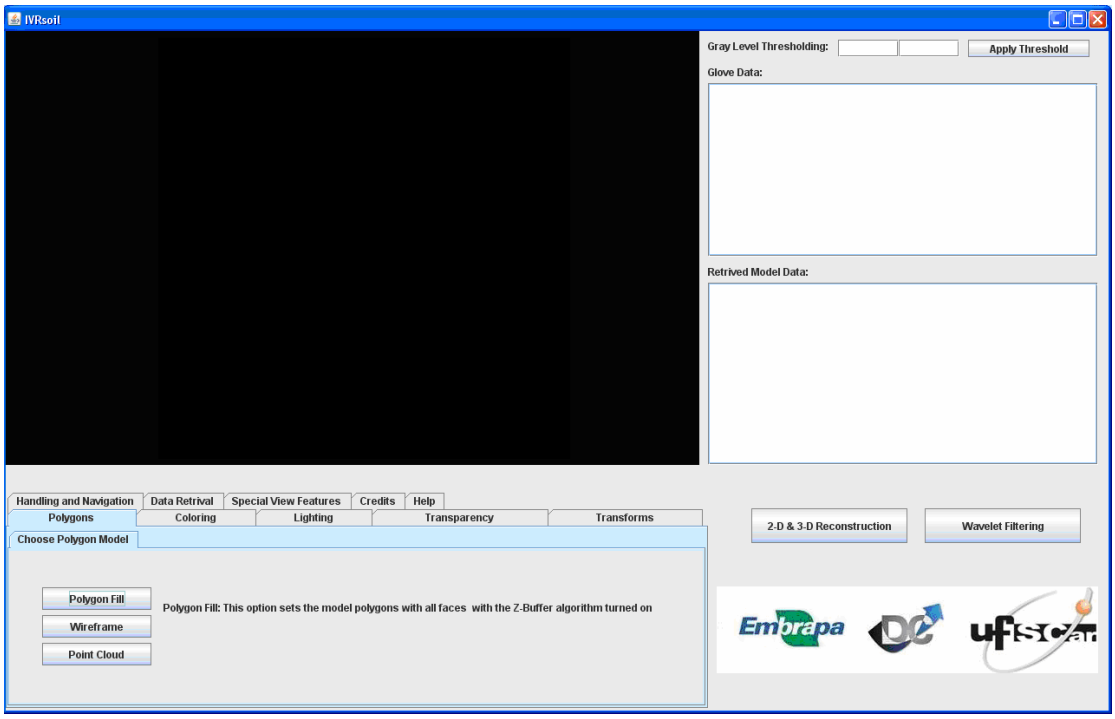


Figura 51 - Representação da interface de RV, onde são apresentados os acessos às opções de renderização, transformações físicas lineares, colorização, transparência, iluminação e modos de visualização.

A *interface* fornece, em sua disposição inicial, acessos aos diferentes módulos que compõem o ambiente de Realidade Virtual, formados por todas as classes da Figura 37. Assim, através das abas da interface, o usuário pode escolher entre os modos de *Polígonos*, *Iluminação*, *Renderização*, *Manipulação*, *Visualização*, *Colorização*, *Transparência* e *Transformações*, cada qual com seus respectivos parâmetros.

Também dispostos na *interface* encontram-se o controle de *Threshold* e as janelas de visualização de dados da luva p5Glove e dos dados obtidos pela extração de atributos. Não menos importante, encontram-se na mesma janela, atalhos para a chamada das classes de Reconstrução 2-D, filtragem por Wavelets e para a Reconstrução 3-D, uma vez que foram constituídas por intermédio de outra linguagem

de programação (*C++*), influindo tal diferença apenas no modo com que as classes são chamadas.

Ao centro da interface encontra-se o *Canvas3D* e sua respectiva *Viewport*, onde todo o contexto tridimensional se insere, desde a visualização até a manipulação e análise dos processos.

5.3. Transformações Físicas Lineares

Considerando as *Transformações Físicas Lineares*, os resultados das implementações demonstraram-se visualmente e matematicamente satisfatórios, uma vez que as operações com matriz de transformação forneceram operações exatas de rotação, translação e escala utilizando-se os parâmetros de entrada fornecidos pelo usuário. Como resultados, são apresentadas as matrizes de transformação para uma translação de duas unidades sobre o eixo *x*, uma rotação de 30º sobre *y*, uma escala de duas unidades em todos os eixos, espelhamento sobre o eixo *x* e alongamento sobre o eixo *x*, além de seus respectivos resultados visuais, antes e depois das transformações, aplicados sobre uma amostra de solo agrícola, como é ilustrado nas Figuras 52, 53, 54, 55 e 56.

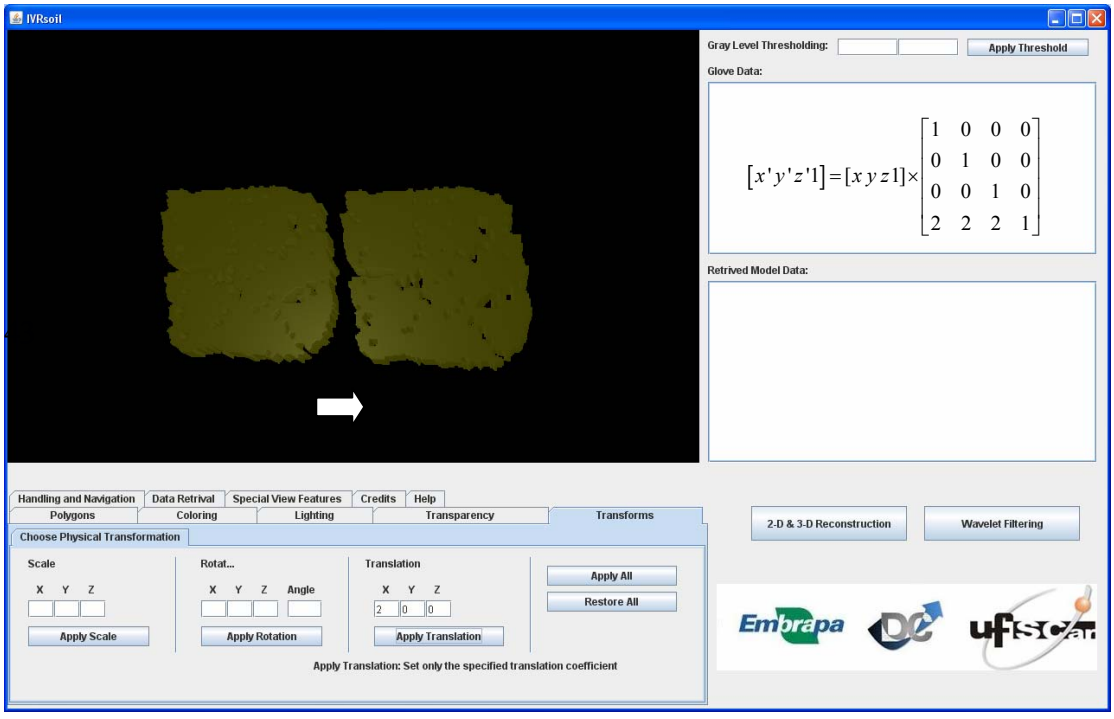


Figura 52 – Resultado da aplicação de translação como transformação física sob os parâmetros: x=2, y=0 e z=0.

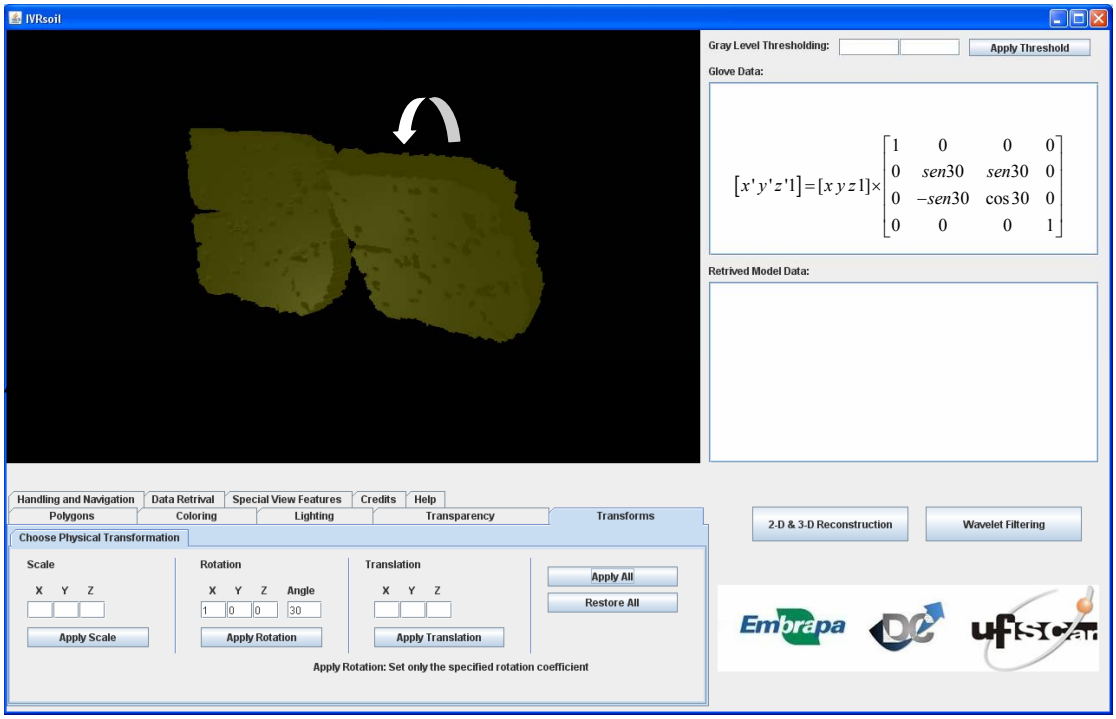


Figura 53 – Resultado da aplicação de rotação como transformação física sob os parâmetros: x=1, y=0, z=0 e ângulo=30º.

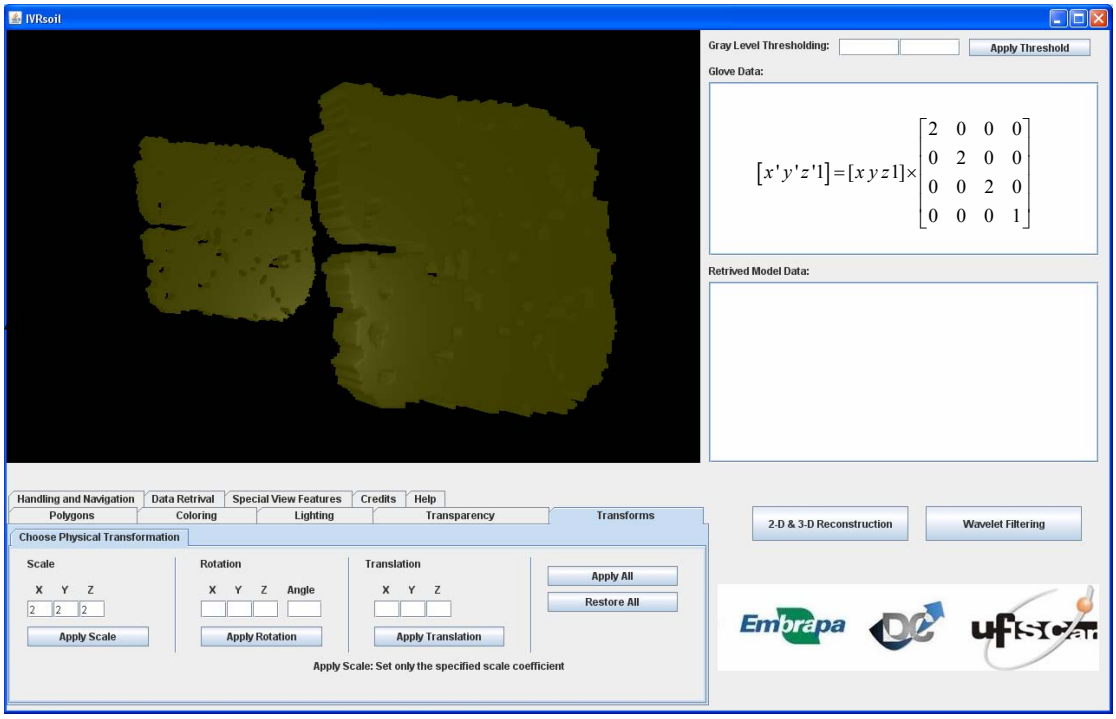


Figura 54 – Resultado da aplicação de escala como resultado de transformação física sob os parâmetros: x=2, y=2 e z=2.

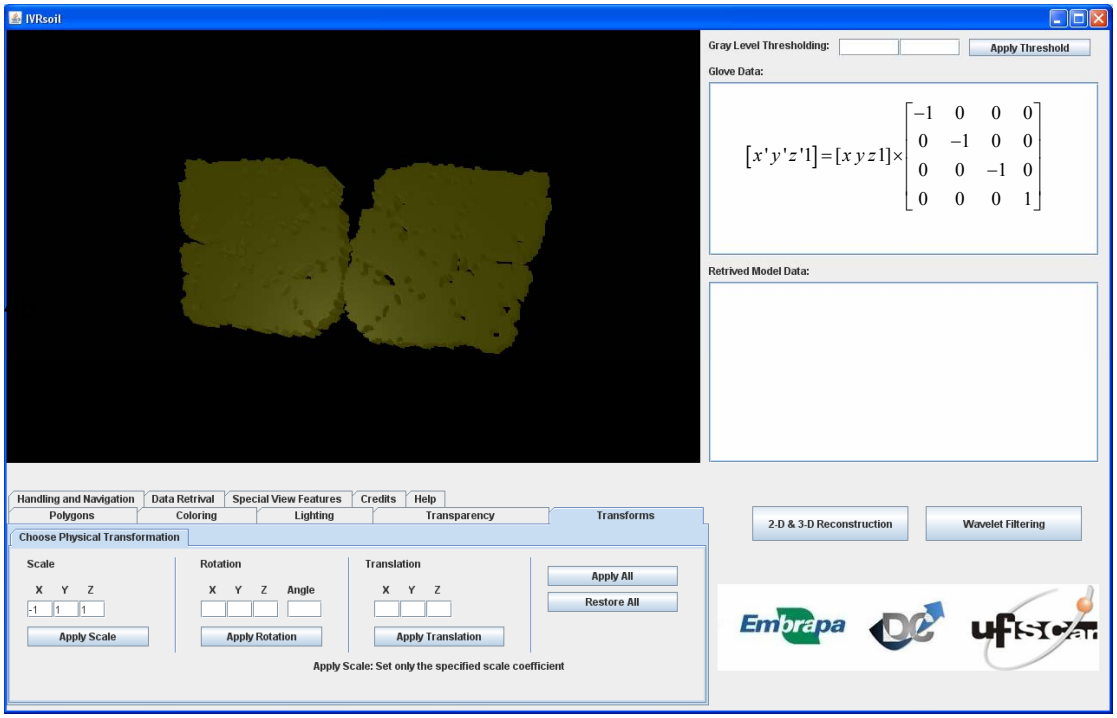


Figura 55 – Resultado da aplicação de espelhamento como transformação física, através de operação de escala, sob os parâmetros: x=-1, y=1 e z=1.

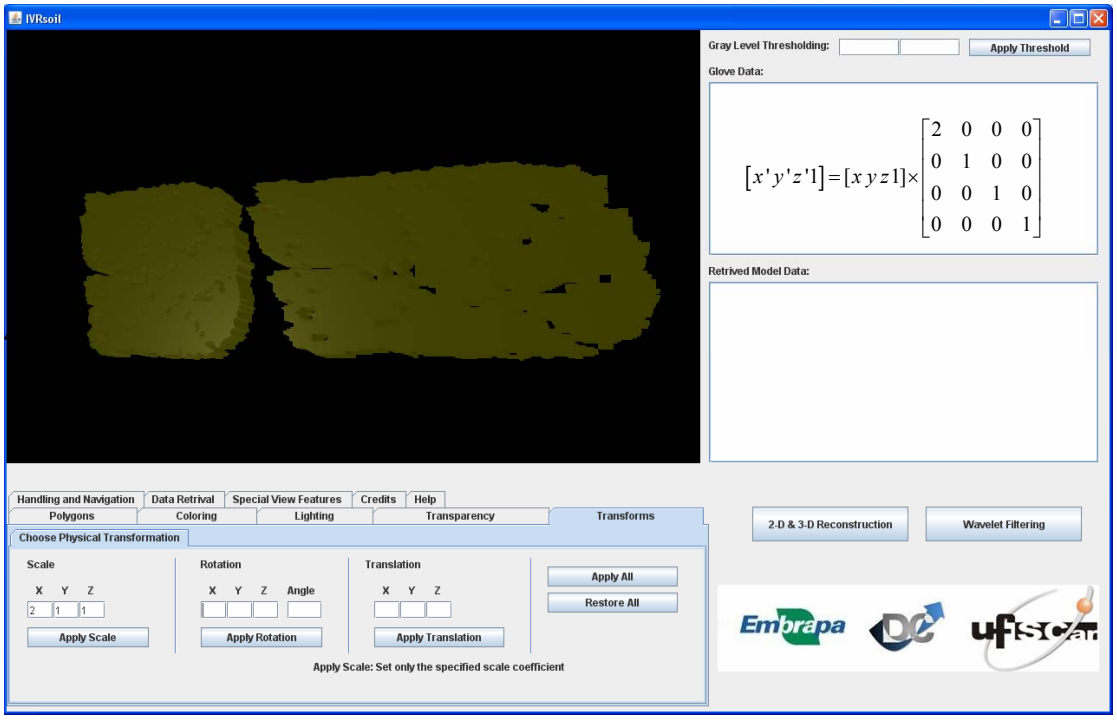


Figura 56 – Resultado da aplicação de alongamento como transformação linear através de operação de escala, sob os parâmetros: x=2, y=1 e z=1.

5.4. Transparência

A respeito da classe *Transparência*, a mesma retornou resultados visuais satisfatórios, de forma que todos os vértices, arestas e planos obtiveram seus níveis de opacidade alterados em tempo de execução, permitindo visualizar superfícies interiores outrora oclusas por camadas mais externas. Uma aplicação de interesse procura visualizar e identificar um canal poroso dentro da amostra, sem a retirada das superfícies que os encobrem. As Figuras 57, 58 e 59 representam os resultados da aplicação dos coeficientes de transparência sobre uma imagem de amostra de adubo, sob os parâmetros 0,2, 0,5 e 0,9, nomeados Opaco, Suave, Translúcido e Transparente, onde os menores valores de parâmetros representam os maiores níveis de opacidade e os maiores valores de parâmetros representam, respectivamente, maior grau de transparência. As Figuras 60, 61, 62, 63, 64 e 65 apresentam os resultados para os mesmos coeficientes aplicados sobre duas imagens distintas de amostras de argila.

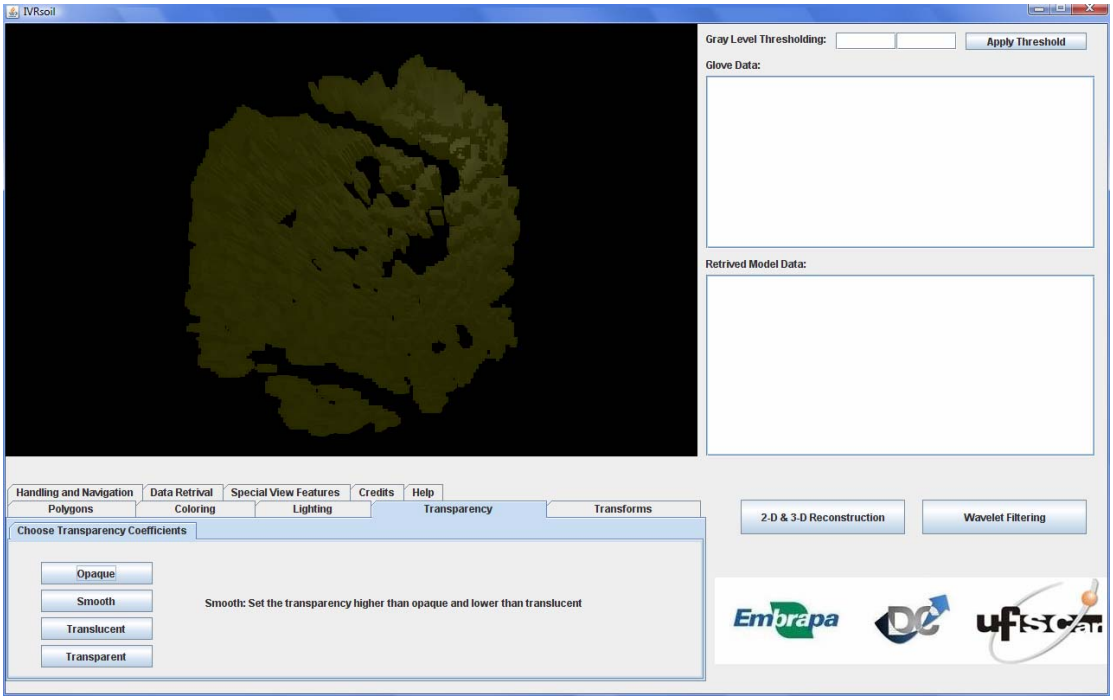


Figura 57 – Resultado da aplicação de Transparência na amostra de adubo sob o coeficiente Suave (0,2), produzindo alto grau de opacidade.

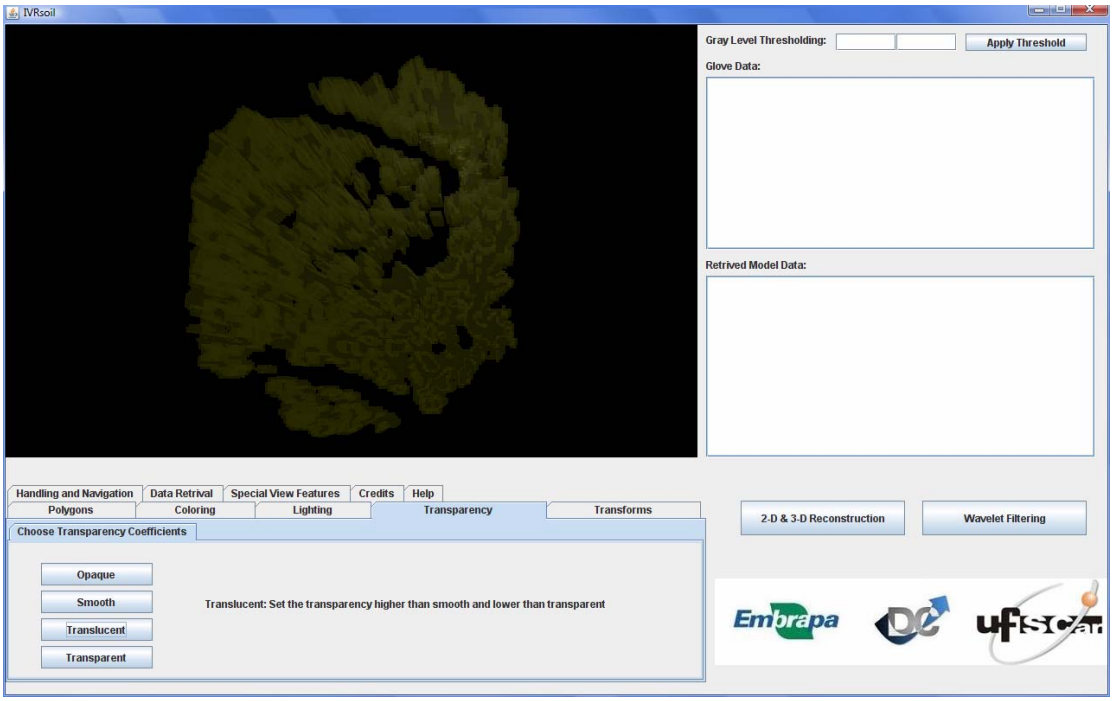


Figura 58 – Resultado da aplicação de Transparência na amostra de adubo sob o coeficiente Translúcido (0,5), produzindo efeito intermediário de opacidade.

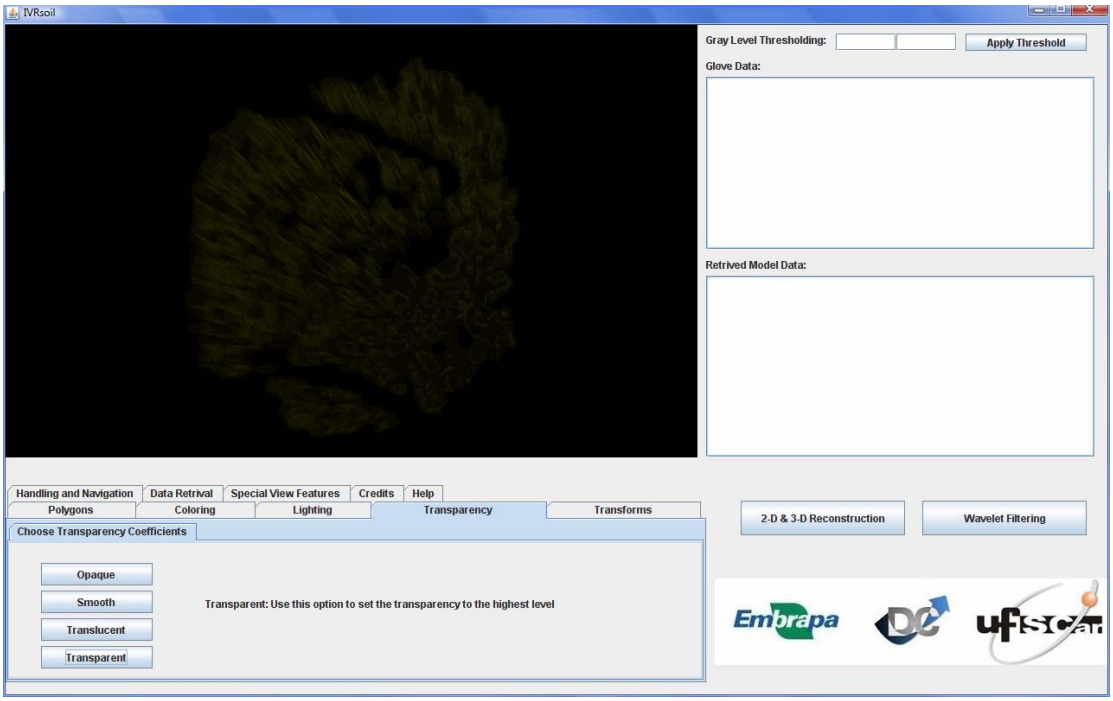


Figura 59 – Resultado da aplicação de Transparência na amostra de adubo sob o coeficiente Transparente (0,9), produzindo efeito mínimo de opacidade.

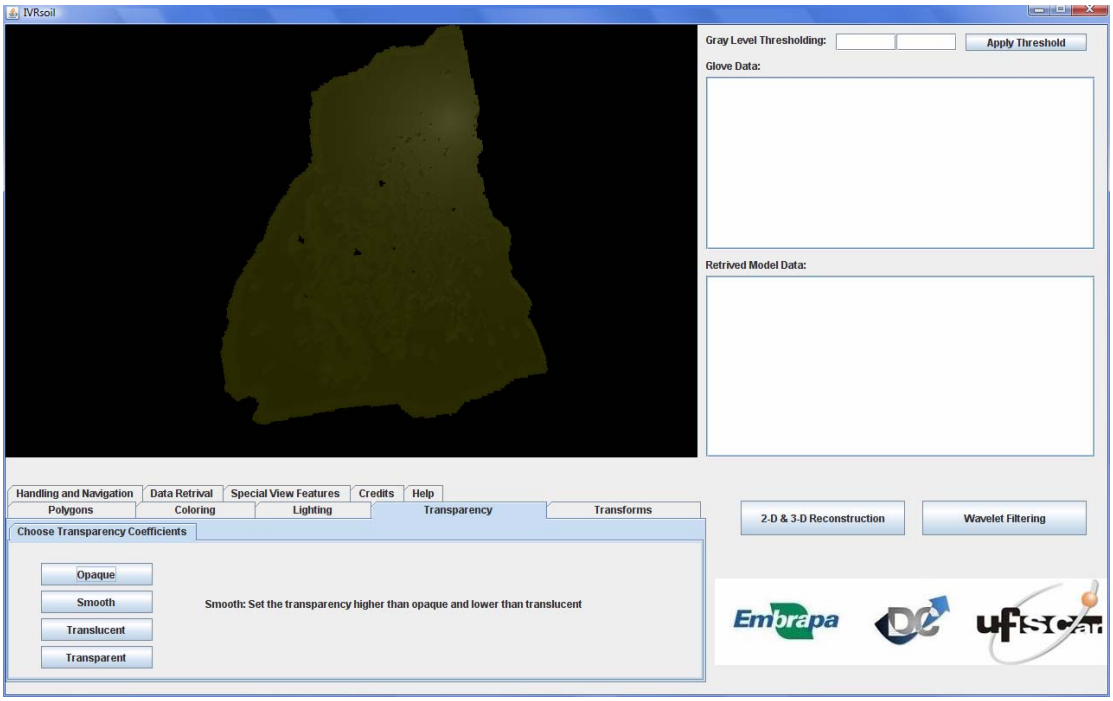


Figura 60 – Resultado da aplicação de Transparência na amostra de argila sob o coeficiente Suave (0,2), produzindo alto grau de opacidade.

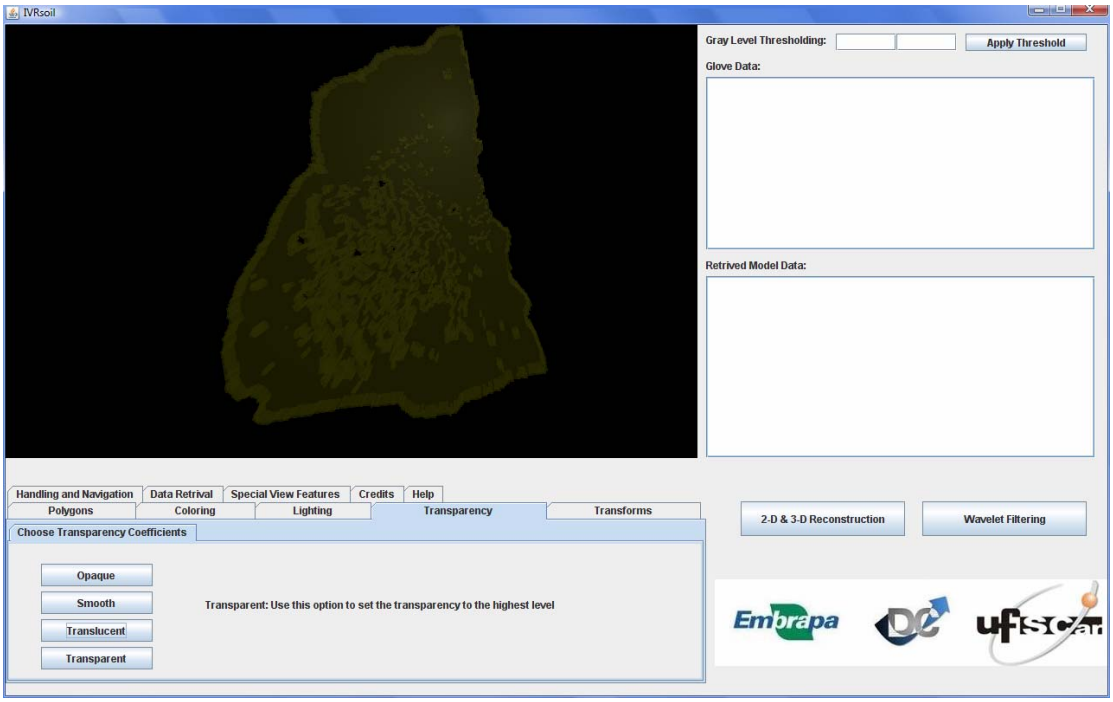


Figura 61 – Resultado da aplicação de Transparência na amostra de argila sob o coeficiente Translúcido (0,5), produzindo efeito intermediário de opacidade.

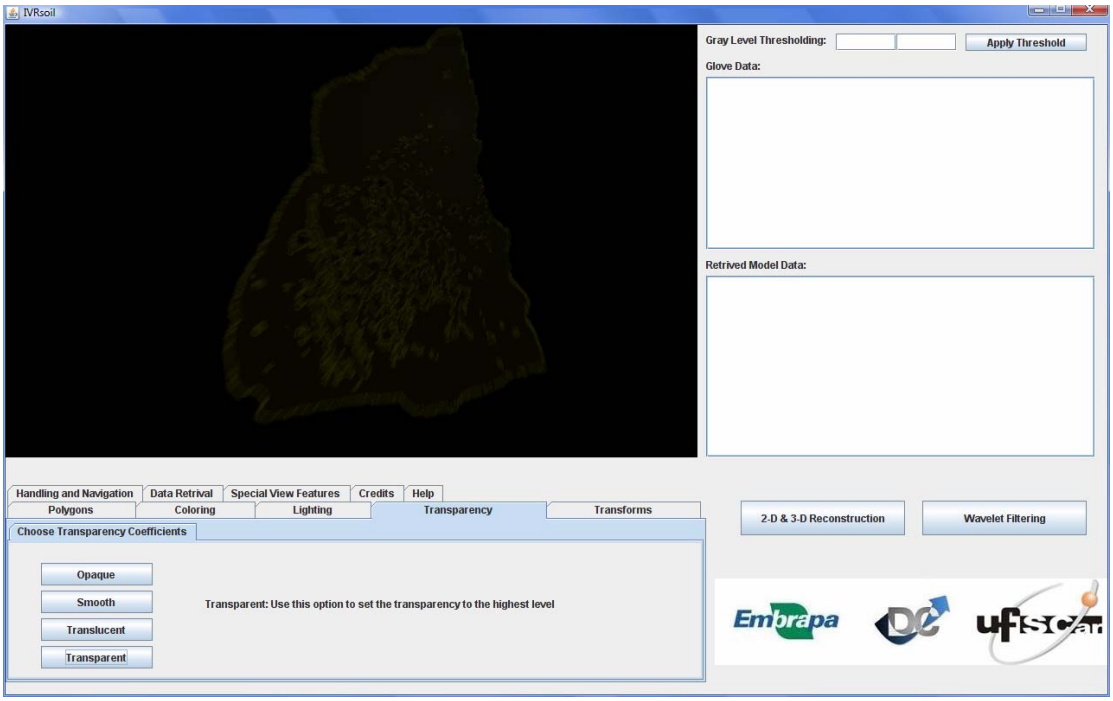


Figura 62 – Resultado da aplicação de Transparência na amostra de argila sob o coeficiente Transparente (0,9), produzindo efeito mínimo de opacidade.

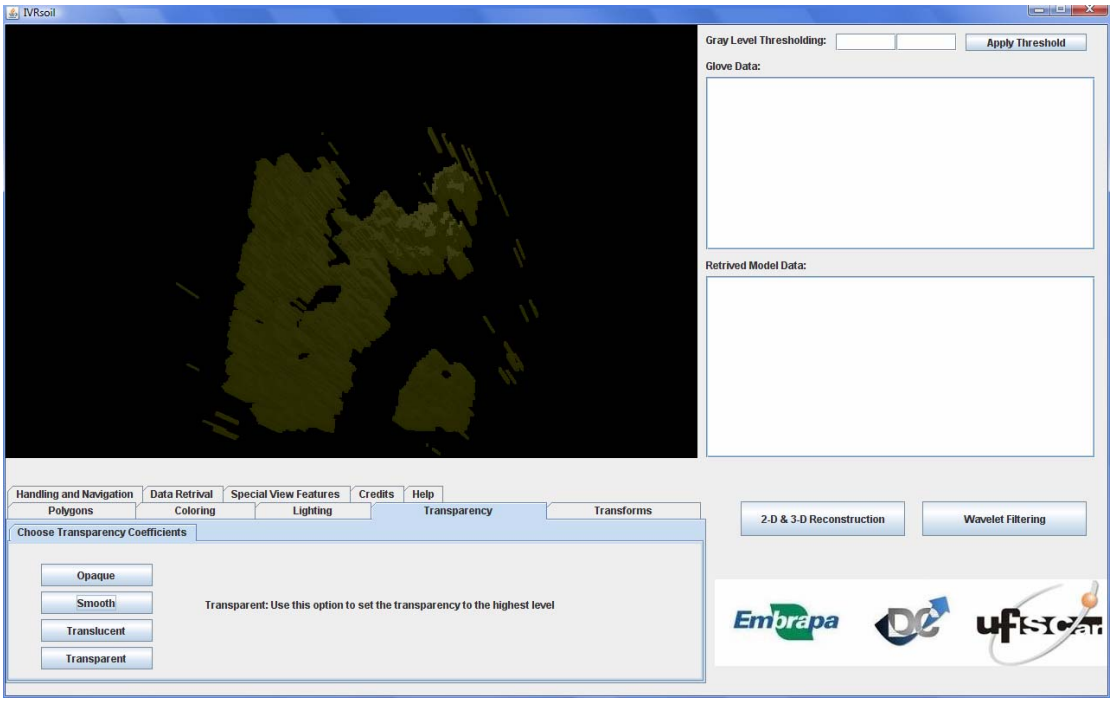


Figura 63 – Resultado da aplicação de Transparência na amostra de argila sob o coeficiente Suave (0,2), produzindo alto grau de opacidade.

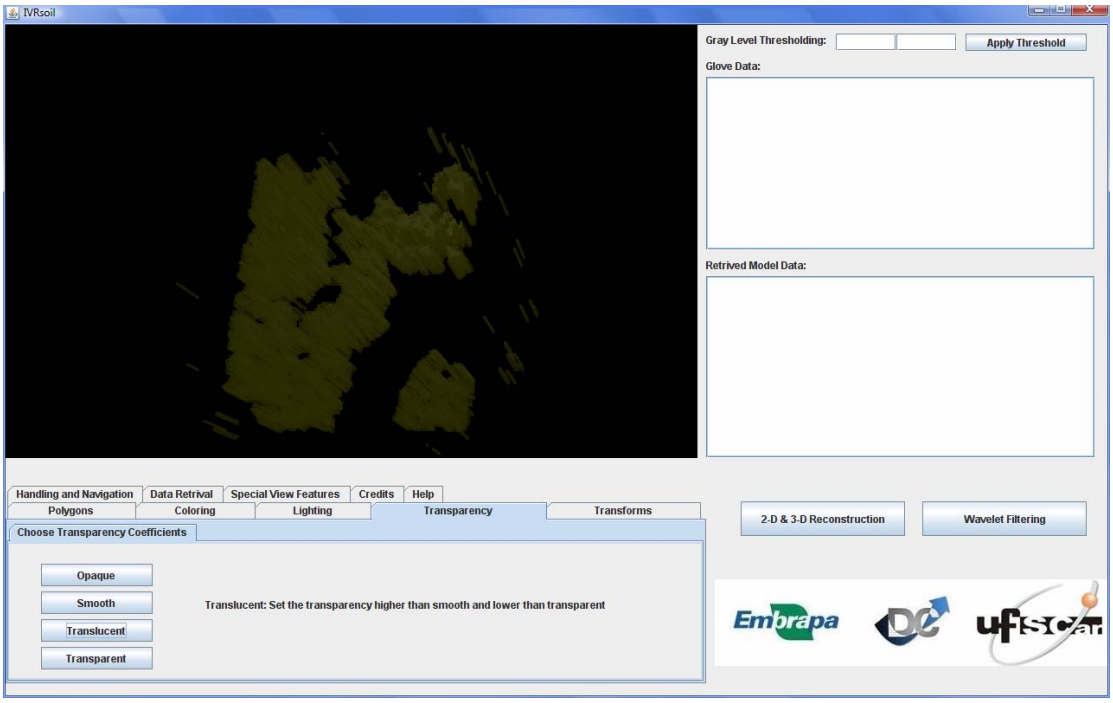


Figura 64 – Resultado da aplicação de Transparência na amostra de argila sob o coeficiente Translúcido (0,5), produzindo efeito intermediário de opacidade.

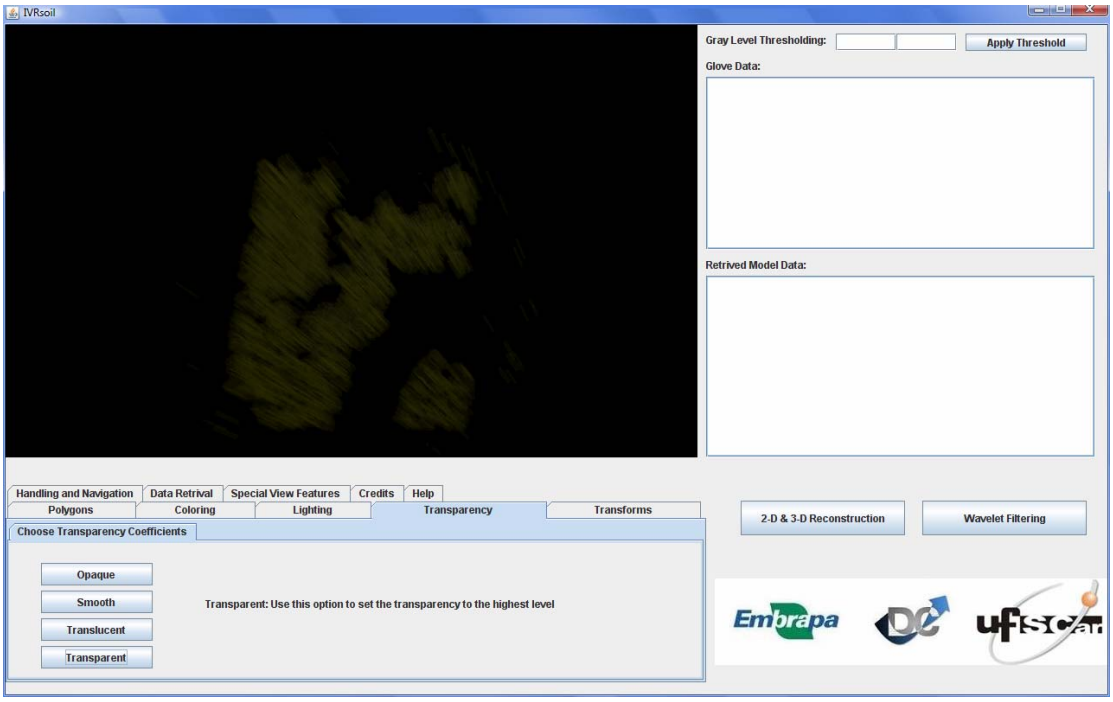


Figura 65 – Resultado da aplicação de Transparência na amostra de argila sob o coeficiente Transparente (0,9), produzindo efeito mínimo de opacidade.

5.5. Polígonos

A classe de controle de *Polígonos*, através da efetiva separação entre pontos, arestas e planos, permitiu a visualização individual de quaisquer partes das amostras reconstruídas, especialmente os pontos interpolados pelo algoritmo de reconstrução, especificamente pela técnica de *B-Splines*, a qual constrói novos pontos no modelo 3-D à medida que forem necessários. Com a simples visualização de planos, tais pontos estimados não podem ser identificados, uma vez que se fundem às faces da geometria. As Figuras 66, 67, 68, 69, 70 e 71 ilustram os resultados da aplicação da classe *Polígonos*, mediante os coeficientes de Pontos (POLYGON_POINTS), Arestas (POLYGON_LINES) e Faces (POLYGON_FACES), respectivamente, sobre duas imagens distintas de amostras de cimento. As Figuras 72, 73 e 74 apresentam os resultados da classe *Polígonos* sobre uma imagem de amostra de solo degradado sob os mesmos coeficientes.

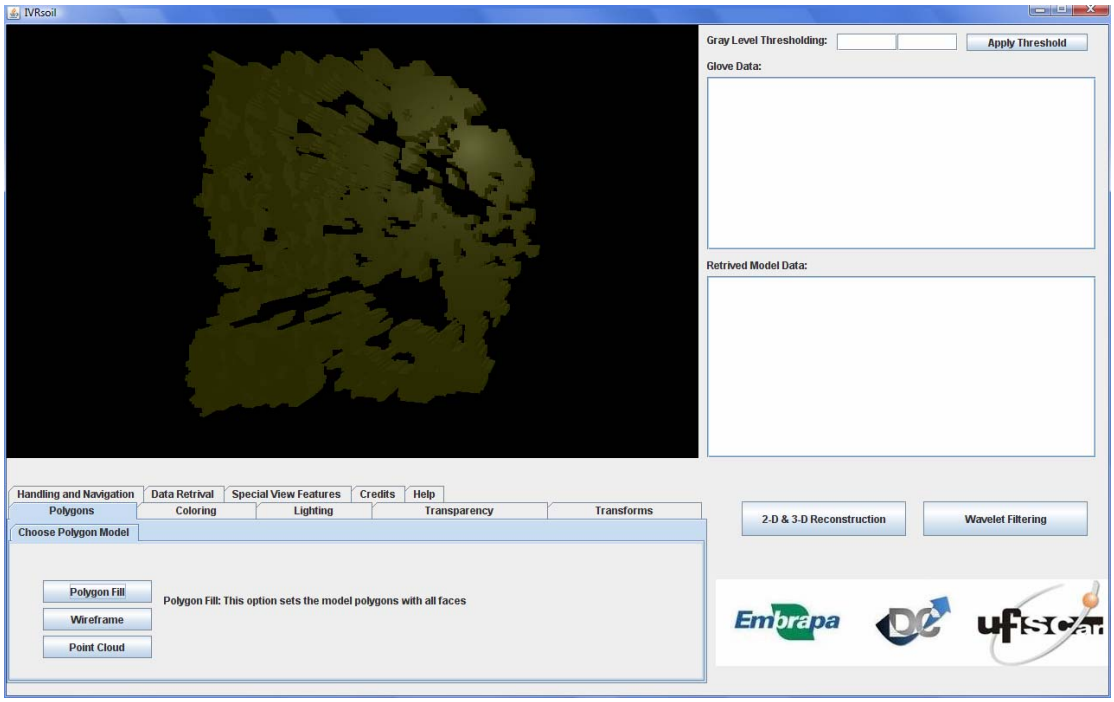


Figura 66 – Resultado da aplicação da classe Polígonos sobre uma amostra de cimento sob exibição de faces (coeficiente POLYGON_FACES).

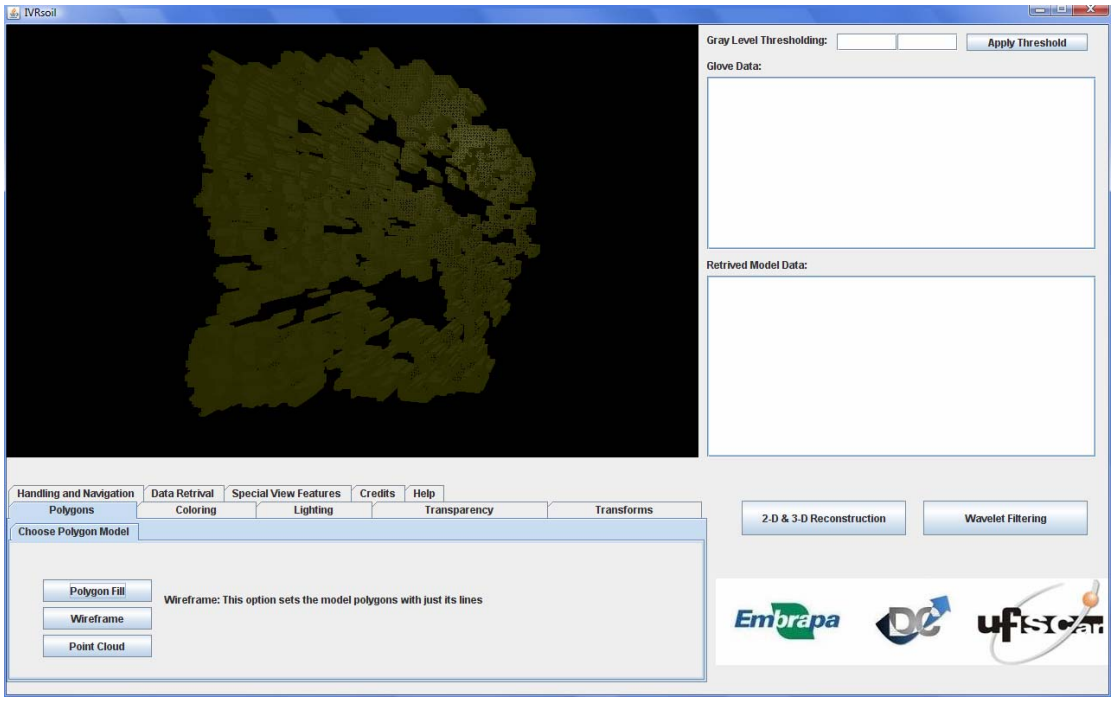


Figura 67 – Resultado da aplicação da classe Polígonos sobre uma amostra de cimento sob exibição de arestas (coeficiente POLYGON_LINES).

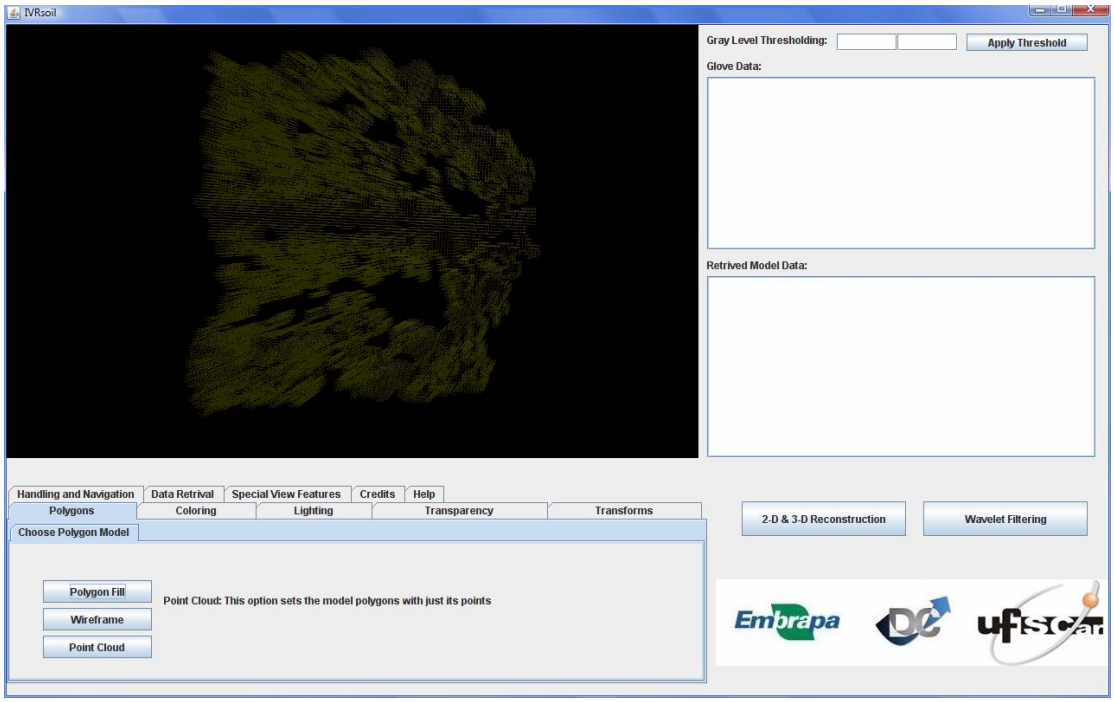


Figura 68 – Resultado da aplicação da classe Polígonos sobre uma amostra de cimento sob exibição de nuvem de pontos (coeficiente POLYGON_POINTS).

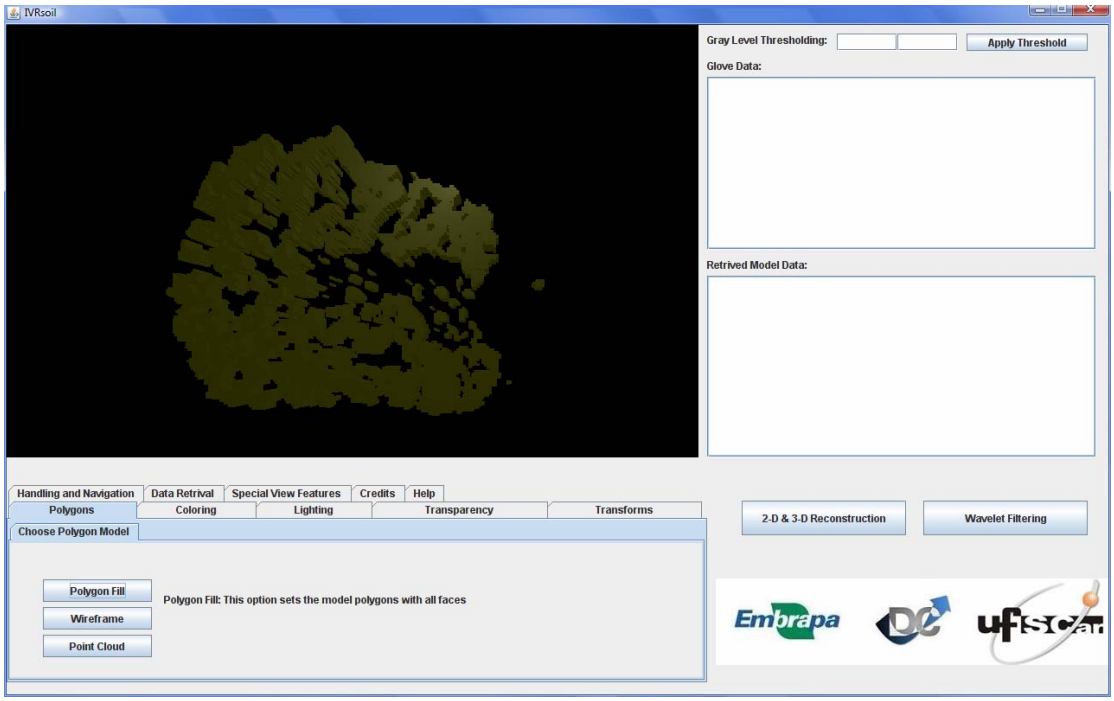


Figura 69 – Resultado da aplicação da classe Polígonos sobre uma amostra de cimento sob exibição de faces (coeficiente POLYGON_FACES).

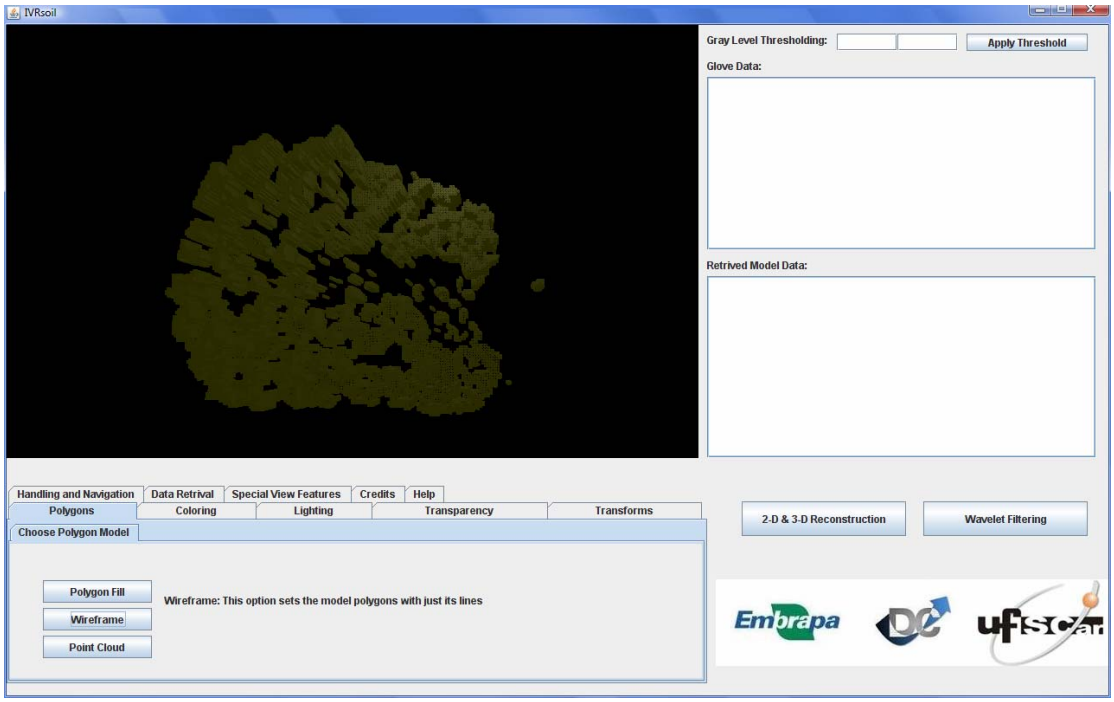


Figura 70 – Resultado da aplicação da classe Polígonos sobre uma amostra de cimento sob exibição de arestas (coeficiente POLYGON_LINES).

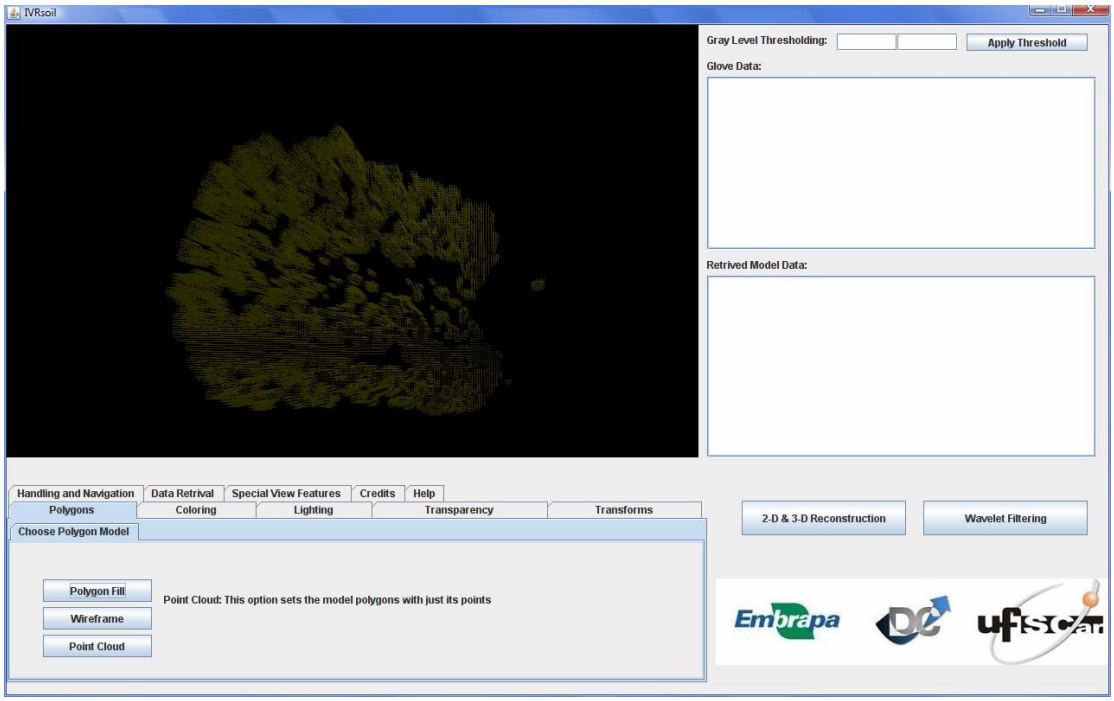


Figura 71 – Resultado da aplicação da classe Polígonos sobre uma amostra de cimento sob exibição de nuvem de pontos (coeficiente POLYGON_POINTS).

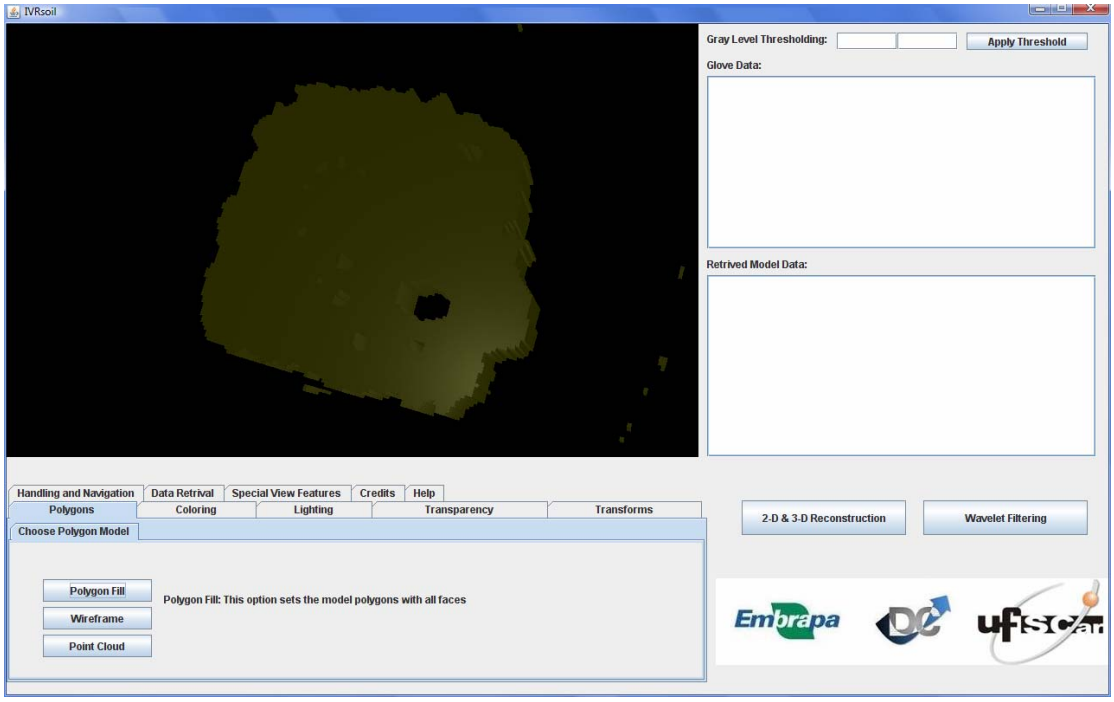


Figura 72 – Resultado da aplicação da classe Polígonos sobre uma amostra de solo degradado sob exibição de faces (coeficiente POLYGON_FACES).

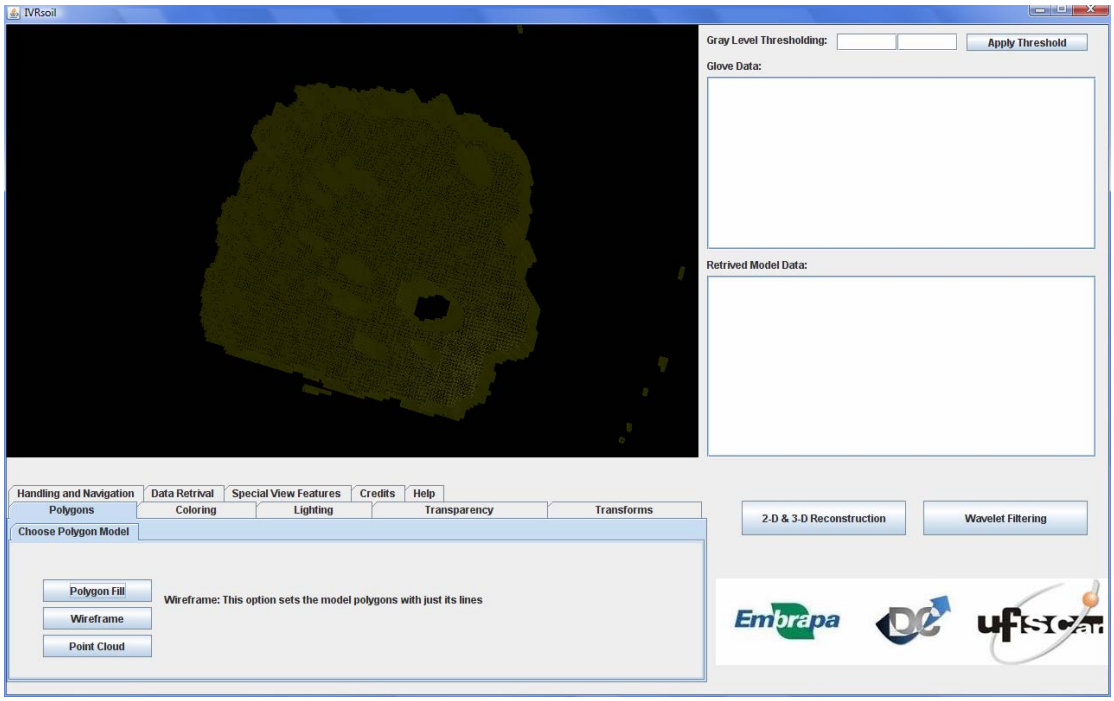


Figura 73 – Resultado da aplicação da classe Polígonos sobre uma amostra de solo degradado sob exibição de arestas (coeficiente POLYGON_LINES).

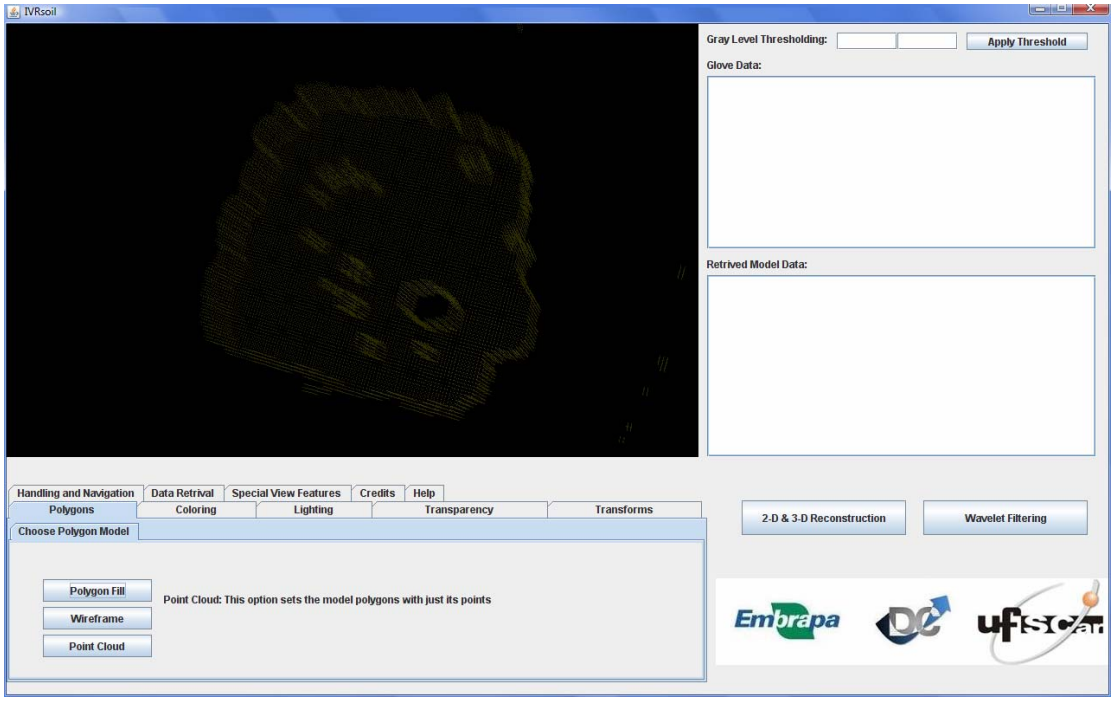


Figura 74 – Resultado da aplicação da classe Polígonos sobre uma amostra de solo degradado sob exibição de nuvem de pontos (coeficiente POLYGON_POINTS).

5.6. *Threshold*

Utilizando-se dos parâmetros de níveis de cinza superior e inferior, escolhidos pelo usuário através dos campos presentes na interface principal, a classe de visualizar uma amostra com diversos níveis de cinza em sua composição cromática, realizou a efetiva separação dos mesmos, permitindo a visualização de somente uma determinada região de interesse ao usuário. Tal operação facilita a localização dos poros e análise de estruturas interiores da imagem da amostra, uma vez que é possível visualizar pontos, arestas e faces outrora oclusas pelas estruturas. As Figuras 75, 76 e 77 ilustram os resultados da utilização da classe *Threshold* sobre uma imagem de amostra de areia, sob os limiares 200-255, 100-199, 10-99, respectivamente. As Figuras 78, 79, 80, 81, 82 e 83 ilustram os resultados da utilização da mesma classe sobre duas imagens de amostras de solo degradado sob os mesmos limiares para cada modelo tridimensional.

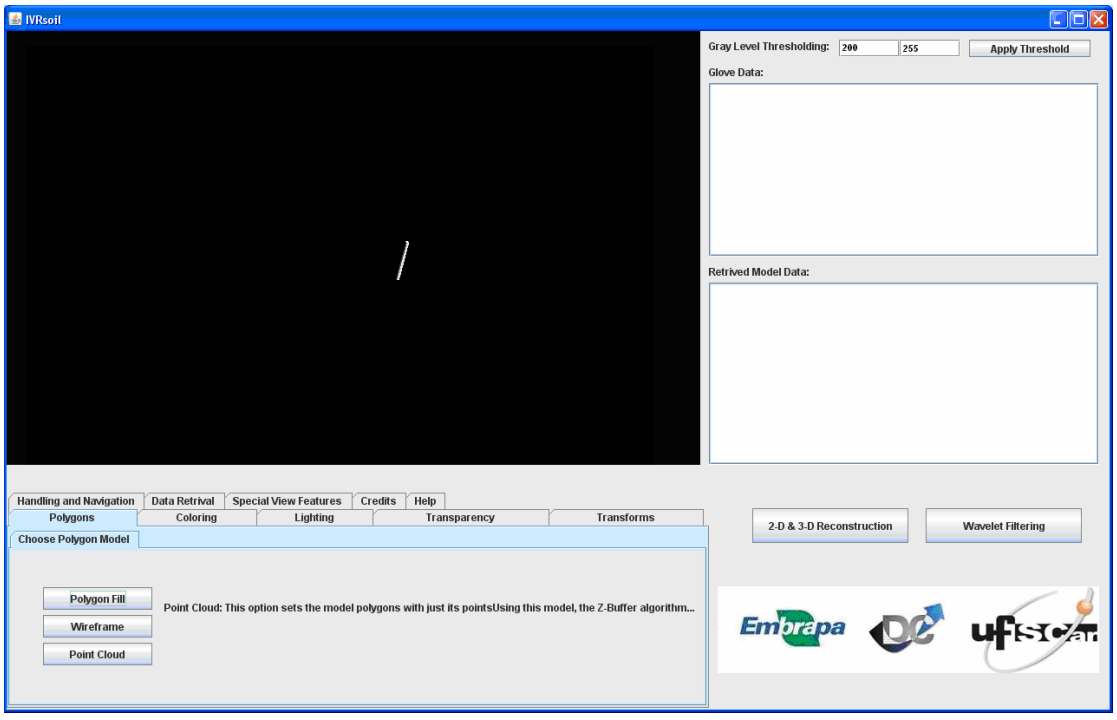


Figura 75 – Resultado da aplicação do limiar de 200-255 da classe Threshold sobre a imagem tomográfica da amostra de areia.

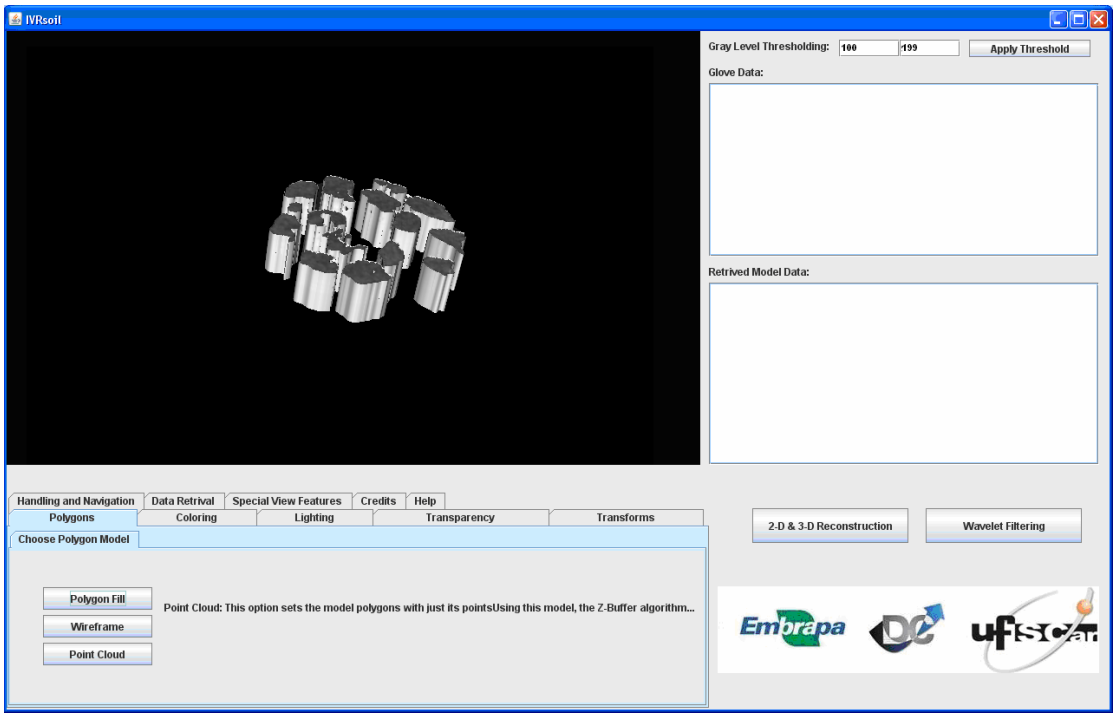


Figura 76 – Resultado da aplicação do limiar 100-199 da classe Threshold sobre a imagem tomográfica da amostra de areia.

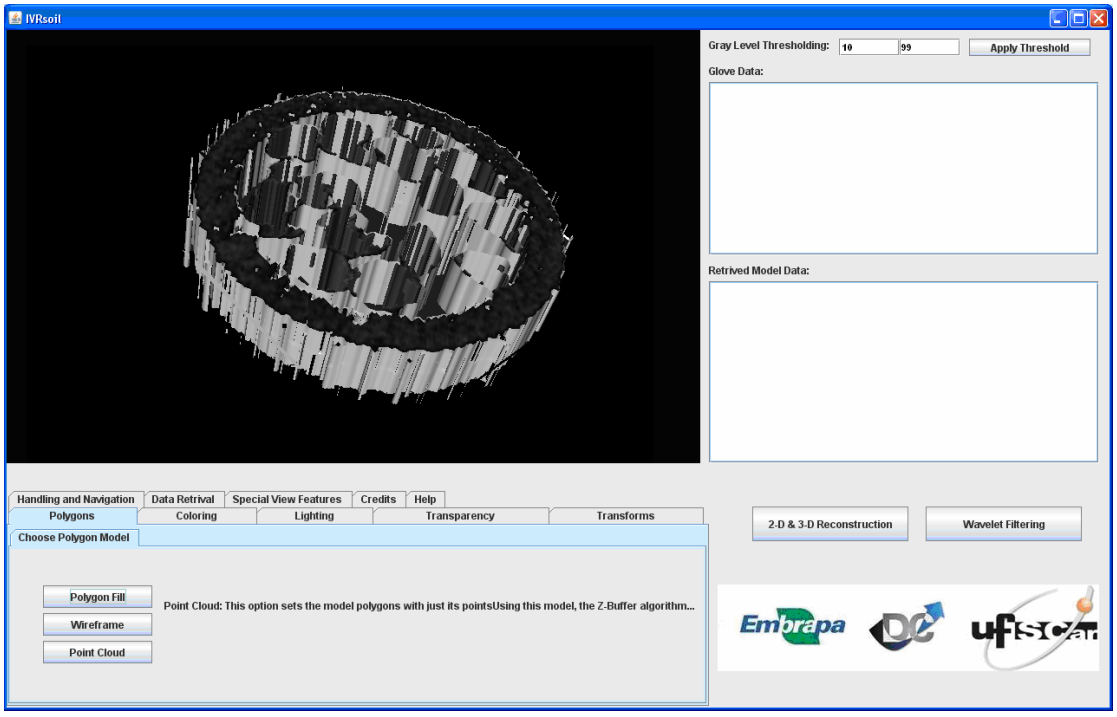


Figura 77 – Resultado da aplicação do limiar 10-99 da classe Threshold sobre a imagem tomográfica da amostra de areia.

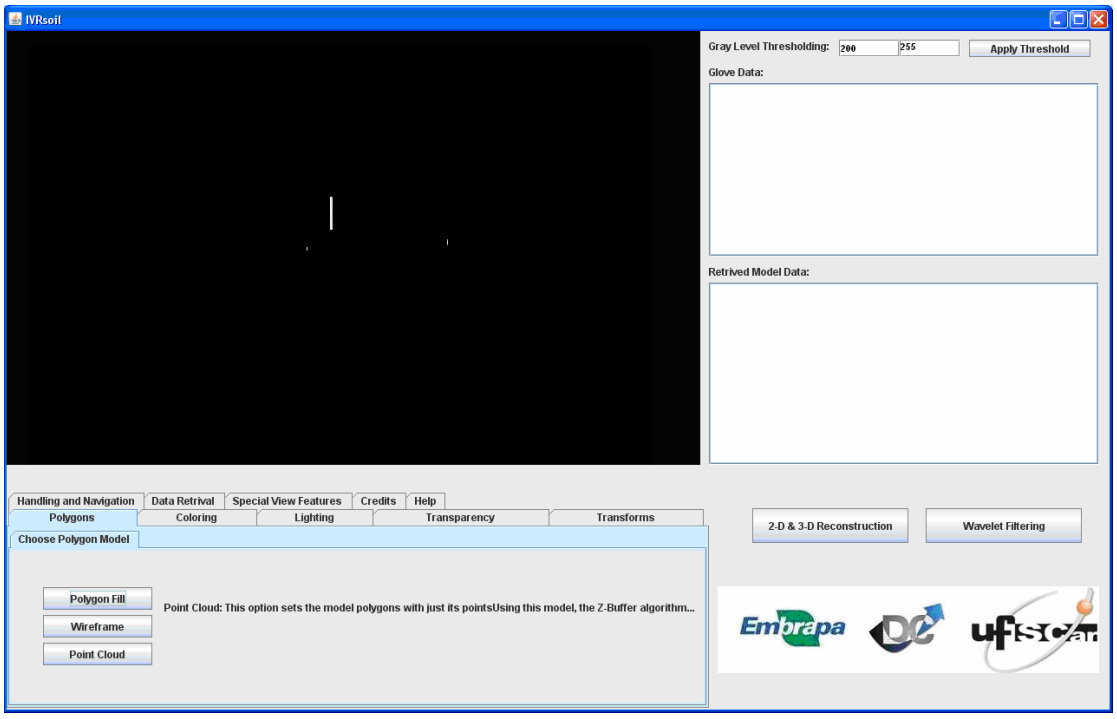


Figura 78 – Resultado da aplicação do limiar 200-255 da classe Threshold sobre a imagem tomográfica da amostra de solo degradado.

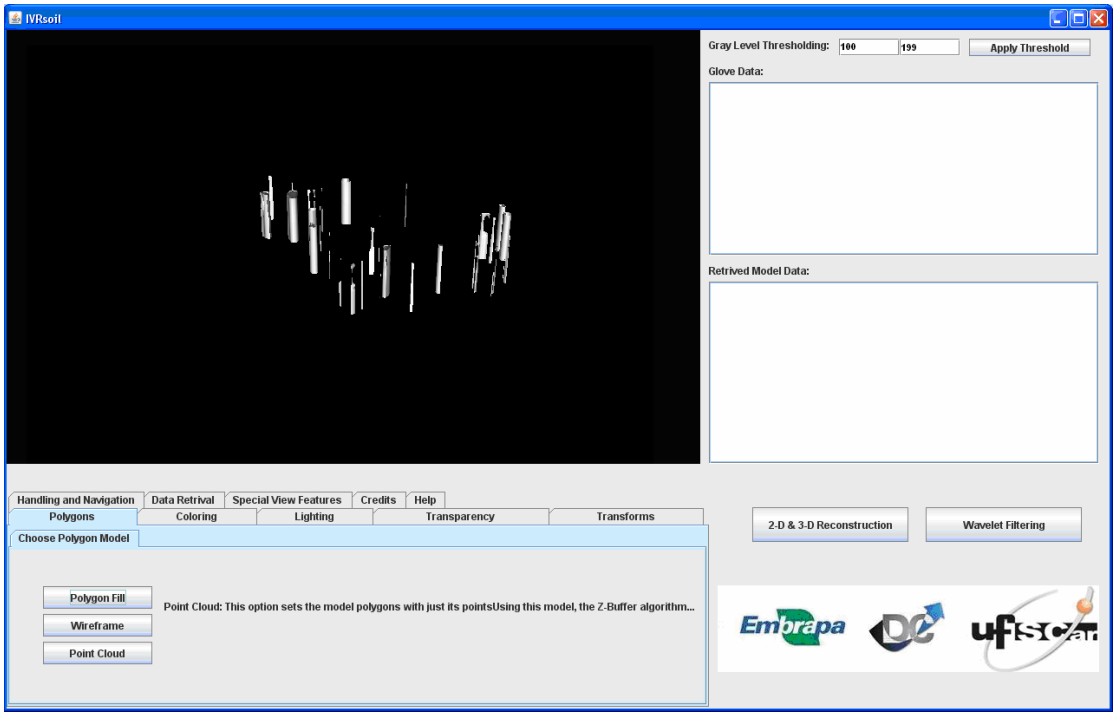


Figura 79 – Resultado da aplicação do limiar 100-199 da classe Threshold sobre a imagem tomográfica da amostra de solo degradado.

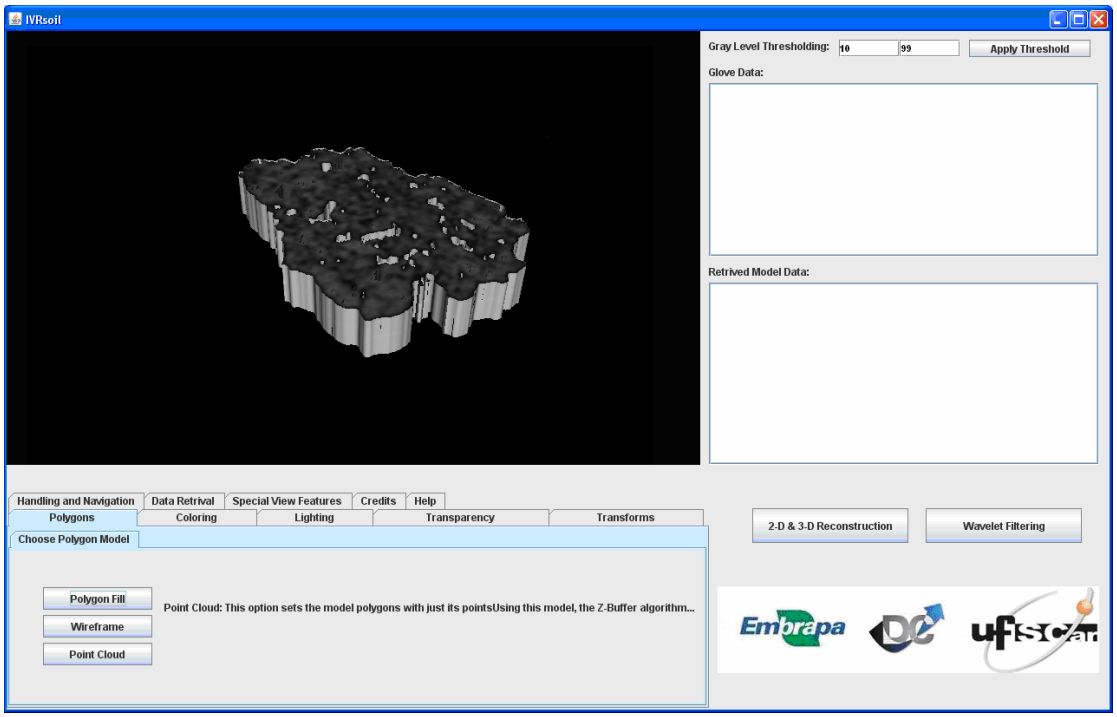


Figura 80 – Resultado da aplicação do limiar 10-99 da classe Threshold sobre a imagem tomográfica da amostra de solo degradado

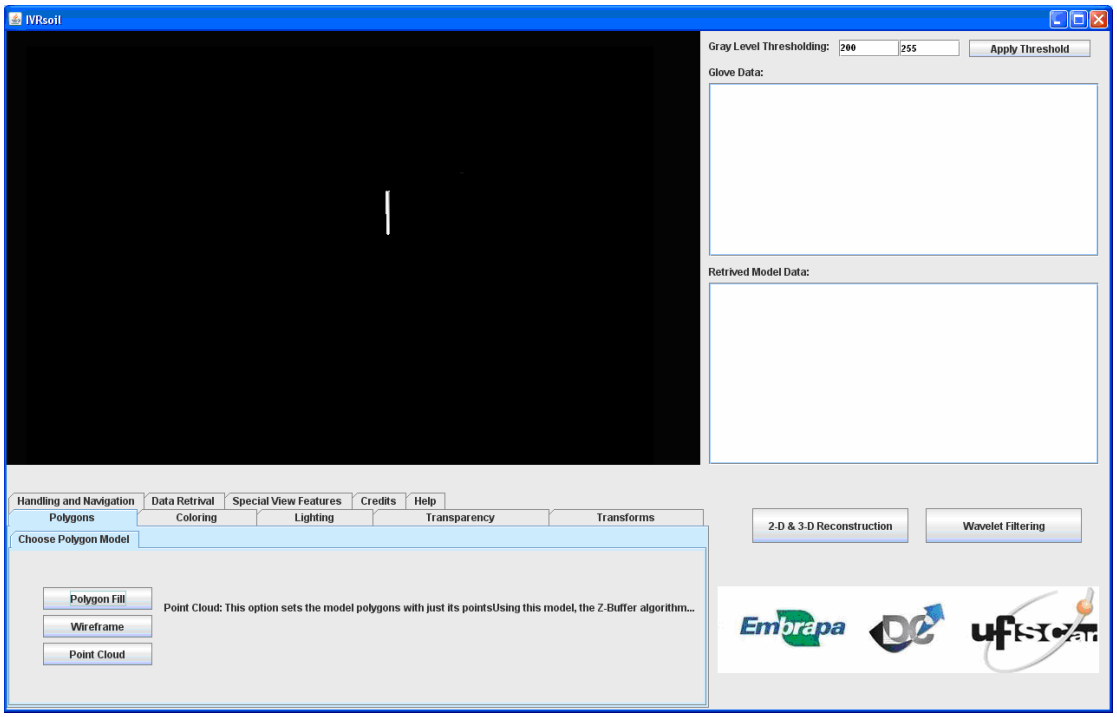


Figura 81 – Resultado da aplicação do limiar 200-255 da classe Threshold sobre a imagem tomográfica da amostra de cimento.

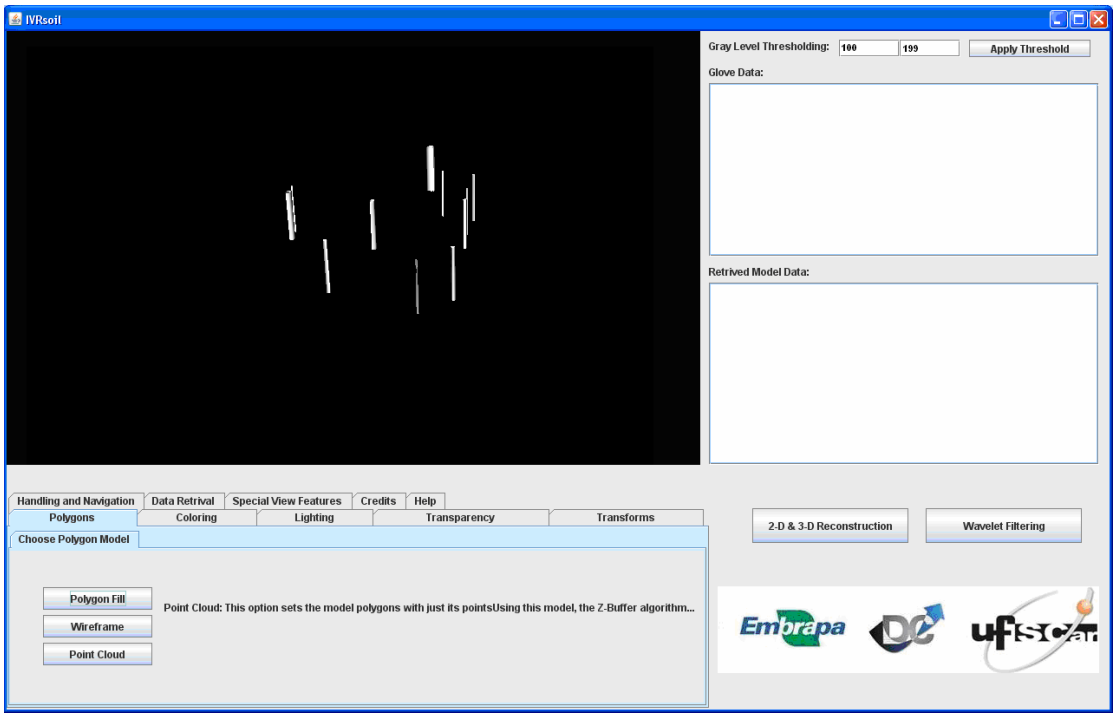


Figura 82 – Resultado da aplicação do limiar 100-199 da classe Threshold sobre a imagem tomográfica da amostra de cimento.

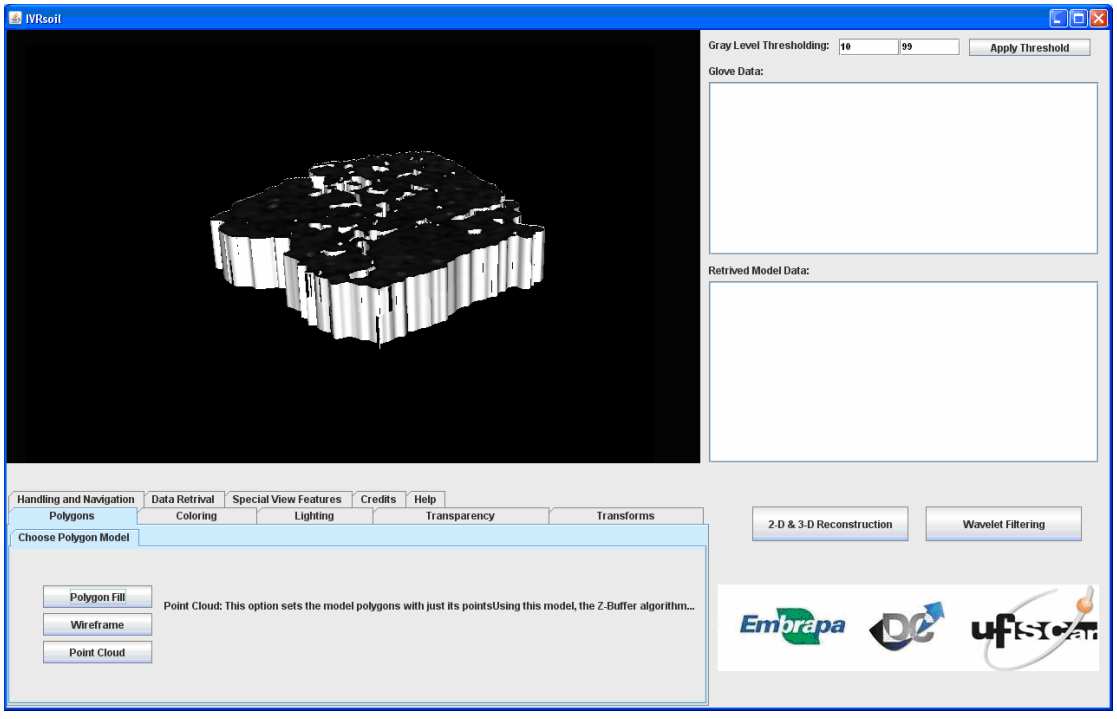


Figura 83 – Resultado da aplicação do limiar 10-99 da classe Threshold sobre a imagem tomográfica da amostra de cimento.

5.7. Extração de Atributos

Através da classe *Extração de Atributos*, características intrínsecas da cena e de amostras agrícolas puderam ser obtidas, através de *mouse* ou da *P5Glove*, tendo como origem dos dados as representações tridimensionais, as quais estão presentes apenas nas projeções 1-D ou então nas imagens 2-D, como é o caso dos níveis de cinza e coeficientes de atenuação linear.

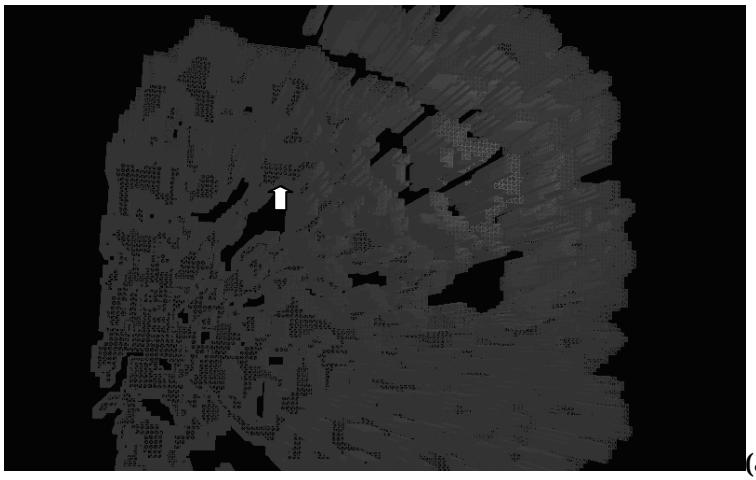
Tais dados foram divididos em duas categorias: Relativos à Cena e Relativos aos dados Tomográficos. Dentre os dados da cena sintética, os dados obtidos foram: as fronteiras, as quais representam os limites da geometria ou os limites da geometria que a envolve; o grafo de cena, que representa a hierarquia dos nodos presentes na árvore; a geometria atual no modelo e sua composição; a distância de determinado voxel em relação às coordenadas escolhidas na cena; o vértice mais próximo ao ponto escolhido na cena; as coordenadas tridimensionais de tais coordenadas; e a reta normal na face mais próxima que envolve as coordenadas escolhidas.

Quanto aos dados tomográficos, os dados obtidos foram; atributos de cor, que representam as cores individuais de cada voxel, independente da intensidade de iluminação; os coeficientes de material, que representam as cores que cada voxel reflete ao se incidir determinada intensidade de luz; atributos de transparência; atributos de polígonos; a luminância (tons de cinza), coeficiente do sistema *HSL* de cores, o qual representa a intensidade de níveis de cinza encontrado no voxel escolhido; a saturação e matiz, também do coeficiente *HSL*; e finalmente o coeficiente de atenuação linear, calculado com base nos níveis de cinza.

Em teste realizado mediante escolha de um voxel arbitrário, obteve-se dados relativos à cena e à amostra tomográfica:

As Figuras 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94 e 95 ilustram resultados do processo de extração de dados por escolha de voxels, utilizando-se do *mouse* ou da luva de dados *P5Glove*.

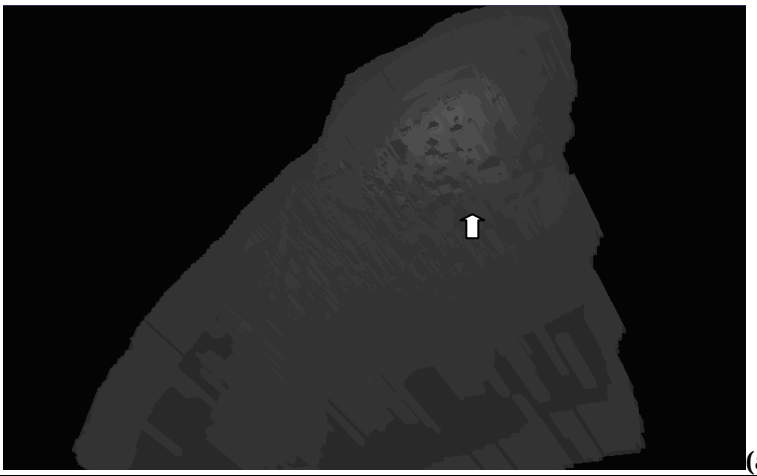
Os resultados produzidos são exibidos em um local específico na interface principal do Ambiente de Realidade Virtual, denominado *dados obtidos do modelo*, localizado no quadrante inferior direito da mesma, e foram ampliados para efeito de observação.



(a)

javax.media.j3d.ColoringAttributes: Color=(0.03, 0.07, 0.04) ShadeModel=SHADE_GOURAUD
javax.media.j3d.Material: AmbientColor=(0.4, 0.4, 0.4) EmissiveColor=(0.0, 0.0, 0.0) DiffuseColor=(0.71, 0.70, 0.65) SpecularColor=(0.3, 0.3, 0.3) Shininess=128.0 LightingEnable=true ColorTarget=2
Transparency Level: 0.3
Polygons (2-Fill/1-Wireframe/0-Points): 1
Gray Level: 156.82
Saturation: 22.86
Attenuation Coefficient: 0.6521
Bounding box: Lower=-0.87 -1.0 -0.15 Upper=0.875 1.0 0.15
BranchGraphs: 3
Geometry: javax.media.j3d.TriangleArray@4b0bbb
Distance: 10.28
Closest Vertex: (0.92, -0.25, -10.24)
Point Coordinates: (0.75, -0.25, -10.26)
Point Normal: (1.0, 0.0, 0.0) **(b)**

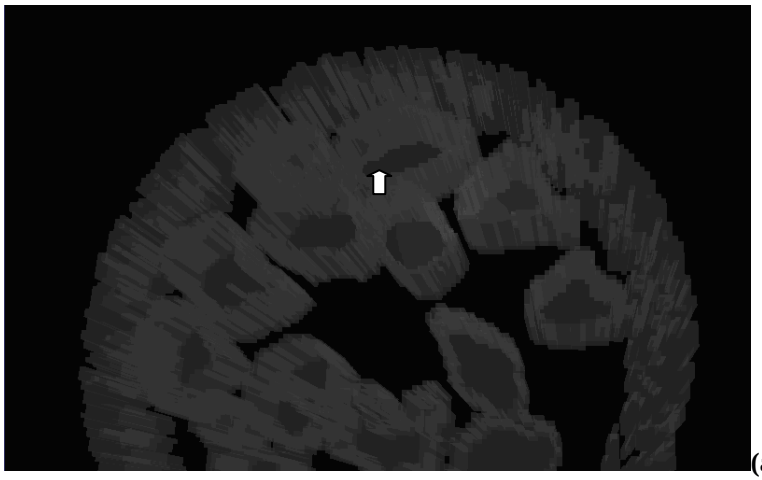
Figura 84– Resultado da obtenção de dados da amostra de adubo: (a) Amostra (b) Dados obtidos



(a)

javax.media.j3d.ColoringAttributes: Color=(0.01, 0.02, 0.02) ShadeModel=SHADE_GOURAUD
javax.media.j3d.Material: AmbientColor=(0.1, 0.1, 0.1) EmissiveColor=(0.0, 0.0, 0.0) DiffuseColor=(0.39, 0.10, 0.25) SpecularColor=(0.2, 0.2, 0.2) Shininess=128.0 LightingEnable=true ColorTarget=2
Transparency Level: 0.3
Polygons (2-Fill/1-Wireframe/0-Points): 2
Gray Level: 89.0
Saturation: 12.52
Attenuation Coefficient: 0.12
Bounding box: Lower=-0.875 -1.0 -0.15 Upper=0.875 1.0 0.15
BranchGraphs: 3
Geometry: javax.media.j3d.TriangleArray@4b0bbb
Distance: 6.26
Closest Vertex: (0.6.1, 0.35, -6.36)
Point Coordinates: (0.48, 0.35, -6.48)
Point Normal: (1.0, 0.0, 0.0) **(b)**

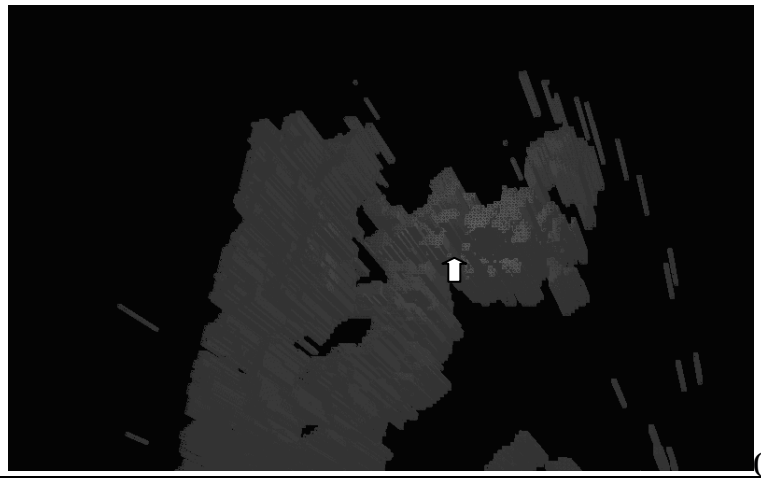
Figura 85 – Resultado da obtenção de dados da amostra de argila: (a) Amostra (b) Dados obtidos



(a)

javax.media.j3d.ColoringAttributes: Color=(0.16, 0.09, 0.18) ShadeModel=SHADE_GOURAUD
javax.media.j3d.Material: AmbientColor=(0.2, 0.2, 0.2) EmissiveColor=(0.0, 0.0, 0.0) DiffuseColor=(0.16, 0.09, 0.18) SpecularColor=(0.3, 0.3, 0.3) Shininess=128.0 LightingEnable=true ColorTarget=2
Transparency Level: 0.5
Polygons (2-Fill/1-Wireframe/0-Points): 1
Gray Level: 59.45
Saturation: 8.03
Attenuation Coefficient: 0.96
Bounding box: Lower=-0.87 -1.0 -0.15 Upper=0.87 1.0 0.15
BranchGraphs: 3
Geometry: javax.media.j3d.TriangleArray@4b0bbb
Distance: 8.15
Closest Vertex: (0.15, -0.12, -8.98)
Point Coordinates: (0.75, -0.52, -8.25)
Point Normal: (1.0, 0.0, 0.0) **(b)**

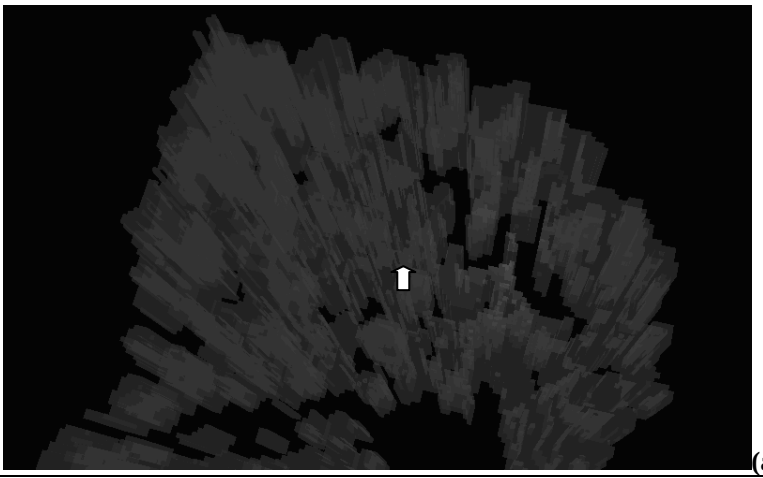
Figura 86 – Resultado da obtenção de dados da amostra de areia: (a) Amostra (b) Dados obtidos



(a)

javax.media.j3d.ColoringAttributes: Color=(0.02, 0.07, 0.04) ShadeModel=SHADE_GOURAUD
javax.media.j3d.Material: AmbientColor=(0.4, 0.4, 0.4) EmissiveColor=(0.0, 0.0, 0.0) DiffuseColor=(0.37, 0.71, 0.64) SpecularColor=(0.3, 0.3, 0.3) Shininess=128.0 LightingEnable=true ColorTarget=2
Transparency Level: 0.2
Polygons (2-Fill/1-Wireframe/0-Points): 2
Gray Level: 186.21
Saturation: 32.46
Attenuation Coefficient: 0.81
Bounding box: Lower=-0.87 -1.0 -0.15 Upper=0.87 1.0 0.154
BranchGraphs: 3
Geometry: javax.media.j3d.TriangleArray@4b0bbb
Distance: 9.43
Closest Vertex: (-0.16, -0.94, -9.34)
Point Coordinates: (-0.23, -0.75, -9.13)
Point Normal: (1.0, 0.0, 0.0) **(b)**

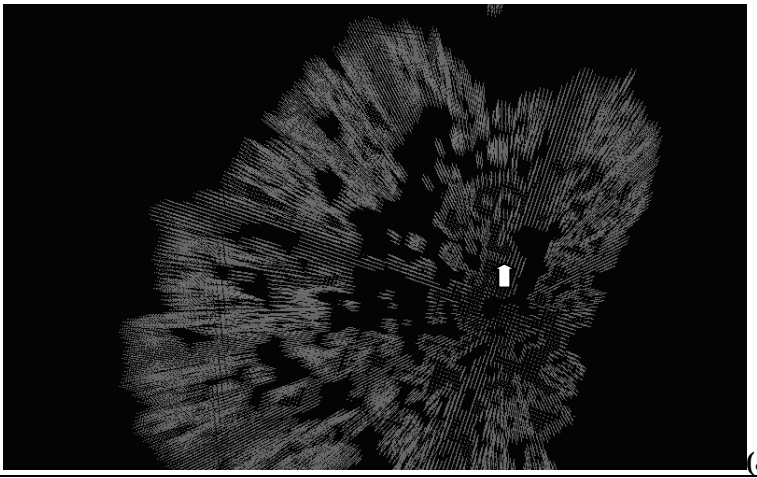
Figura 87 – Resultado da obtenção de dados da amostra de argila: (a) Amostra (b) Dados obtidos.



(a)

javax.media.j3d.ColoringAttributes: Color=(0.02, 0.02, 0.03) ShadeModel=SHADE_GOURAUD
javax.media.j3d.Material: AmbientColor=(0.2, 0.2, 0.2) EmissiveColor=(0.0, 0.0, 0.0) DiffuseColor=(0.71, 0.70, 0.65) SpecularColor=(0.3, 0.3, 0.3) Shininess=128.0 LightingEnable=true ColorTarget=2
Transparency Level: 0.7
Polygons (2-Fill/1-Wireframe/0-Points): 2
Gray Level: 75.96
Saturation: 9.186
Attenuation Coefficient: 0.52
Bounding box: Lower=-0.875 -1.0 -0.15 Upper=0.87 1.0 0.15
BranchGraphs: 3
Geometry: javax.media.j3d.TriangleArray@4b0bbb
Distance: 10.91
Closest Vertex: (0.13, 0.31, -10.98)
Point Coordinates: (0.13, 0.24, -10.18)
Point Normal: (1.0, 0.0, 0.0) **(b)**

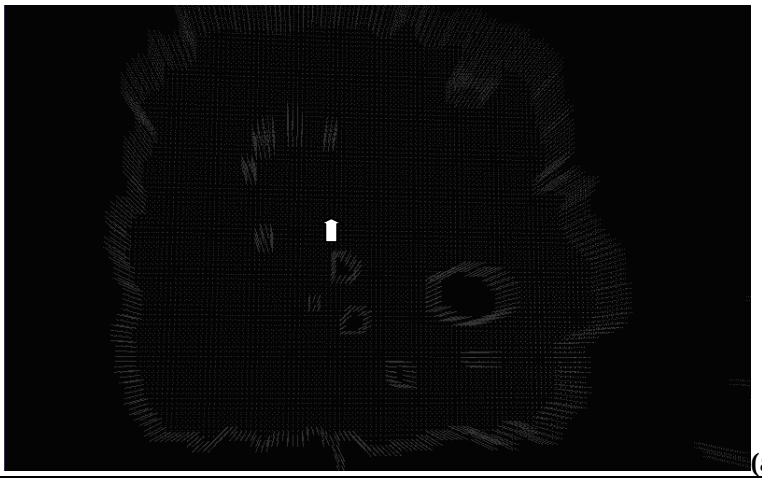
Figura 88 – Resultado da obtenção de dados da amostra de cimento: (a) Amostra, (b) Dados obtidos



(a)

javax.media.j3d.ColoringAttributes: Color=(0.03, 0.03, 0.04) ShadeModel=SHADE_GOURAUD
javax.media.j3d.Material: AmbientColor=(0.1, 0.1, 0.1) EmissiveColor=(0.0, 0.0, 0.0) DiffuseColor=(0.60, 0.50, 0.69) SpecularColor=(0.1, 0.1, 0.1) Shininess=128.0 LightingEnable=true ColorTarget=2
Transparency Level: 0.4
Polygons (2-Fill/1-Wireframe/0-Points): 0
Gray Level: 195.50
Saturation: 31.76
Attenuation Coefficient: 0.73
Bounding box: Lower=-0.87 -1.0 -0.15 Upper=0.87 1.0 0.15
BranchGraphs: 3
Geometry: javax.media.j3d.TriangleArray@4b0bbb
Distance: 6.36
Closest Vertex: (0.21, 0.32, -6.12)
Point Coordinates: (0.26, 0.24, -6.92)
Point Normal: (1.0, 0.0, 0.0) **(b)**

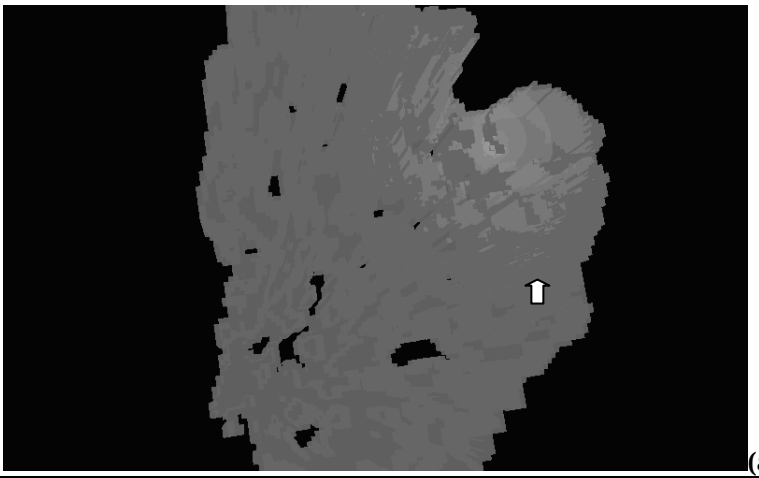
Figura 89 – Resultado da obtenção de dados da amostra de solo degradado: (a) Amostra, (b) Dados



(a)

javax.media.j3d.ColoringAttributes: Color=(0.03, 0.02, 0.02) ShadeModel=SHADE_GOURAUD
javax.media.j3d.Material: AmbientColor=(0.1, 0.1, 0.1) EmissiveColor=(0.0, 0.0, 0.0) DiffuseColor=(0.02, 0.03, 0.05) SpecularColor=(0.1, 0.1, 0.1) Shininess=128.0 LightingEnable=true ColorTarget=2
Transparency Level: 0.8
Polygons (2-Fill/1-Wireframe/0-Points): 0
Gray Level: 62.78
Saturation: 8.45
Attenuation Coefficient: 0.38
Bounding box: Lower=-0.87 -1.0 -0.15 Upper=0.87 1.0 0.15
BranchGraphs: 3
Geometry: javax.media.j3d.TriangleArray@4b0bbb
Distance: 4.68
Closest Vertex: (0.16, -0.16, -4.94)
Point Coordinates: (0.24, -0.19, -4.28)
Point Normal: (1.0, 0.0, 0.0) **(b)**

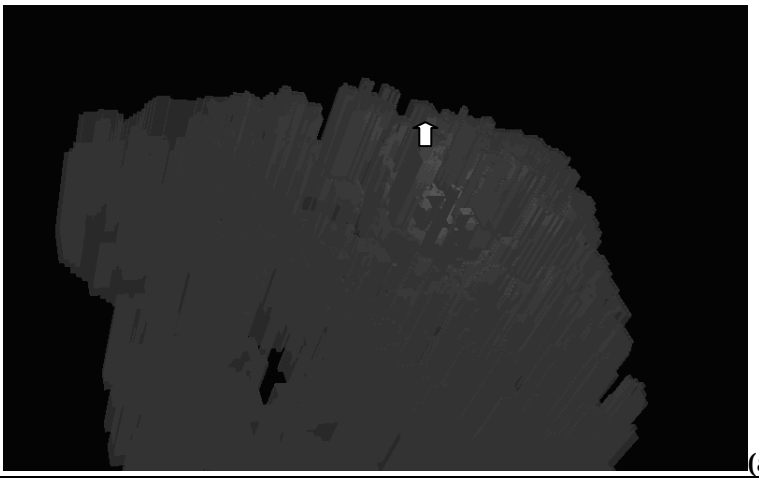
Figura 90 – Resultado da obtenção de dados da amostra de solo degradado: (a) Amostra, (b) Dados



(a)

javax.media.j3d.ColoringAttributes: Color=(0.15, 0.17, 0.14) ShadeModel=SHADE_GOURAUD
javax.media.j3d.Material: AmbientColor=(0.4, 0.4, 0.4) EmissiveColor=(0.0, 0.0, 0.0) DiffuseColor=(0.4, 0.4, 0.4) SpecularColor=(0.3, 0.3, 0.3) Shininess=128.0 LightingEnable=true ColorTarget=2
Transparency Level: 0.2
Polygons (2-Fill/1-Wireframe/0-Points): 2
Gray Level: 202.0
Saturation: 20.68
Attenuation Coefficient: 0.75
Bounding box: Lower=-0.87 -1.0 -0.15 Upper=0.87 1.0 0.15
BranchGraphs: 3
Geometry: javax.media.j3d.TriangleArray@4b0bbb
Distance: 12.68
Closest Vertex: (0.05, 0.07, -12.68)
Point Coordinates: (0.09, 0.08, -12.94)
Point Normal: (1.0, 0.0, 0.0) **(b)**

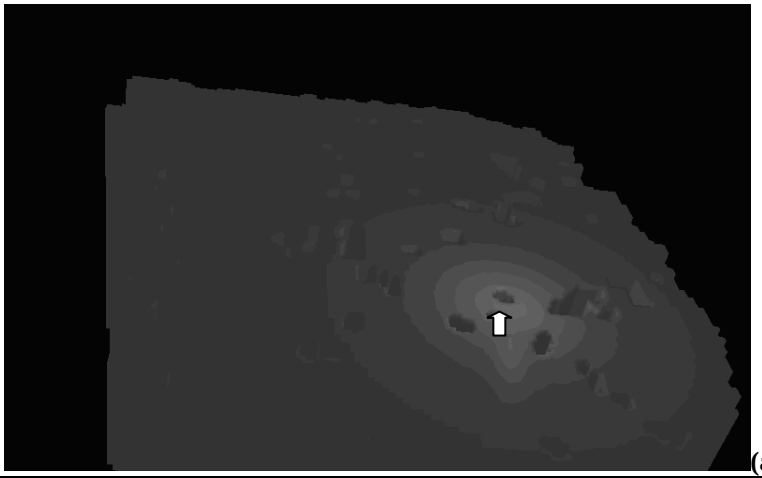
Figura 91 – Resultado da obtenção de dados da amostra de cimento: (a) Amostra, (b) Dados obtidos.



(a)

javax.media.j3d.ColoringAttributes: Color=(0.09, 0.09, 0.09) ShadeModel=SHADE_GOURAUD
javax.media.j3d.Material: AmbientColor=(0.3, 0.3, 0.3) EmissiveColor=(0.0, 0.0, 0.0) DiffuseColor=(0.3, 0.3, 0.3) SpecularColor=(0.3, 0.3, 0.3) Shininess=128.0 LightingEnable=true ColorTarget=2
Transparency Level: 0.3
Polygons (2-Fill/1-Wireframe/0-Points): 2
Gray Level: 124.97
Saturation: 6.15
Attenuation Coefficient: 0.41
Bounding box: Lower=-0.87 -1.0 -0.15 Upper=0.87 1.0 0.15
BranchGraphs: 3
Geometry: javax.media.j3d.TriangleArray@4b0bbb
Distance: 3.68
Closest Vertex: (1.31, 0.31, -3.74)
Point Coordinates: (1.33, 0.20, -3.95)
Point Normal: (1.0, 0.0, 0.0) **(b)**

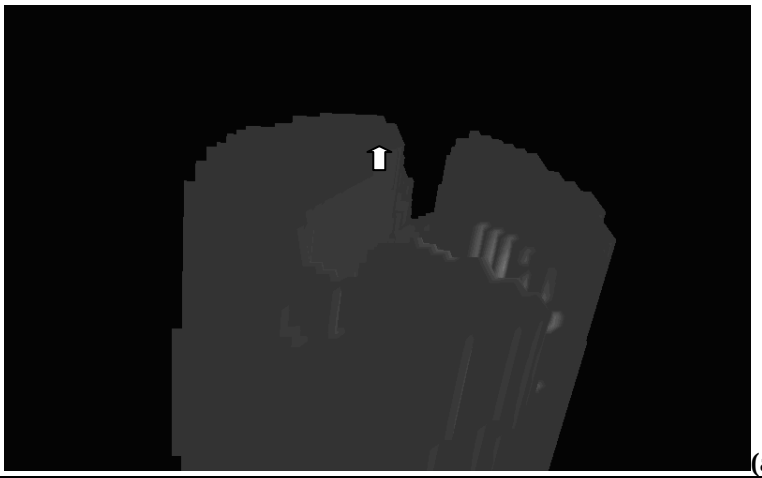
Figura 92 – Resultado da obtenção de dados da amostra de solo degradado: (a) Amostra, (b) Dados



(a)

javax.media.j3d.ColoringAttributes: Color=(0.18, 0.17, 0.14) ShadeModel=SHADE_GOURAUD
javax.media.j3d.Material: AmbientColor=(0.4, 0.4, 0.4) EmissiveColor=(0.0, 0.0, 0.0) DiffuseColor=(0.4, 0.4, 0.4) SpecularColor=(0.3, 0.3, 0.3) Shininess=128.0 LightingEnable=true ColorTarget=2
Transparency Level: 0.1
Polygons (2-Fill/1-Wireframe/0-Points): 2
Gray Level: 198.00
Saturation: 20.92
Attenuation Coefficient: 0.58
Bounding box: Lower=-0.87 -1.0 -0.15 Upper=0.875 1.0 0.1
BranchGraphs: 3
Geometry: javax.media.j3d.TriangleArray@4b0bbb
Distance: 3.67
Closest Vertex: (0.12, 0.36, -3.68)
Point Coordinates: (0.28, 0.39, -3.81)
Point Normal: (1.0, 0.0, 0.0) **(b)**

Figura 93 – Resultado da obtenção de dados da amostra de solo (a) Amostra, (b) Dados obtidos.



(a)

javax.media.j3d.ColoringAttributes: Color=(0.09, 0.94, 0.09) ShadeModel=SHADE_GOURAUD
javax.media.j3d.Material: AmbientColor=(0.4, 0.4, 0.4) EmissiveColor=(0.0, 0.0, 0.0) DiffuseColor=(0.4, 0.4, 0.4) SpecularColor=(0.3, 0.3, 0.3) Shininess=128.0 LightingEnable=true ColorTarget=2
Transparency Level: 0.2
Polygons (2-Fill/1-Wireframe/0-Points): 2
Gray Level: 180.0
Saturation: 16.64
Attenuation Coefficient: 0.51
Bounding box: Lower=-0.87 -1.0 -0.15 Upper=0.87 1.0 0.15
BranchGraphs: 3
Geometry: javax.media.j3d.TriangleArray@4b0bbb
Distance: 11.94
Closest Vertex: (0.15, 0.16, 11.63)
Point Coordinates: (0.12, 0.09, -11.98)
Point Normal: (1.0, 0.0, 0.0) **(b)**

Figura 94 – Resultado da obtenção de dados da amostra de solo degradado: (a) Amostra, (b) Dados

(a)

javax.media.j3d.ColoringAttributes: Color=(0.06, 0.06, 0.06) ShadeModel=SHADE_GOURAUD
javax.media.j3d.Material: AmbientColor=(0.1, 0.1, 0.1) EmissiveColor=(0.0, 0.0, 0.0) DiffuseColor=(0.1, 0.1, 0.1) SpecularColor=(0.3, 0.3, 0.3) Shininess=128.0 LightingEnable=true ColorTarget=2
Transparency Level: 0.3
Polygons (2-Fill/1-Wireframe/0-Points): 1
Gray Level: 44.00
Saturation: 7.65
Attenuation Coefficient: 0.39
Bounding box: Lower=-0.87 -1.0 -0.15 Upper=0.87 1.0 0.15
BranchGraphs: 3
Geometry: javax.media.j3d.TriangleArray@4b0bbb
Distance: 13.63
Closest Vertex: (0.00, 0.00, -13.97)
Point Coordinates: (0.00, 0.00, -13.57)
Point Normal: (1.0, 0.0, 0.0) **(b)**

Figura 95 – Resultado da obtenção de dados da amostra de vidro: (a) Amostra, (b) Dados obtidos.

5.8. Iluminação

Através do sistema de *Iluminação*, todas as luzes da cena atual podem ser alteradas em tempo de execução, traduzindo os campos de cada janela de parâmetros, em *feedbacks* visuais, dependendo da fonte de luz e da refletância das superfícies dos materiais iluminados, independente do tipo de polígonos renderizado. Dependendo do tipo de iluminação adotado, parâmetros são também adotados ou ignorados, entretando, sempre preservando a cor, os limites e direção dos raios de luz. Desta maneira, a fonte de luz é inserida na cena e atinge as retas normais, iluminando as entidades presentes na mesma. As Figuras 96, 97, 98 e 99 ilustram a utilização da iluminação Ambiente, Direcional, Pontual e Angular (Spot), respectivamente, bem como seus conjuntos de parâmetros, em função da modalidade, sobre uma imagem de uma amostra de solo degradado. As Figuras 100, 101, 102 e 103 ilustram a utilização das mesmas fontes sobre uma imagem de uma amostra de cimento. As Figuras 104, 105, 106 e 107 ilustram a utilização da classe *Iluminação*, também das mesmas fontes de iluminação, sobre uma imagem de uma amostra de solo degradado. As cores das fontes de luz foram geradas randomicamente para efeito de observação.

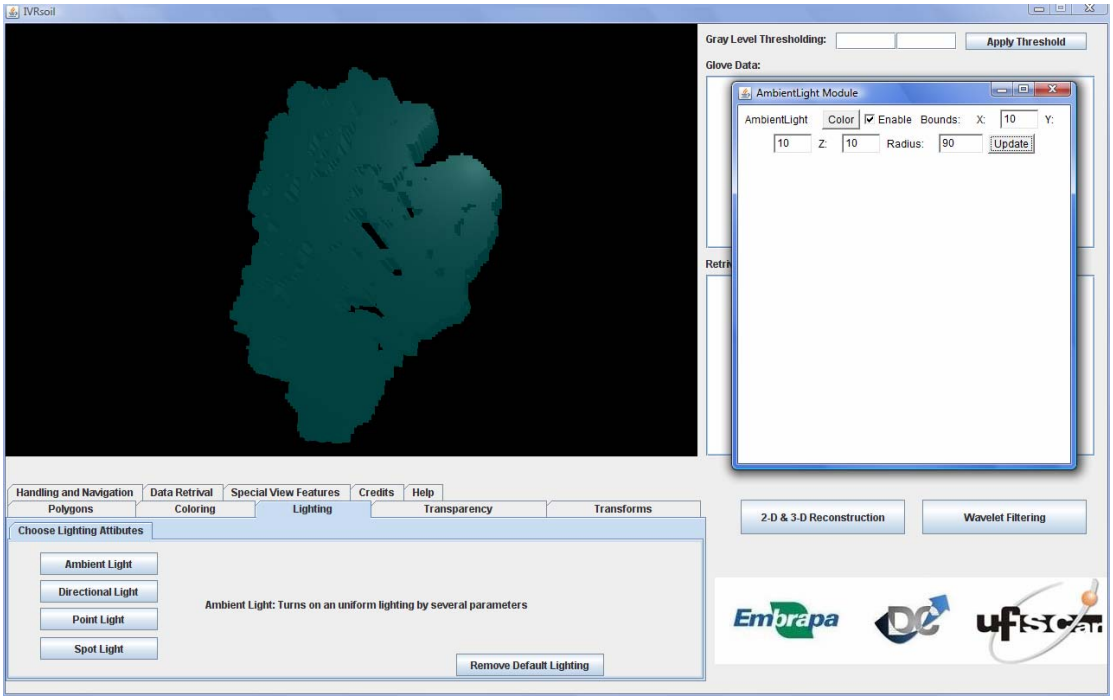


Figura 96 – Resultado da aplicação de fonte de luz Ambiente sobre uma amostra de solo degradado com os parâmetros: Fronteiras = (10, 10, 10); Raio = 90.

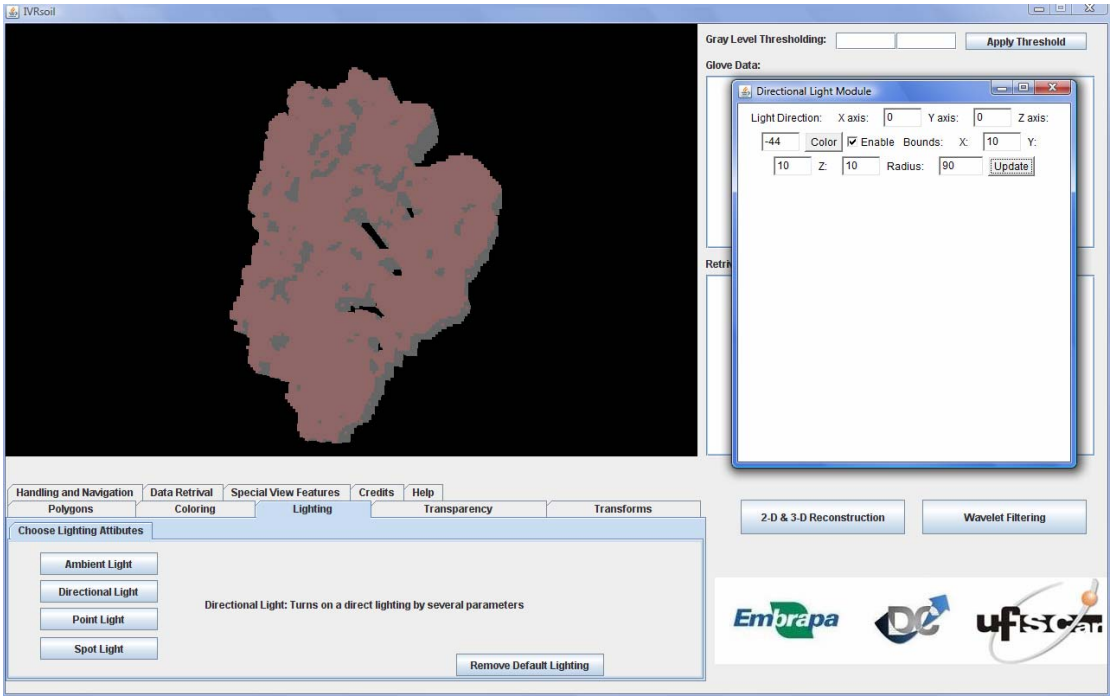


Figura 97 – Resultado da aplicação de fonte de luz Direcional sobre uma amostra de solo degradado com os parâmetros: Fronteiras = (10, 10, 10); Raio = 90; Direção da luz = (0, 0, -44).

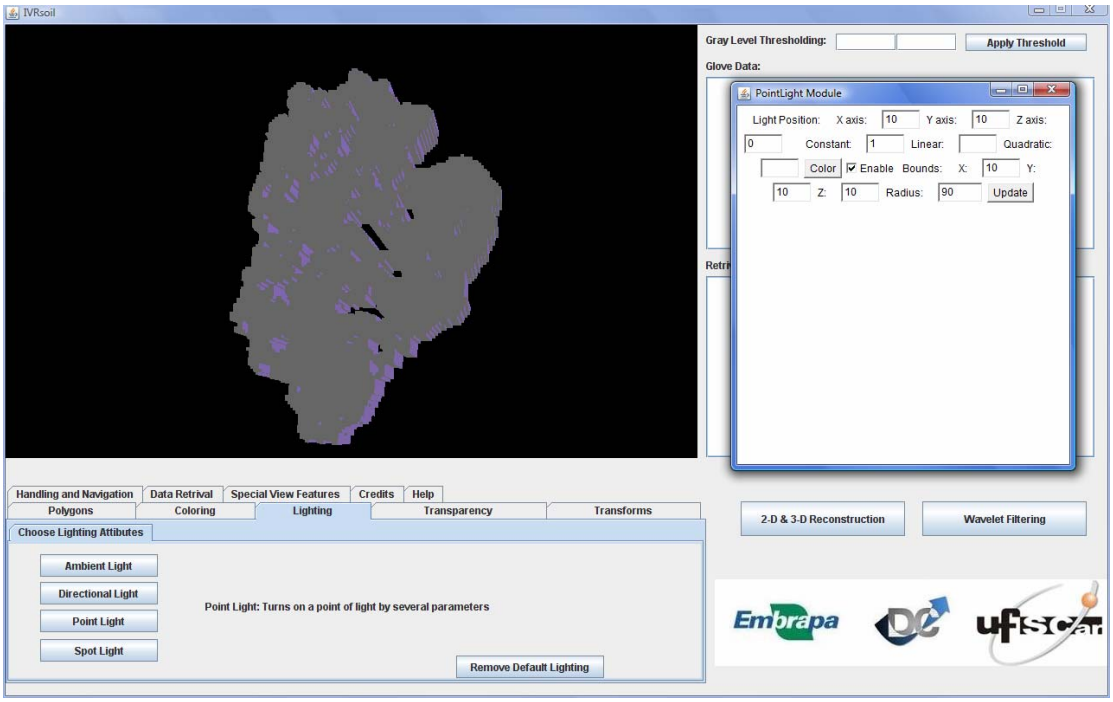


Figura 98 – Resultado da aplicação de fonte de luz Pontual sobre uma amostra de solo degradado com os parâmetros: Fronteiras = (10, 10, 10); Raio = 90; Posição = (10, 10, 0); Constante de atenuação = 1.

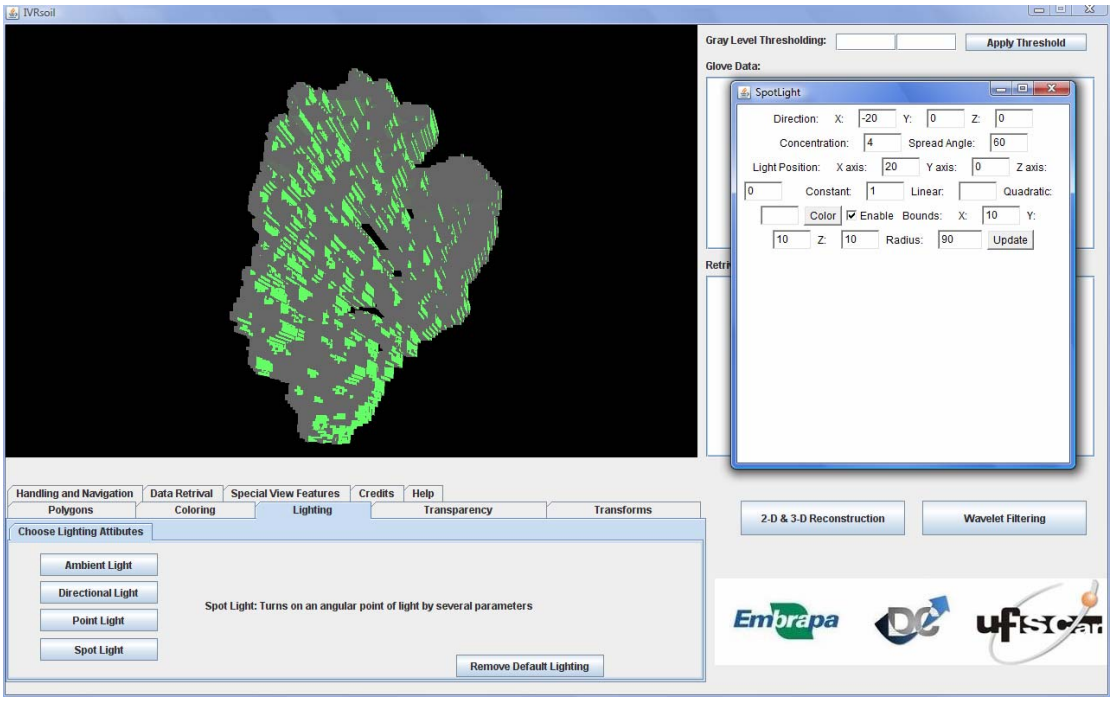


Figura 99 – Resultado da aplicação de fonte de luz Angular sobre uma amostra de solo degradado com os parâmetros: Fronteiras = (10, 10, 10); Raio = 90; Direção = (-20, 0, 0); Concentração = 4; Angulo de espalhamento = 60; Posição = (20, 0, 0); Constante de atenuação = 1.

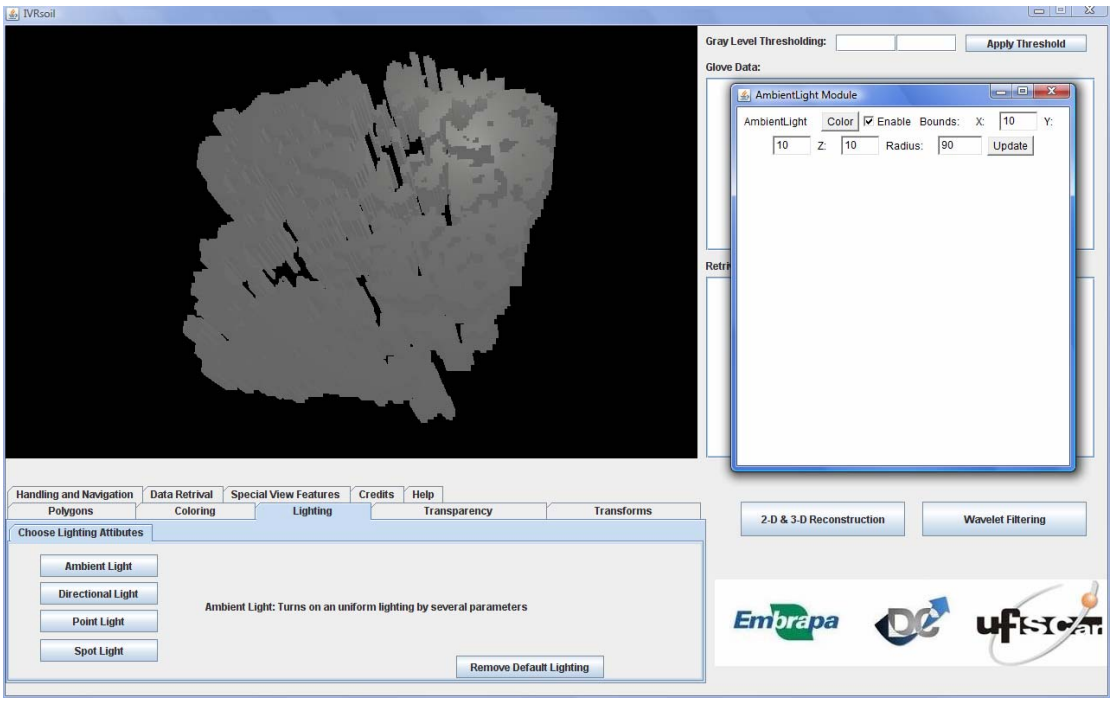


Figura 100 – Resultado da aplicação de fonte de luz Ambiente sobre uma amostra de cimento com os parâmetros: Fronteiras = (10 ,10 ,10); Raio = 90.

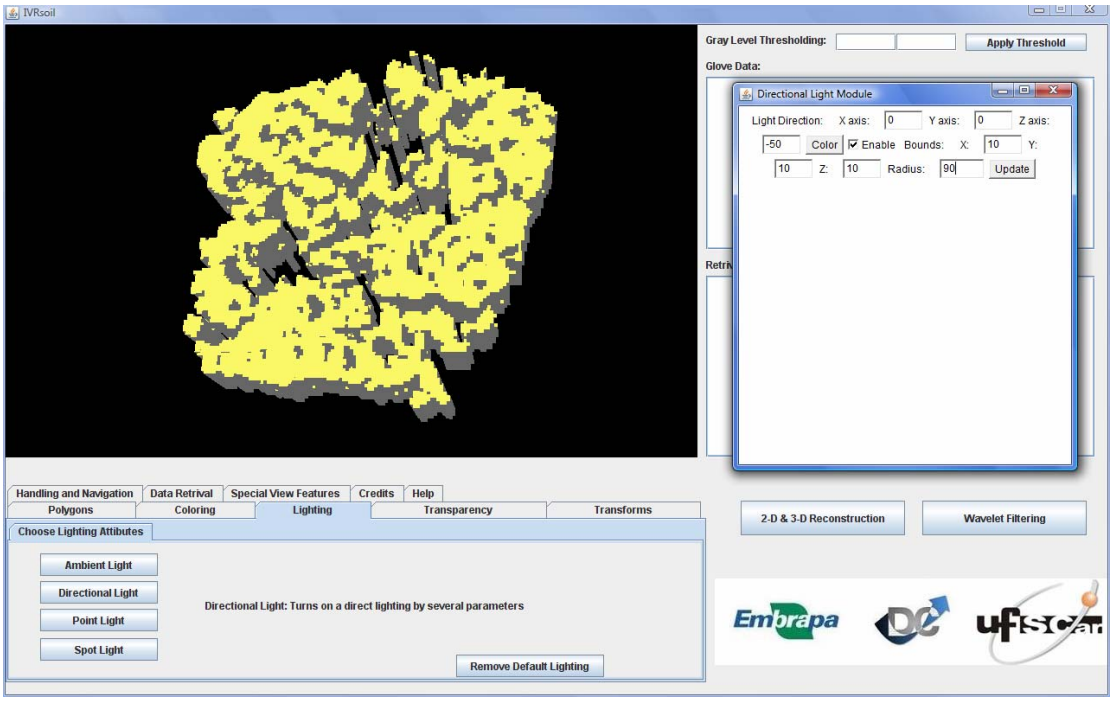


Figura 101 – Resultado da aplicação de fonte de luz Direcional sobre uma amostra de cimento com os parâmetros: Fronteiras = (10, 10, 10); Raio = 90; Direção = (0, 0, -50).

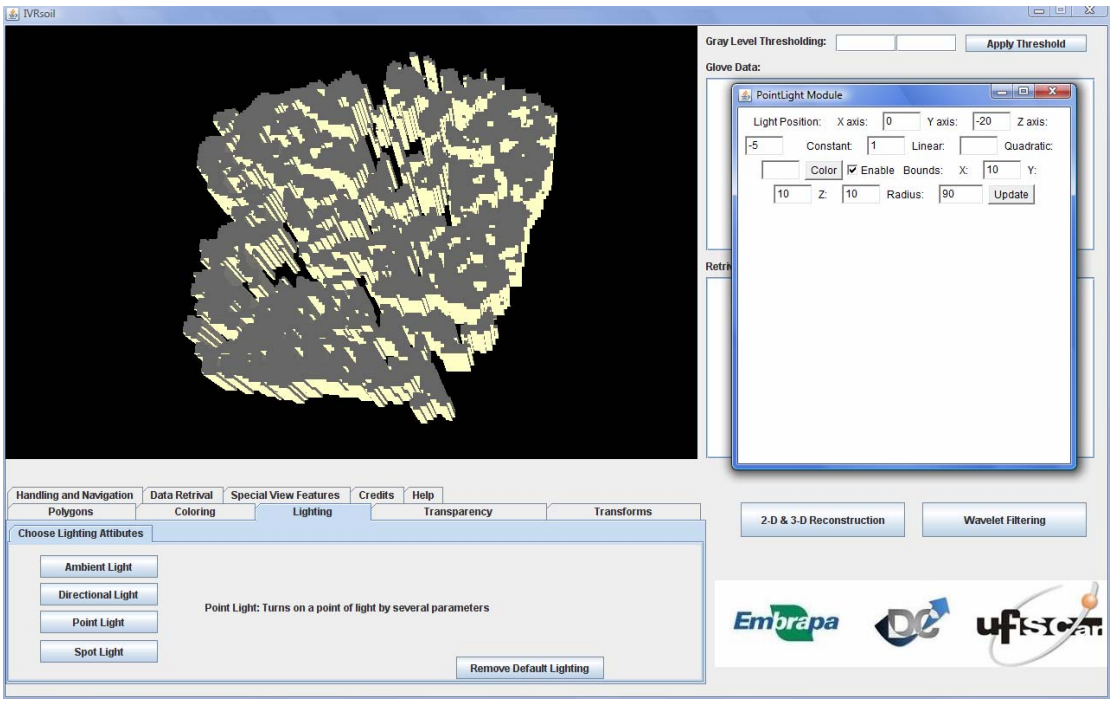


Figura 102 – Resultado da aplicação de fonte de luz Pontual sobre uma amostra de cimento com os parâmetros: Fronteiras = (10, 10, 10); Raio = 90; Posição = (0, -20, -5); Constante de atenuação = 1.

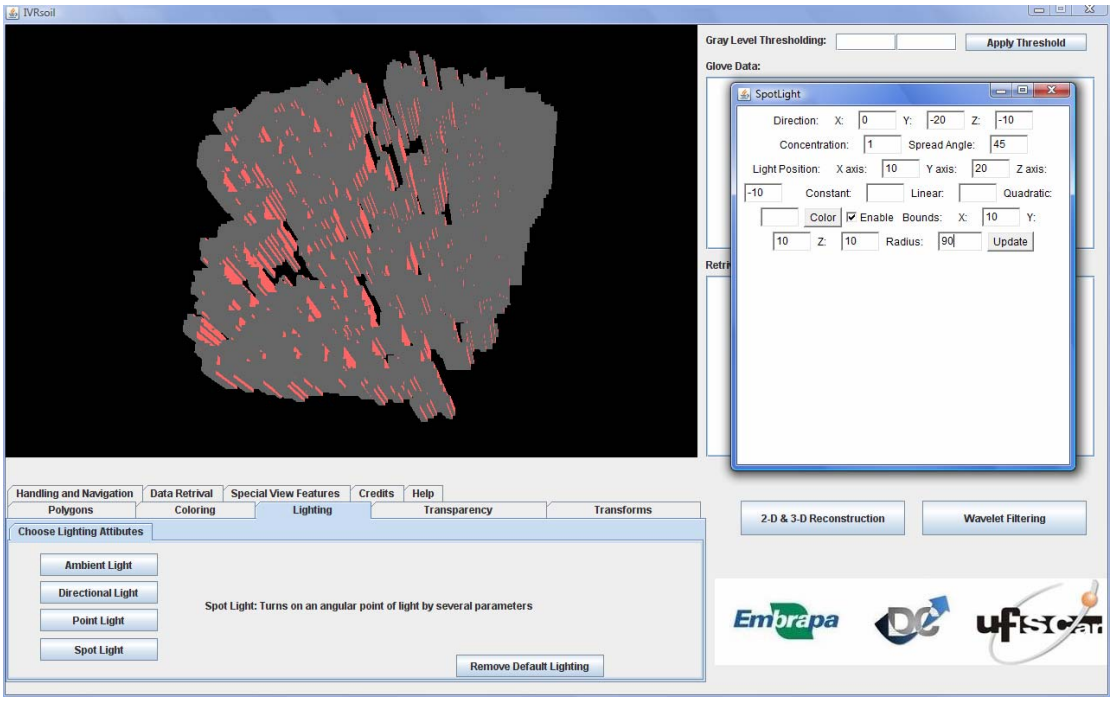


Figura 103 – Resultado da aplicação de fonte de luz Angular sobre uma amostra de solo degradado com os parâmetros: Fronteiras =(10, 10, 10); Raio = 90; Direção = (0, -20, -10); Concentração = 1; Ângulo de espalhamento = 45; Posição = (10, 20, -10); .

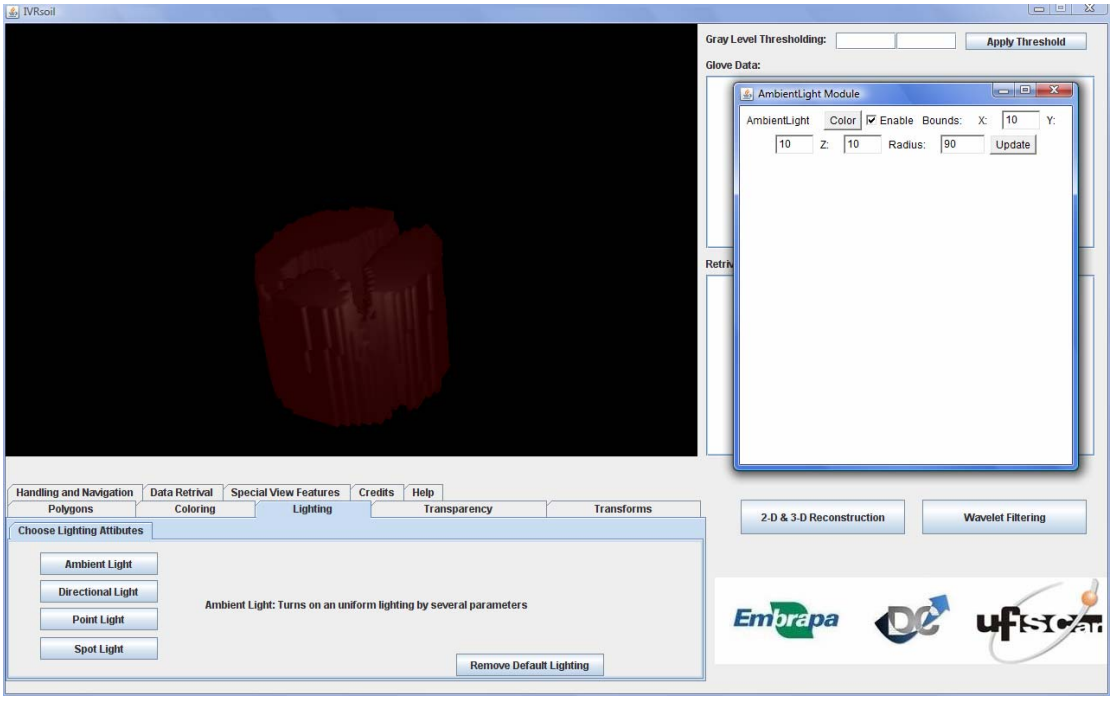


Figura 104 – Resultado da aplicação de fonte de luz Ambiente sobre uma amostra de solo com os parâmetros: Fronteiras = (10, 10, 10); Raio = 90.

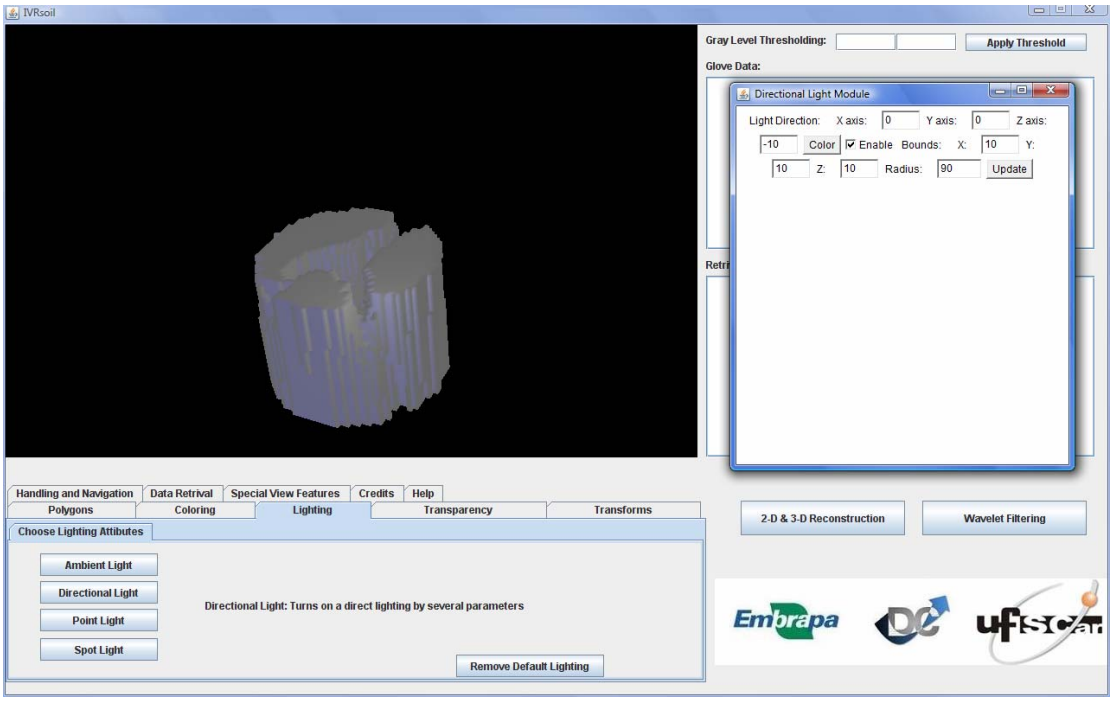


Figura 105 – Resultado da aplicação de fonte de luz Direcional sobre uma amostra de solo com os parâmetros: Fronteiras = (10, 10, 10); Raio = 90; Direção = (0, 0, -10).

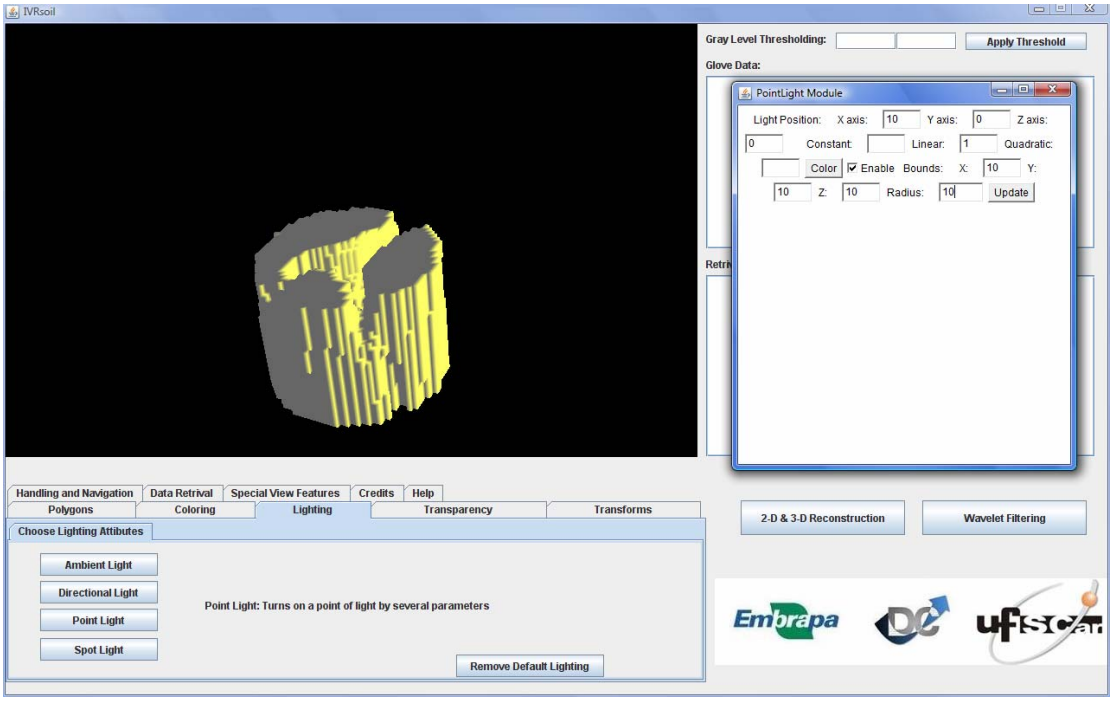


Figura 106 – Resultado da aplicação de fonte de luz Pontual sobre uma amostra de solo com os parâmetros: Fronteiras = (10, 10, 10); Raio = 10; Posição = (10, 0, 0); Atenuação Linear = 1.

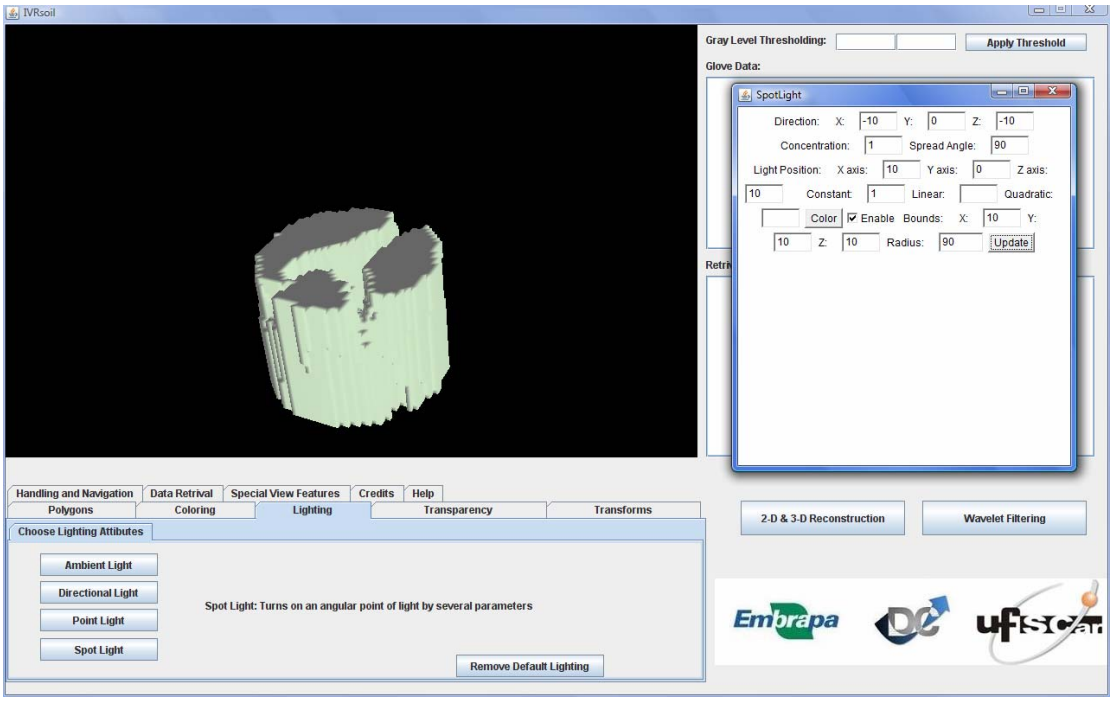


Figura 107 – Resultado da aplicação de fonte de luz Angular sobre uma amostra de solo com os parâmetros: Fronteiras=(10,10,10); Raio=90; Direção=(-10,0,-10); Concentração=1; Ângulo de espalhamento=90; Posição=(10, 0, 10); Atenuação constante=1.

5.9. Colorização

A classe *Colorização* possibilitou ampliar a interatividade do sistema, uma vez que viabiliza ao usuário, a troca da aparência das amostras tridimensionais em tempo de execução, atingindo *feedbacks* visuais. Provida pela interface principal do Ambiente de Realidade Virtual, a paleta de cores permitem que sejam escolhidas as cores desejadas e que sejam atribuídas sob uma determinada região escolhida da amostra, onde a aparência do *voxel* escolhido é então substituída pela nova cor selecionada. As Figuras 108 e 109 ilustram resultados da colorização de amostras como um todo. A Figura 110 ilustra o resultado de um processo de troca de aparência com base em classes específicas de coeficientes de atenuação linear (em cm^{-1}) de imagens tridimensionais.

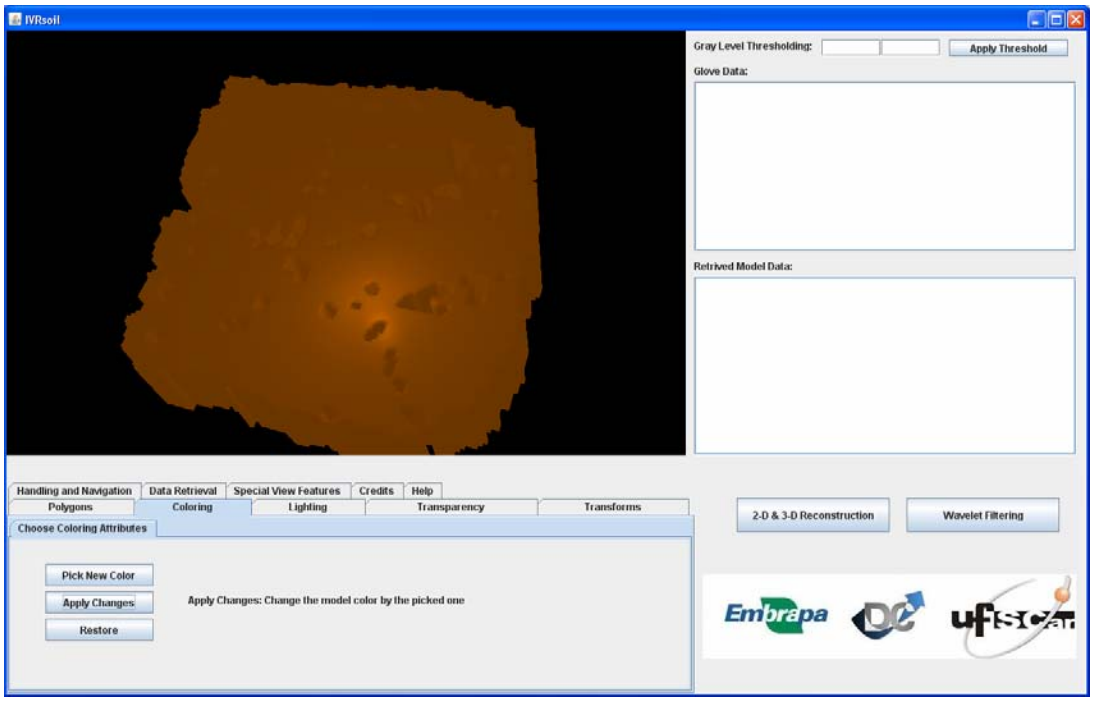


Figura 108- Resultado da aplicação da classe Colorização sobre amostra de solo degradado sob os coeficientes RGB = (147, 59, 41), aplicados em sua totalidade.

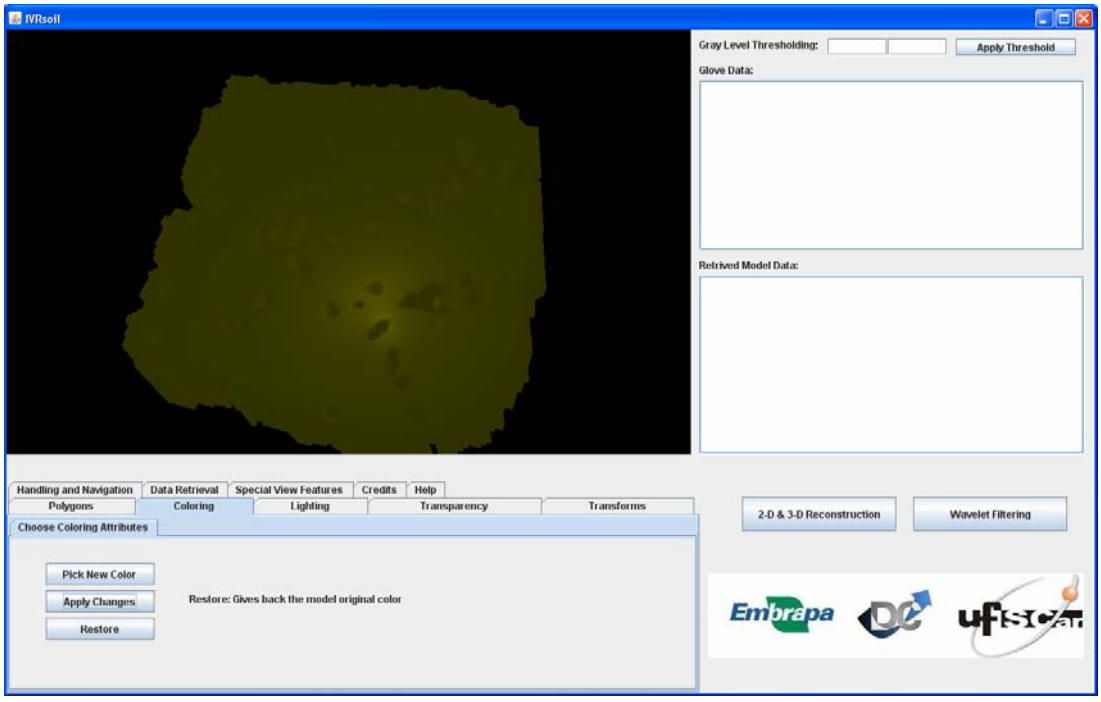


Figura 109 – Resultados da aplicação da classe Colorização sobre amostra de solo degradado sob os coeficientes RGB = (92, 113, 59), aplicados em sua totalidade.

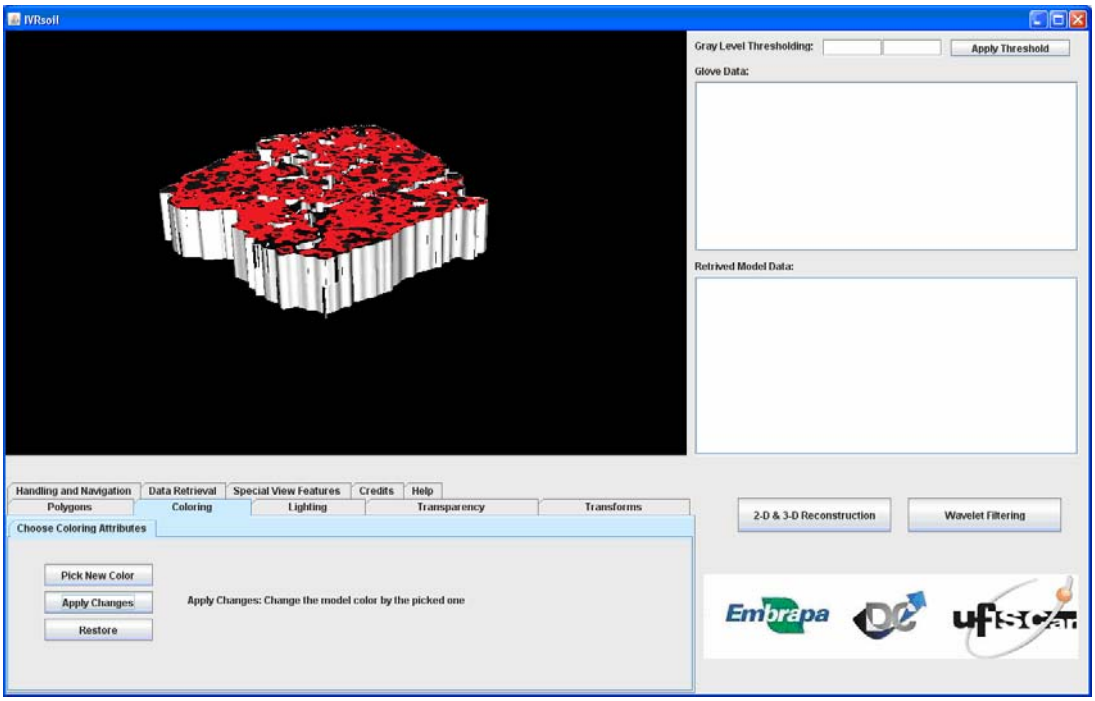


Figura 110 – Resultado da aplicação da classe Colorização sob os coeficientes RGB = (248, 6, 6), sobre apenas uma determinada região da amostra de solo degradado, definida por seus tons de cinza

5.10. Manipulação Convencional de Cena e Manipulação Convencional de Modelo

A classe *Manipulação Convencional de Cena*, responsável por realizar translações e rotações da cena virtual e todo seu contexto mediante movimentação de mouse e teclado, foi analisada, considerando a relação das entradas desejadas e as saídas obtidas. Desta maneira, as câmeras da cena virtual foram deslocadas para posições arbitrárias do *Canvas3D* utilizando dispositivos convencionais de entrada de dados, provendo assim a possibilidade de navegação. Pode-se constatar que o usuário pode explorar o ambiente virtual como se o mesmo estivesse presente no campo sintético, se aproximando e distanciando de determinada região de interesse, porém, de forma simulada, já que os movimentos do usuário se restringem ao apertar de teclas.

De forma análoga, a *Manipulação Convencional de Modelo* proporcionou a alteração das matrizes de transformação de forma interativa, ao contrário da solicitação de parâmetros da classe de *Transformações Físicas Lineares*.

5.11. Manipulação Não Convencional de Cena, Manipulação Não Convencional de Modelo, Quatérnios e Filtro

Em relação à *Manipulação Não Convencional de Cena*, os *feedbacks* produzidos pela implementação da classe indicam o total controle da navegação do usuário na cena, utilizando-se da luva *P5Glove*. Neste contexto, de acordo com o posicionamento do dispositivo, o usuário pode navegar efetivamente pela cena, onde os deslocamentos de sua mão são fielmente traduzidos em movimentos da cena, deslocando as câmeras do ambiente em tempo real. De forma análoga, tais movimentos também são traduzidos em deslocamentos das amostras tridimensionais. Tais movimentos são iniciados a partir do fechar das mãos, onde os sensores de dobra dos dedos são ativados e a ação iniciada. A classe de *Manipulação Não Convencional de Modelo* simula a sustentação manual das amostras 3-D, bem como sua total movimentação dentro da cena, também sob 6 graus de liberdade.

Atuando em paralelo, a classe *Filtro* também viabilizou a leitura armazenada a cada movimento e os novos valores calculados para o posicionamento de cada *LED*. Para cada novo posicionamento de cada um dos oito *LED's* emissores de luz, o algoritmo implementado tomou uma série de amostras, calculou a variância e adotou para a luva o valor mais adequado, ou seja, aquele que apresentou menor distância em relação à medida de variância. Tal processo impediu a oscilação de leituras e garantiu a cada *LED* um valor fixo, sustentando a cena e as amostras dentro do *Canvas3D*, já que a movimentação dos mesmos está atrelada aos movimentos do dispositivo. A Tabela 1 apresenta dados de leituras obtidas e novos valores atribuídos a cada um dos oito *LED's*, justificando sua utilização.

	Amostras do *LED1*	**Média**	**Variância**	**Novo valor**
X	*5,77, 5,23, 5,55, 5,14, 5,41*	*5,42*	*0,0635*	*5,41*
Y	*1,63, 1,78, 1,39, 1,45, 1,50*	*1,55*	*0,0244*	*1,50*
Z	*2,11, 2,16, 2,18, 2,19, 2,17*	*2,162*	*0,0010*	*2,16*
	Amostras do *LED2*	**Média**	**Variância**	**Novo valor**
X	*3,33, 3,31, 3,47, 3,39, 3,40*	*3,38*	*0,0040*	*3,39*
Y	*0,12, 0,28, 0,19, 0,20, 0,13*	*0,184*	*0,0041*	*0,19*
Z	*2,31, 2,28, 2,34, 2,25, 2,33*	*2,302*	*0,0014*	*2,31*

	Amostras do *LED3*	**Média**	**Variância**	**Novo valor**
X	*6,44, 6,48, 6,52, 6,50, 6,46*	*6,48*	*0,001*	*6,48*
Y	*0,34, 0,41, 0,38, 0,35, 0,47*	*0,39*	*0,0027*	*0,38*
Z	*2,14, 2,21, 2,19, 2,16, 2,24*	*2,188*	*0,0016*	*2,19*
	Amostras do *LED4*	**Média**	**Variância**	**Novo valor**
X	*6,70, 6,66, 6,73, 6,77, 6,68*	*6,708*	*0,0019*	*6,70*
Y	*0,53, 0,56, 0,61, 0,59, 0,62*	*0,582*	*0,0014*	*0,59*
Z	*6,42, 6,39, 6,46, 6,49, 6,38*	*6,428*	*0,0022*	*6,42*
	Amostras do *LED5*	**Média**	**Variância**	**Novo valor**
X	*6,22, 6,20, 6,29, 6,26, 6,31*	*6,256*	*0,0021*	*6,26*
Y	*0,10, 0,18, 0,11, 0,06, 0,05*	*0,1*	*0,0027*	*0,10*
Z	*6,91, 6,96, 6,98, 6,93, 6,97*	*6,95*	*0,0009*	*6,96*
	Amostras do *LED6*	**Média**	**Variância**	**Novo valor**
X	*4,77, 4,23, 4,55, 4,14, 4,41*	*4,42*	*0,0635*	*4,41*
Y	*2,63, 2,78, 2,39, 2,45, 2,50*	*2,55*	*0,0243*	*2,50*
Z	*3,91, 3,96, 3,88, 3,99, 3,87*	*3,922*	*0,0027*	*3,91*
	Amostras do *LED7*	**Média**	**Variância**	**Novo valor**
X	*6,19, 6,23, 6,22, 6,14, 6,11*	*6,178*	*0,0027*	*6,19*
Y	*0,63, 0,78, 0,39, 0,45, 0,50*	*0,55*	*0,0244*	*0,50*
Z	*6,02, 6,06, 6,01, 6,03, 6,09*	*6,042*	*0,0011*	*6,03*
	Amostras do *LED8*	**Média**	**Variância**	**Novo valor**
X	*3,70, 3,66, 3,61, 3,69, 3,63*	*3,658*	*0,0015*	*3,66*
Y	*0,13, 0,18, 0,12, 0,15, 0,10*	*0,136*	*0,0010*	*0,13*
Z	*6,79, 6,71, 6,73, 6,77, 6,75*	*6,75*	*0,001*	*6,75*

Tabela 1 – Representação das amostras de valores lidos pelo dispositivo, média dos mesmos, variância e novos valores adotados para cada eixo de cada um dos oito *LED's* da *P5Glove*.

A classe *Quatérnios*, quando avaliada apresentou com precisão a conversão do sistema de *Euler* para as coordenadas em quatérnios, uma vez que dadas as coordenadas do eixo aleatório a ser rotacionado e o ângulo desejado. O resultado da aplicação do método *quat4f*, de *Transform3D* , produziu mudanças de orientação suavizadas, ao contrário da primeira abordagem com eixos fixos, onde a ausência de

eixos intermediários e a limitação do numero de *LED's* interferia no fluxo natural das rotações. A Tabela 2 e a Figura 111 apresentam o resultado de uma rotação de 180º , considerando a posição inicial da *LED* = (-1,0, 1,0, 0,0), sendo a origem dos eixos (0, 0, 0) e a posição inicial da amostra em um ponto específico = (0, 0, 0). A Tabela 3 e a Figura 112 apresentam os resultados de uma rotação de 180º sobre um eixo estimado, sendo a posição inicial do *LED* = (1,0, 1,0, 0,0), partindo da origem dos eixos (0, 0, 0) e posição da amostra em um ponto específico = (0, 0, 0). A Tabela 4 e a Figura 113 apresentam os resultados de uma rotação de 180º sobre um eixo estimado, sendo a posição inicial do *LED* = (0,0, 1,0, 0,0), partindo da origem dos eixos (0, 0, 0) e a posição inicial da amostra em um ponto específico = (0, 0, 0).

Posição Inicial (ponto da amostra)	**Quatérnios**	**Posição Final** (ponto da amostra)
(0,0, 0,0, 0,0)	vetor: (0,0, 0,0, 0,0), escalar: 0,9238	(-0,7071, 0,7071, 0,0)
(-0,7071, 0,7071, 0,0)	vetor: (-0,2705, 0,2705, 0,0), escalar: 0,9238	(-0,9999, 0,9999, 0,0)
(-0,9999, 0,9999, 0,0)	vetor: (-0,3826, 0,3826, 0,0), escalar: 0,9238	(-1,2928, 1,2928, 0,0)
(-1,2928, 1,2928, 0,0)	vetor: (-0,4947, 0,4947, 0,0), escalar: 0,9238	(-1,6862, 1,6862, 0,0)

Tabela 2 - Resultado de uma rotação de 180º, considerando a posição inicial da *LED* = (-1,0, 1,0, 0,0), sendo a origem dos eixos (0, 0, 0) e a posição inicial da amostra em um ponto específico = (0, 0, 0).

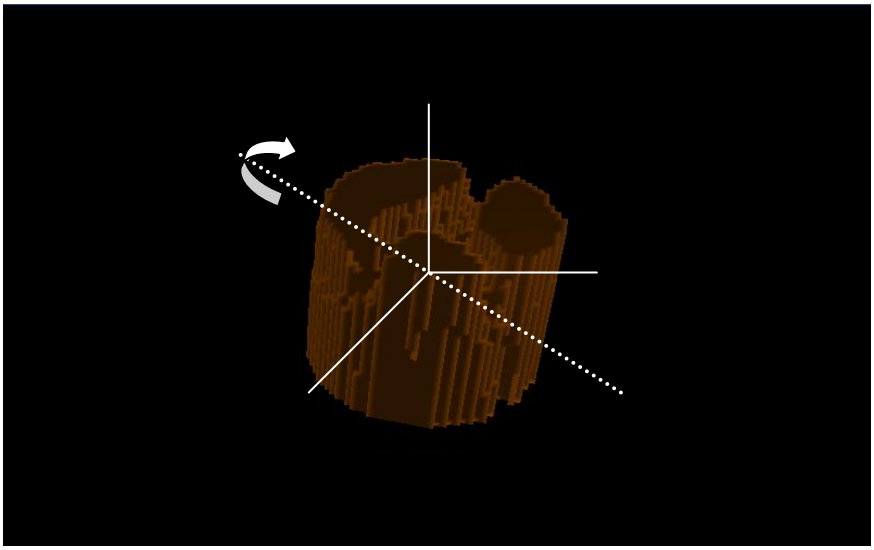


Figura 111 - Representação da rotação descrita em torno do eixo pontilhado definido pela coordenada da LED = (-1.0, 1.0, 0.0) passando pela origem dos eixos

Posição Inicial (ponto da amostra)	**Quatérnios**	**Posição Final** (ponto da amostra)
(0,0, 0,0, 0,0)	vetor: (0,0, 0,0, 0,0), escalar: 0,9238	(0,7071, 0,7071, 0,0)
(0,7071, 0,7071, 0,0)	vetor: (0,2705, 0,2705, 0,0), escalar: 0,9238	(0,9999, 0,9999, 0,0)
(0,9999, 0,9999, 0,0)	vetor: (0,3826, 0,3826, 0,0), escalar: 0,9238	(1,2928, 1,2928, 0,0)
(1,2928, 1,2928, 0,0)	vetor: (0,4947, 0,4947, 0,0), escalar: 0,9238	(1,6862, 1,6862, 0,0)

Tabela 3 - Resultado de uma rotação de 180º, considerando a posição inicial da *LED* = (1,0, 1,0, 0,0), sendo a origem dos eixos (0, 0, 0) e a posição inicial da amostra em um ponto específico = (0, 0, 0).

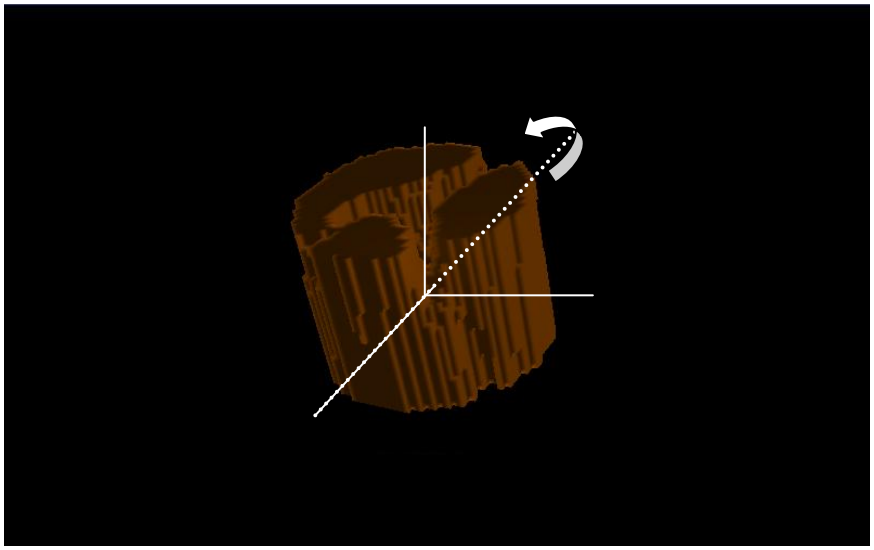


Figura 112 -Representação da rotação descrita em torno do eixo pontilhado definido pela coordenada da LED = (1.0, 1.0, 0.0) passando pela origem dos eixos

Posição Inicial (ponto da amostra)	**Quatérnios**	**Posição Final** (ponto da amostra)
(0,0, 0,0, 0,0)	vetor: (0,0, 0,0, 0,0), escalar: 0,9238	(0,0, 0,7071, 0,0)
(0,0, 0,7071, 0,0)	vetor: (0,0, 0,2705, 0,0), escalar: 0,9238	(0,0, 0,8535, 0,0)
(0,0, 0,8535, 0,0)	vetor: (0,0, 0,3266, 0,0), escalar: 0,9238	(0,0, 0,9204, 0,0)
(0,0, 0,9204, 0,0)	vetor: (0,0, 0,3522, 0,0), escalar: 0,9238	(0,0, 0,9552, 0,0)

Tabela 4 - Resultado de uma rotação de 180º, considerando a posição inicial da *LED* = (0,0, 1,0, 0,0), sendo a origem dos eixos (0, 0, 0) e a posição inicial da amostra em um ponto específico = (0, 0, 0).

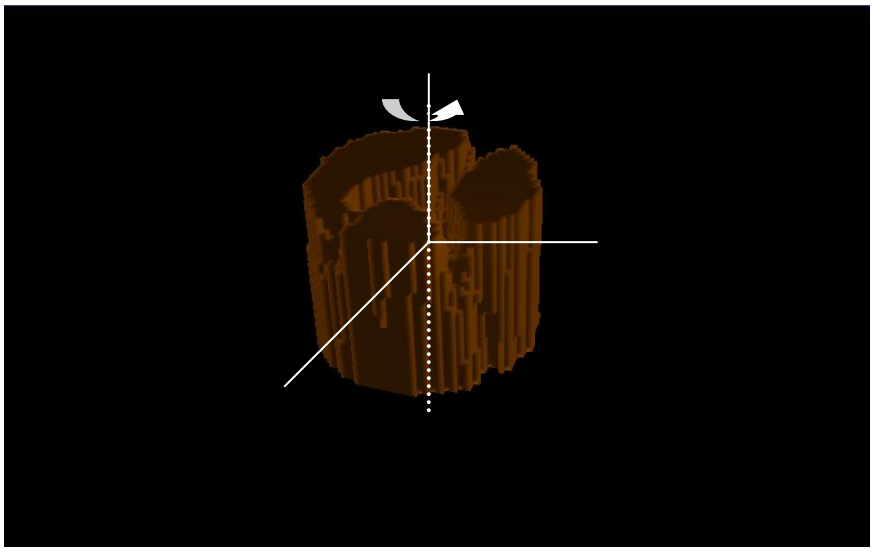


Figura 113 - Representação da rotação descrita em torno do eixo pontilhado definido pela coordenada da LED = (0.0, 1.0, 0.0) passando pela origem dos eixos

5.12. Visualização

Considerando a classe de *Visualização*, a mesma foi testada e avaliada ao que diz respeito à imersão do usuário via ambiente virtual nas imagens das amostras sob análise.

Através dos Anaglifos, usuários podem obter o efeito semi-imersivo utilizando apenas um óculos de lentes vermelha e azul, configurando-se como uma alternativa de baixo custo. Utilizando-se dos principios de estereoscopia, tais lentes filtram as respectivas visões do olho esquerdo e direito, fazendo com que cada olho veja somente uma imagem. Obtidas as duas perspectivas, o cérebro humano as fundir em apenas uma e gera uma imagem acinzentada com efeito de profundidade, como se a amostra estivesse saltando da tela, caracterizando uma paralaxe positiva, onde a imagem se forma na frente do *display*. A Figura 114 ilustra o resultado obtido com uma imagem tomográfica de uma amostra de adubo, sob visualização com anaglifo.

Por meio da opção de *Visualização* por Realidade Aumentada, a amostra tridimensional reconstruída pôde ser trazida ao ambiente real mantendo todas suas características originais, tais como forma, fronteiras, tamanho e atributos de aparência e material. Uma vez posicionado o marcador, de forma que possa ser totalmente rastreado pela *Webcam*, o modelo 3-D pôde também ser transladado e rotacionado, obedecendo o alcance e foco do dispositivo, configurando-se como alternativa viável

de visualização. A Figura 115 apresenta o resultado da aplicação da *Visualização* por Realidade Aumentada sobre uma amostra de vidro.

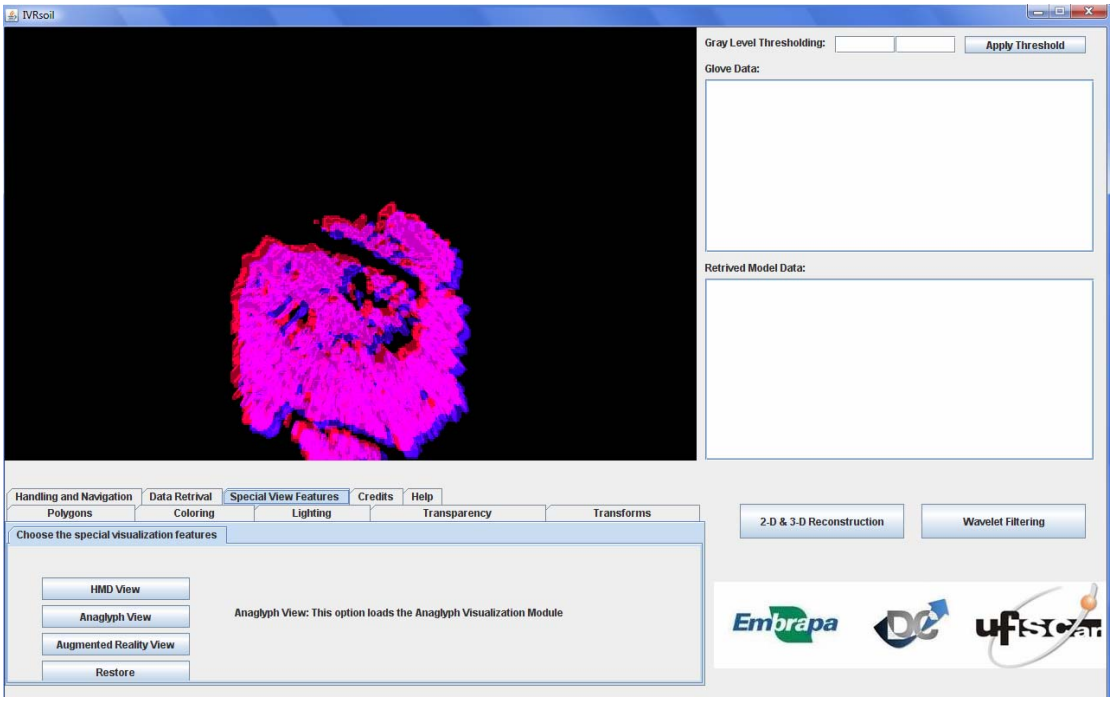


Figura 114 – Visualização por Anaglifo de uma imagem de amostra de adubo, onde a imagem final é composta pelas visões do olho esquerdo (vermelho) e olho direito (azul), as quais produzem o efeito imersivo ao serem visualizadas por lentes vermelha e azul.

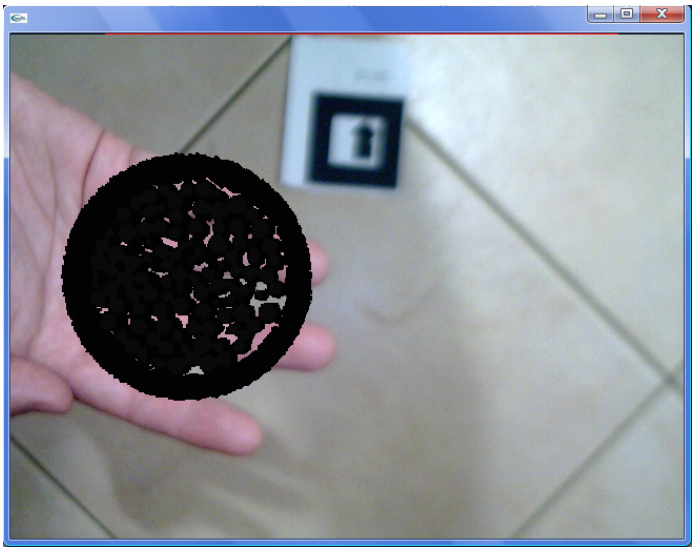


Figura 115 – Resultado da aplicação de Realidade Aumentada da classe Visualização sobre uma imagem tomográfica de amostra de vidro, onde a mesma é inserida no ambiente real através da identificação dos marcadores.

Também, através da classe de *Visualização*, as amostras tridimensionais puderam ser examinadas pelo *Head Mounted Display* de forma realmente imersiva. Inicialmente, o *Canvas3D*, responsável pela renderização das imagens tridimensionais, foi maximizado de forma a omitir no dispositivo as partes correspondentes da interface principal, a fim de focar apenas a região onde a amostra é exibida. Desta maneira, cada display do *HMD* forma uma imagem, as quais são fundidas pelo cérebro de forma análoga ao Anaglifo, também obedecendo às leis da estereoscopia. Como resultado, de forma mais satisfatória que o efeito produzido pelo Anaglifo, a imagem final se apresenta consideravelmente maior e com paralaxe mais significativa, ou seja, com maior efeito de profundidade. A Figura 116 apresenta uma simulação do efeito provocado pela utilização do *HMD* sobre uma amostra de areia, onde as visões dos olhos esquerdo e direito são renderizadas de forma independentes e fundidas pelo cérebro em apenas uma com tamanho e profundidade aumentados.

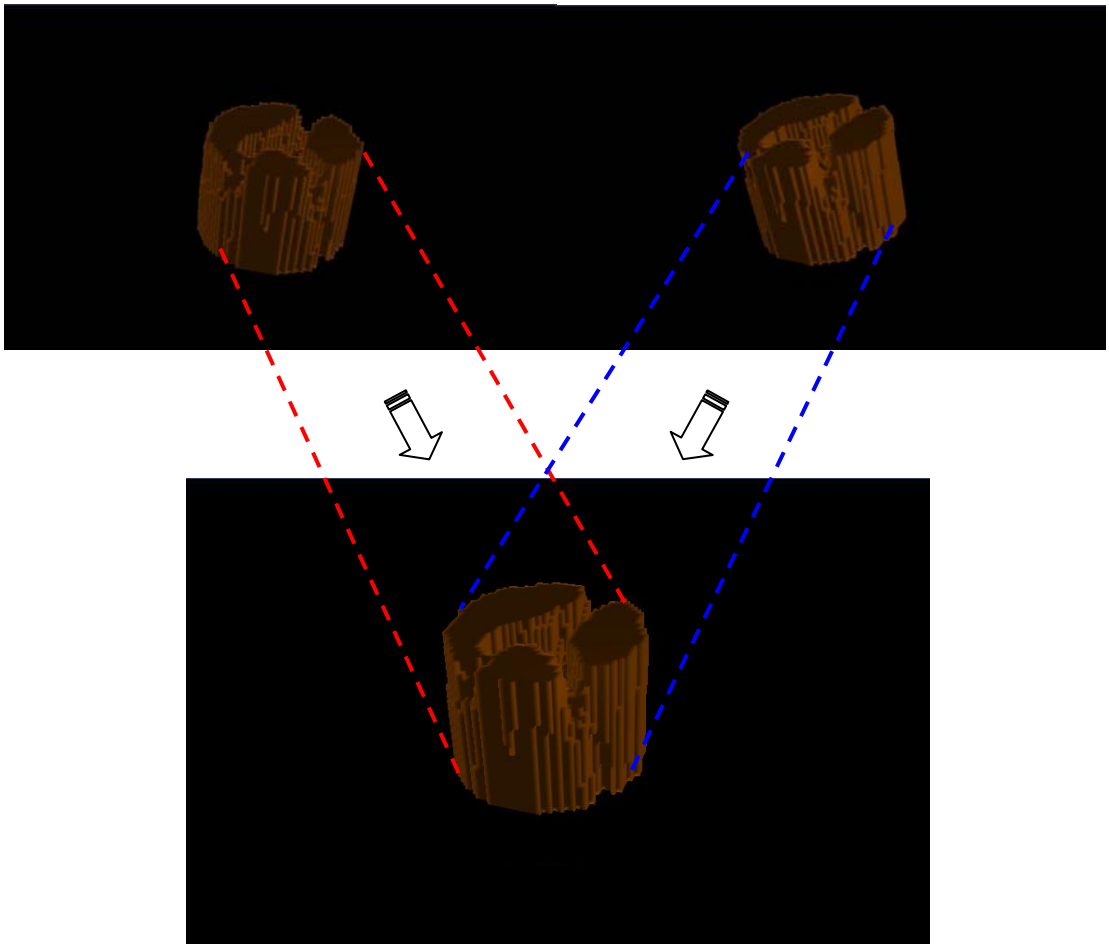


Figura 116 – Simulação do efeito provocado pelo uso do HMD para a visualização de amostras tridimensionais.

5.13. Estudo de Caso

Para justificar a implementação do sistema de visualização e análise de amostrais agrícolas tridimensionais da ciência de solos, o presente estudo abriga o estudo de casos em simulação da formação dos caminhos preferências de fluxo de água e solutos (*Fingering*) (Crestana e Posadas, 1998), e a aferição da porcentagem de volume de vazios das amostras tridimensionais, descritos em detalhes nesta seção.

Em Ciência de Solos, os poros (ou volume de vazios) configuram-se como os espaços vazios localizados no interior de uma determinada amostra agrícola, onde água, solutos e outros fluidos possam transpassar-la, no caso de poros conexos, ou simplesmente se acumularem em pequenas cavidades. Tais poros podem ser maiores ou menores, o que torna a amostra mais ou menos densa, o que influi diretamente na permeabilidade da mesma. Também podem estar preenchidos com água, ar ou ambos, forma mais encontrada na natureza (Crestana e Posadas, 1998). A porosidade do solo (*n)* é expressa em percentagem e definida como o volume de poros dividido pelo volume total de uma amostra de solo:

$$n = \frac{Vp}{Vt} \cdot 100 \qquad (5.1)$$

onde Vp é o volume de poros e Vt o volume total da amostra.

Na existência de poros conexos, formam-se os chamados caminhos preferenciais, ou seja, trajetos comuns e constantes a grande parte do fluxo de fluidos que atinge a amostra agrícola, formando estruturas identificáveis e estabelecidas, cujos formatos assemelham-se a dedos, origem do termo *fingering*. Para a simulação de tais eventos característicos da Ciência de Solos, foram utilizados partes de processos já implementados sobre o sistema de visualização e análise de amostras tridimensionais.

Conforme descrito nas classes *Manipulação Convencional de Cena* e *Manipulação Não Convencional de Cena*, à medida que as câmeras são deslocadas com os processos de navegação, ativados por interação de teclado ou da luva de dados *P5Glove*, os caminhos percorridos podem ser demarcados, deixando registrado o trajeto, sob forma visual e matemática.

A cada movimento de dispositivo identificado, tem-se uma nova posição da câmera sob coordenadas cartesianas (*x, y, z*). Tais posições são únicas e ocupadas somente uma por vez. Desta maneira, ativado o processo de demarcação, a partir de qualquer ponto, pode-se simular o caminho percorrido por um determinado fluxo de fluidos ao atingir uma amostra agrícola. Ao realizar um determinado movimento, o ponto atual ocupado pela câmera recebe um *Shape3D* sob a forma de uma esfera azul, a qual simula a presença de uma gota de fluido ocupando a posição outrora ocupada pela câmera, deixando um rastro azulado por onde a câmera passou.

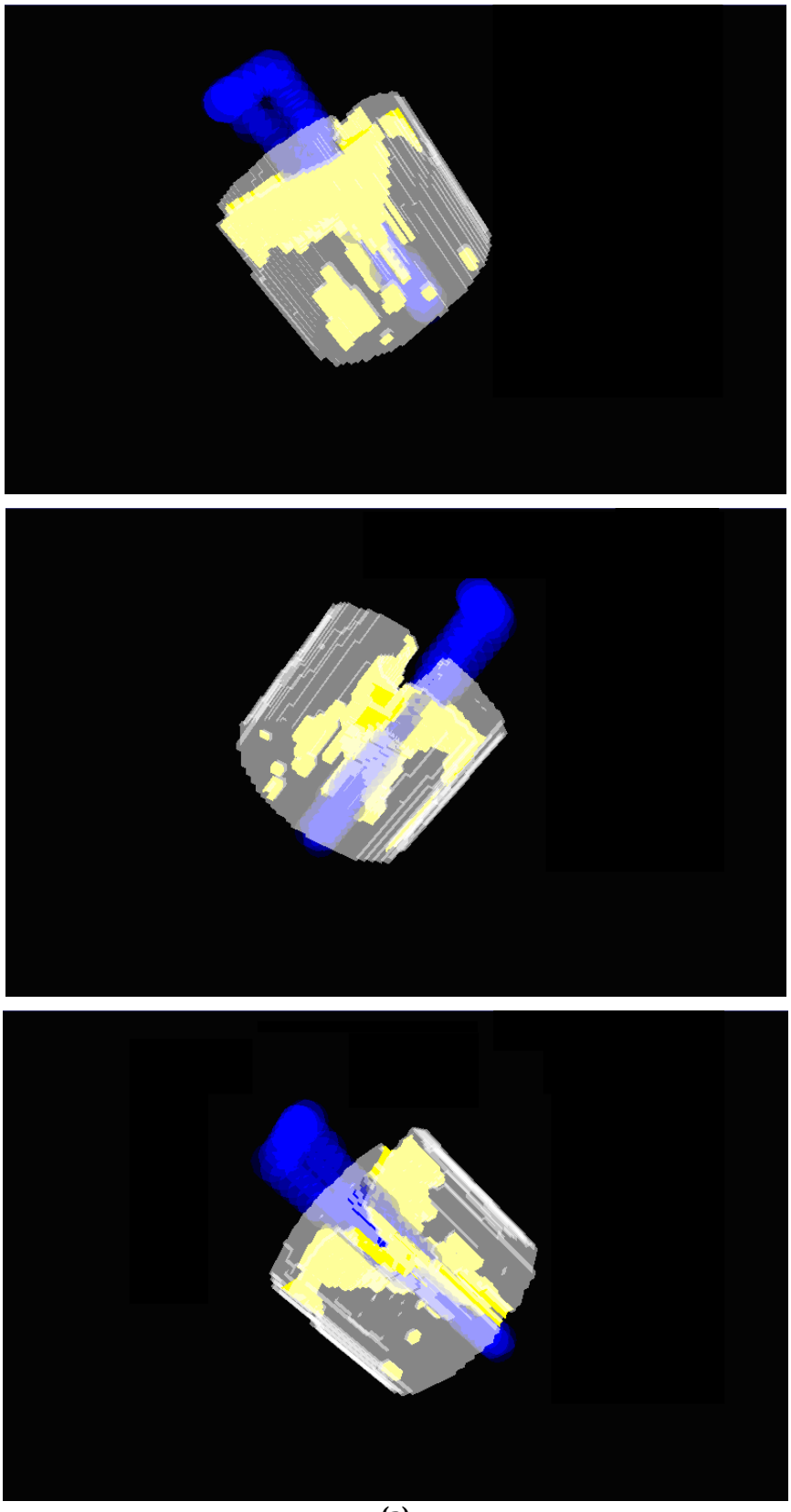
De forma análoga à simples manipulação de cena, tal demarcação obedece às leis impostas pelas classes *Colisão Convencional* e *Colisão Não Convencional*, ou seja, o caminho percorrido é impedido de transpassar as faces não porosas da amostra agrícola, obrigando o fluxo de fluidos a passar por dentre os poros conexos, os caminhos preferenciais. O processo pode ser repetido inúmeras vezes, simulando a entrada de diversos fluxos de fluidos, de forma semelhante às situações reais.

A partir do cálculo de fronteiras, onde os limites da amostra tridimensional são identificados no espaço, como na classe Extração de Atributos, através do uso de *getBounds* sobre instâncias *Shape3D*, aliado a um algoritmo de detecção de bordas tridimensionais chamado de *Polytope*, presente no pacote *Bounds* da *API Java3D*. Tal algoritmo encarrega-se de traçar inúmeros planos ao redor das superfícies da amostra, percorrendo toda sua extensão a fim de delimitar com exatidão suas fronteiras, permitindo que as partes não porosas da amostra, inclusive as internas, possam ser identificadas, permitindo a aferição de seu volume em cm^3.

Identificada a parte não porosa, as porções restantes, os vazios da amostra, as quais apresentam cor e material zero devido à ausência de atenuação de fótons, e que obedeçam os limites impostos pela parte não porosa, são então preenchidas com uma geometria amarela semi-transparente, visando um maior destaque junto à amostra. Tal geometria representa o total de poros, parte que se deseja calcular o volume em cm^3.

Com tais dados disponíveis, é possível calcular o volume total da amostra (soma das partes não porosas com seu complemento) em cm^3. Desta maneira, a partir do volume total e do volume individual da parte não porosa, é possível calcular de forma precisa o volume representado pelos vazios da amostra tridimensional. As Figuras 117 e 118 ilustram os resultados do estudo de caso em duas imagens de solo degradado onde ambas apresentam: (a) demarcação visual dos caminhos preferenciais

e (b) representação cartesiana dos caminhos preferências do fluxo de água e solutos, e a aferição da porosidade das amostras agrícolas por fronteiras.

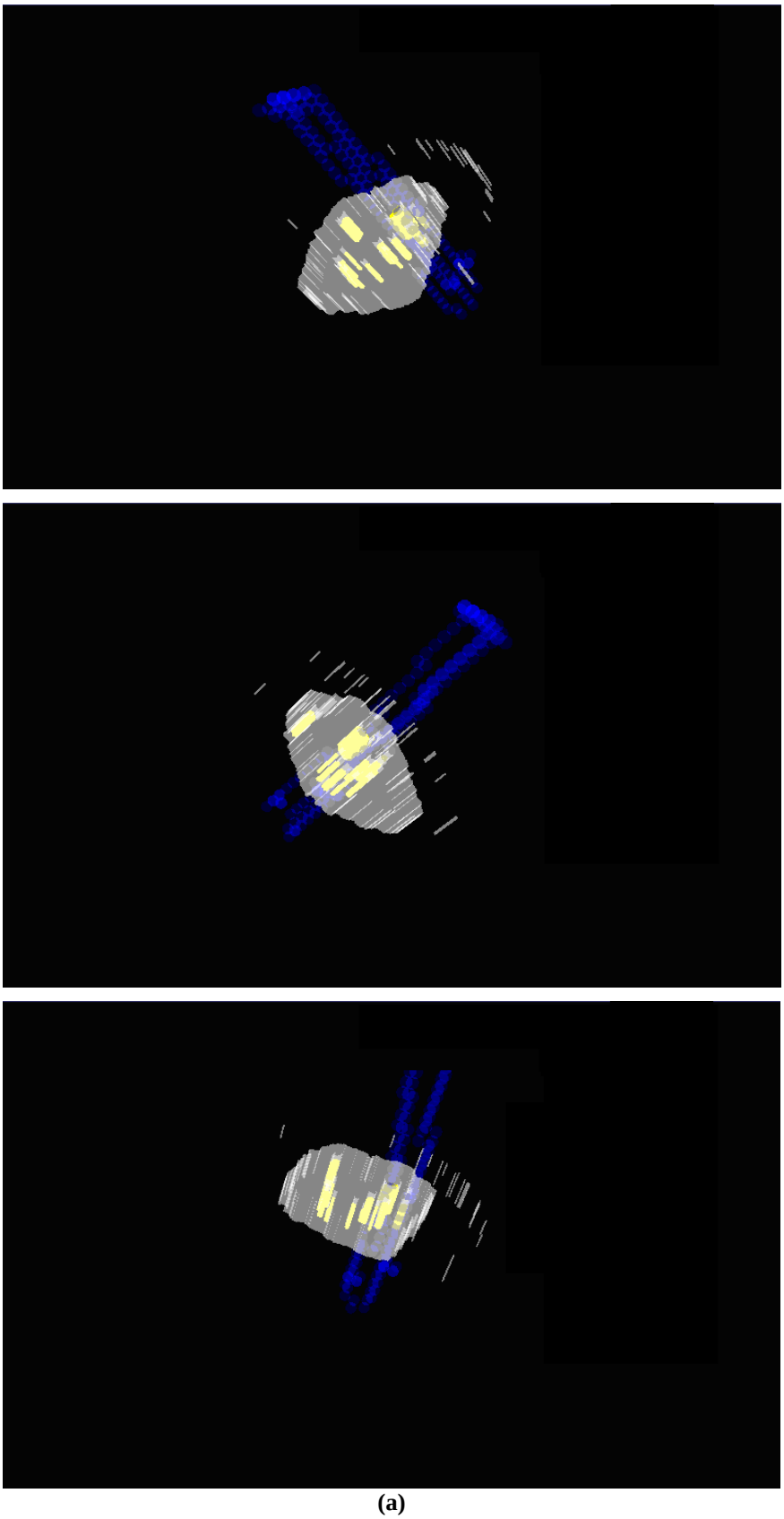


(a)

Volume total: 22,57 cm³ Volume de Poros: 9,67 cm³ Volume da Amostra: 12,90 cm³ Porcentagem de Poros: 42,84 % Porcentagem de Amostra: 57,16 % **Caminho 1:** Down (0,0, 0,91, 3,8) Right (0,09, 0,91, 3,8) Down (0,09, 0,82, 3,8) Right (0,18, 0,82, 3,8) Left (0,09, 0,82, 3,8) Forward (0,09, 0,82, 3,71) Forward (0,09, 0,82, 3,62) Forward (0,09, 0,82, 3,53) Down (0,09, 0,73, 3,53) Forward (0,09, 0,73, 3,44) Forward (0,09, 0,73, 3,35) Forward (0,09, 0,73, 3,26) Forward (0,09, 0,73, 3,17) Forward (0,09, 0,73, 3,08) Forward (0,09, 0,73, 2,99) Forward (0,09, 0,73, 2,90) Down (0,09, 0,64, 2,90) Forward (0,09, 0,64, 2,81) Forward (0,09, 0,64, 2,72) Forward (0,09, 0,64, 2,63)	Forward (0,09, 0,64, 2,54) Forward (0,09, 0,64, 2,45) Forward (0,09, 0,64, 2,36) Down (0,09, 0,55, 2,36) Forward (0,09, 0,55, 2,27) Forward (0,09, 0,55, 2,18) Forward (0,09, 0,55, 2,09) Forward (0,09, 0,55, 2,00) Forward (0,09, 0,55, 1,91) Forward (0,09, 0,55, 1,82) Down (0,09, 0,46, 1,82) Forward (0,09, 0,46, 1,73) **Caminho 2:** Down (0,0, 0,91, 3,8) Down (0,0, 0,82, 3,8) Down (0,0, 0,73, 3,8) Down (0,0, 0,64, 3,8) Down (0,0, 0,55, 3,8) Down (0,0, 0,46, 3,8) Left (-0,09, 0,46, 3,8) Forward (-0,09, 0,46, 3,71) Forward (-0,09, 0,46, 3,62) Forward (-0,09, 0,46, 3,53) Forward (-0,09, 0,46, 3,44) Forward (-0,09, 0,46, 3,35) Forward (-0,09, 0,46, 3,26) Right (0,0, 0,46, 3,26)	Forward (0,0, 0,46, 3,17) Forward (0,0, 0,46, 3,08) Forward (0,0, 0,46, 2,99) Forward (0,0, 0,46, 2,90) Forward (0,0, 0,46, 2,81) Forward (0,0, 0,46, 2,72) UP (0,0, 0,55, 2,72) Down (0,0, 0,46, 2,72) Forward (0,0, 0,46, 2,63) Forward (0,0, 0,46, 2,54) Forward (0,0, 0,46, 2,45) UP (0,0, 0,55, 2,45) Forward (0,0, 0,55, 2,36) Forward (0,0, 0,55, 2,27) Forward (0,0, 0,55, 2,18) Forward (0,0, 0,55, 2,09) Forward (0,0, 0,55, 2,00) Forward (0,0, 0,55, 1,91) Right (0,09, 0,55, 1,91) Forward (0,09, 0,55, 1,82) Forward (0,09, 0,55, 1,73) Forward (0,09, 0,55, 1,64) **Caminho 3:** Down (0,0, 0,91, 3,8) Down (0,0, 0,82, 3,8) Down (0,0, 0,73, 3,8) Right (0,09, 0,73, 3,8)	Down (0,09, 0,64, 3,8) Down (0,09, 0,55, 3,8) Down (0,09, 0,46, 3,8) Forward (0,09, 0,46, 3,71) Forward (0,09, 0,46, 3,62) Forward (0,09, 0,46, 3,53) Forward (0,09, 0,46, 3,44) Forward (0,09, 0,46, 3,35) Forward (0,09, 0,46, 3,26) Forward (0,09, 0,46, 3,17) UP (0,09, 0,55, 3,17) Forward (0,09, 0,55, 3,08) Forward (0,09, 0,50, 2,90) Forward (0,09, 0,55, 2,90) Forward (0,09, 0,55, 2,81) Forward (0,09, 0,55, 2,72) Forward (0,09, 0,55, 2,63) Forward (0,09, 0,55, 2,54) Left (0,0, 0,55, 2,54) Forward (0,0, 0,55, 2,45) Forward (0,0, 0,55, 2,36) Forward (0,0, 0,55, 2,27) Forward (0,0, 0,55, 2,18) Forward (0,0, 0,55, 2,09) Forward (0,0, 0,55, 2,00) Forward (0,0, 0,55, 1,91) UP (0,0, 0,64, 1,91)

(b)

Figura 117 – Resultado do estudo de caso em uma imagem de solo degradado em: (a) demarcação visual, sob diversas orientações e translações, dos caminhos preferenciais, onde a amostra encontra-se em tons de cinza, os vazios ou poros encontram-se em amarelo e o fluxo de água em azul, demarcando os caminhos percorridos e (b) representação cartesiana dos caminhos preferências do fluxo de água e solutos, e a aferição da porosidade das amostras agrícolas por fronteiras.



(a)

Volume total:
10,13 cm³
Volume de Poros:
1,80 cm³
Volume da Amostra:
8,32 cm³
Porcentagem de Poros:
17,80 %
Porcentagem de Amostra:
82,19 %

Caminho 1:
Down
(0,0, 0,91, 3,8)
Down
(0,0, 0,82, 3,8)
Down
(0,0, 0,73, 3,8)
Down
(0,0, 0,64, 3,8)
Down
(0,0, 0,55, 3,8)
Forward
(0,0, 0,55, 3,71)
Forward
(0,0, 0,55, 3,62)
Forward
(0,0, 0,55, 3,53)
Forward
(0,0, 0,55, 3,44)
Forward
(0,0, 0,55, 3,35)
Forward
(0,0, 0,55, 3,26)
Forward
(0,0, 0,55, 3,17)
Forward
(0,0, 0,55, 3,08)
Down
(0,0, 0,46, 3,08)
Forward
(0,0, 0,46, 2,99)
Forward
(0,0, 0,46, 2,90)
Forward
(0,0, 0,46, 2,81)
Forward
(0,0, 0,46, 2,72)
Forward
(0,0, 0,46, 2,63)
Forward
(0,0, 0,46, 2,54)
Forward
(0,0, 0,46, 2,45)
Right
(0,09, 0,46, 2,45)
Left
(0,0, 0,46, 2,45)
Forward
(0,0, 0,46, 2,36)
Forward
(0,0, 0,46, 2,27)
Forward
(0,0, 0,46, 2,18)
Forward
(0,0, 0,46, 2,09)
Forward
(0,0, 0,46, 2,00)
Forward
(0,0, 0,46, 1,91)
Forward
(0,0, 0,46, 1,82)
Forward
(0,0, 0,46, 1,73)
Right
(0,09, 0,46, 1,73)
Left
(0,0, 0,46, 1,73)
Down
(0,0, 0,37, 1,73)
Right
(0,09, 0,37, 1,73)
Left
(0,0, 0,37, 1,73)
UP
(0,0, 0,46, 1,73)
Forward
(0,0, 0,46, 1,64)
Caminho 2:
Left
(-0,09, 1,0, 3,8)
Down
(-0,09, 0,91, 3,8)
Down
(-0,09, 0,82, 3,8)
Left
(-0,18, 0,82, 3,8)
Right
(-0,09, 0,82, 3,8)
Right
(0,0, 0,82, 3,8)
Forward
(0,0, 0,82, 3,71)
Forward
(0,0, 0,82, 3,62)
Forward
(0,0, 0,82, 3,53)
Forward
(0,0, 0,82, 3,44)
Forward
(0,0, 0,82, 3,35)
Forward
(0,0, 0,82, 3,26)
Down
(0,0, 0,73, 3,26)
Forward
(0,0, 0,73, 3,17)
Forward
(0,0, 0,73, 3,08)
Forward
(0,0, 0,73, 2,99)
Forward
(0,0, 0,73, 2,90)
Forward
(0,0, 0,73, 2,81)
Forward
(0,0, 0,73, 2,72)
Down
(0,0, 0,64, 2,72)
Forward
(0,0, 0,64, 2,63)
Forward
(0,0, 0,64, 2,54)
Forward
(0,0, 0,64, 2,45)
Forward
(0,0, 0,64, 2,36)
Forward
(0,0, 0,64, 2,27)
Forward
(0,0, 0,64, 2,18)
Forward
(0,0, 0,64, 2,09)
Right
(0,09, 0,64, 2,09)
Left
(0,0, 0,64, 2,09)
UP
(0,0, 0,73, 2,09)
Forward
(0,0, 0,73, 2,00)
Forward
(0,0, 0,73, 1,91)
Forward
(0,0, 0,73, 1,82)
Forward
(0,0, 0,73, 1,73)
Forward
(0,0, 0,73, 1,64)
Forward
(0,0, 0,73, 1,55)
Forward
(0,0, 0,73, 1,46)
Down
(0,0, 0,64, 1,46)
Right
(0,09, 0,64, 1,46)
Left
(0,0, 0,64, 1,46)
UP
(0,0, 0,73, 1,46)
Forward
(0,0, 0,73, 1,37)
Down
(0,0, 0,91, 3,8)
Down
(0,0, 0,82, 3,8)
Down
(0,0, 0,73, 3,8)
Left
(-0,09, 0,73, 3,8)
Left
(-0,18, 0,73, 3,8)

Caminho 3:
Down
(0,0, 0,91, 3,8)
Down
(0,0, 0,82, 3,8)
Down
(0,0, 0,73, 3,8)
Down
(0,0, 0,64, 3,8)
Left
(-0,09, 0,64, 3,8)
Left
(-0,18, 0,64, 3,8)
Down
(-0,18, 0,55, 3,8)
Right
(-0,09, 0,55, 3,8)
Forward
(-0,09, 0,55, 3,71)
Forward
(-0,09, 0,55, 3,62)
Forward
(-0,09, 0,55, 3,53)
Forward
(-0,09, 0,55, 3,44)
Forward
(-0,09, 0,55, 3,35)
Forward
(-0,09, 0,55, 3,26)
Forward
(-0,09, 0,55, 3,17)
Forward
(-0,09, 0,55, 3,08)

Forward
(-0,09, 0,55, 2,99)
Down
(-0,09, 0,46, 2,99)
Forward
(-0,09, 0,46, 2,90)
Forward
(-0,09, 0,46, 2,81)
Forward
(-0,09, 0,46, 2,72)
Forward
(-0,09, 0,46, 2,63)
Forward
(-0,09, 0,46, 2,54)
Left
(-0,18, 0,46, 2,54)
Forward
(-0,18, 0,46, 2,45)
Forward
(-0,18, 0,46, 2,36)
Forward
(-0,18, 0,46, 2,27)
Forward
(-0,18, 0,46, 2,18)
Forward
(-0,18, 0,46, 2,09)
Forward
(-0,18, 0,46, 2,00)
Forward
(-0,18, 0,46, 1,91)
Forward
(-0,18, 0,46, 1,82)
Forward
(-0,18, 0,46, 1,73)
Forward
(-0,18, 0,46, 1,64)

(b)

Figura 118 – Resultado do estudo de caso em uma imagem de solo degradado em: (a) demarcação visual, sob diversas orientações e translações, dos caminhos preferenciais, onde a amostra encontra-se em tons de cinza, os vazios ou poros encontram-se em amarelo e o fluxo de água em azul, demarcando os caminhos percorridos e (b) representação cartesiana dos caminhos preferências do fluxo de água e solutos, e a aferição da porosidade das amostras agrícolas por fronteiras.

Após todo o processo de simulação da formação dos caminhos preferenciais de água e solutos por dentre as regiões porosas das imagens das amostras tridimensionais, os dados das coordenadas cartesianas de cada medida puderam ser armazenados e recuperados utilizando-se de escrita e leitura de um arquivo texto, cujo conteúdo reproduz exatamente os dados descritos nas Figuras 117 (b) e 118 (b). Uma vez com estes dados novamente disponíveis, foi possível renderizar apenas os trajetos percorridos, formando um mapa tridimensional dos caminhos preferências. A Figura 119 ilustra tais caminhos sob diversas orientações e translações.

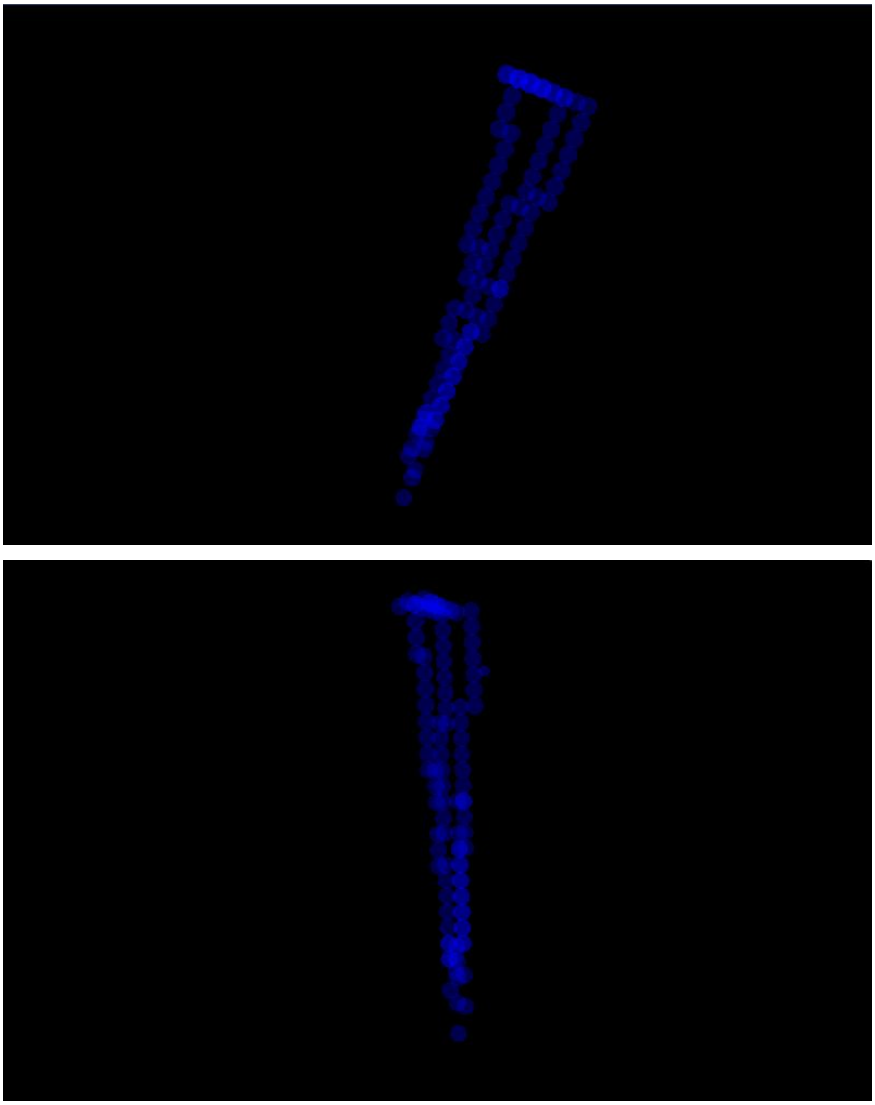


Figura 119 – Representação visual dos mapas de caminhos preferenciais do fluxo de água e solutos, após a recuperação dos dados armazenados no arquivo, mediante prévio estudo de caso.

5.14. Validação do Ambiente

Como validação, um grupo de usuário foi organizado para executar uma rotina de treinamento e execução de tarefas específicas para que todos os recursos disponíveis fossem compreendidos e utilizados. Assim, os usuários foram convidados a responderem um questionário para avaliar a qualidade do ambiente de Realidade Virtual, o qual se encontra no Anexo I.

Dentre o grupo de usuários que participaram da avaliação do ambiente, encontraram-se engenheiros, cientisdas da computação, agrônomos, biólogos e usuários não familiarizados com ambientes científicos computacionais.

Os temas abordados no questionário incluíram: Interface, composta pelas janelas, botões e acessos aos recursos; Customização de amostra, que compreende as técnicas computacionais gráficas para editar os modelos; Customização do ambiente, o qual tratou da edição da cena virtual; Obtenção de dados, que representa a aferição e exibição de dados da amostra; Detecção de colisão, que tratou das técnicas de bloqueio de movimentos ao se encontrar obstáculos; Imersão, ligada à transposição do usuário a outra realidade e Interação, que avaliou o processo de reposta do sistema às ações dos usuários.

Os resultados médios obtidos, em escala crescente de 1 a 6, foram:

1 – Quanto á interface: 6

2 – Quanto à customização de amostra: 6

3 – Quanto à customização do ambiente: 6

4 – Quanto à obtenção de dados: 6

5 – Quanto à detecção de colisão: 5

6 – Quanto à imersão: 5

7 – Quanto à interação: 6

No Capítulo VI são apresentadas as conclusões e sugestões para trabalhos futuros.

Capítulo VI
Conclusões

O presente trabalho apresentou a preparação, desenvolvimento e validação de um sistema de Realidade Virtual (RV) dedicado à inspeção de amostras tomográficas de solos agrícolas, utilizando dados de imagens importadas e reconstruídas em 3-D. Através de uma *interface* com o usuário, tais imagens foram então submetidas a processos de manipulação e visualização tridimensional, onde, por intermédio de recursos computacionais gráficos, o sistema buscou somar imersão e interação do usuário com as amostras. Tais recursos envolveram controle de renderização, iluminação, colorização, extração de atributos e transformações físicas, além da integração de dispositivos não convencionais de entrada e saída de dados, tais como um *Head-Mounted-Display* (vídeo-capacete) e uma luva digital.

Assim, considerando a fase de reconstrução de imagens tomográficas, verificou-se uma redução no nível de ruído das imagens através da filtragem com a transformada *Wavelet Doubechies* bidimensional, somadas às filtragens unidimensionais com janelas de *Hamming*. Tal fase permitiu que a qualidade do produto da reconstrução bidimensional fosse melhorada, fornecendo cortes mais fidedignos aos modelos reais ensaiados.

A classe de importação (*Loader*) apresentou os resultados esperados, considerando a complexidade computacional envolvida pela fusão de duas tecnologias diferentes, uma vez que os algoritmos originais de reconstrução produziam apenas modelos *Visualization Toolkit*. A partir do uso da classe desenvolvida, todos os recursos presentes em tal arquivo puderam ser acessados pelo ambiente programado em *Java3D*, tais como geometria, aparência e material, tornando simples a integração das amostras reconstruídas com os recursos do *Ambiente de Realidade Virtual*.

Adicinalmente, todas as características originais das amostras ensaiadas puderam ser preservadas ao se realizar tal conversão, permitindo que caraterísticas inerentes a uma imagem tomográfica sejam mantidas, tais como a escala de cores e opacidade. Com tais valores mantidos, novas medidas puderam ser obtidas diretamente no ambiente tridimensional programado em *Java3D*, dentre elas os coeficientes de atenuação linear e as medidas de porosidade, além de tornar o processo mais realista.

A respeito do desenvolvimento do *Ambiente de Realidade Virtual* e as demais ferramentas gráficas de visualização tais como os controles de *Polígonos*, *Transparência*, *Colorização, Extração de Atributos* e *Iluminação*, as mesmas apresentaram *feedbacks* visuais adequados, essenciais para prover uma melhor qualidade das imagens e contribuir para processos de análise. Por meio de tais classes, os modelos tridimensionais e a cena virtual puderam ter seu contexto customizado, adequando os mesmos às necessidades do utilizador no que diz respeito à medidas estruturais e melhor visualização da amostra envolvida no processo.

Paralelamente, a implementação de *Transformações*, *Manipulações*, *Colisão e Visualização,* por dispositivos convencionais e não convencionais de RV, possibilitou alcançar a interação, imersão e envolvimento do usuário com a cena e com as amostras tridimensionais. Através de tais classes, os usuários puderam atender às amostras e à cena virtual, observando aos efeitos reativos em tempo real, provocando os sentidos do utilizador.

Comprovou-se também a ocorrência de *Gimbal Lock* e da falha de interpolação ao se tentar comutar rotações em eixos de *Euler*, bem como a eficiência da solução proposta através da implementação do sistema de parametrização por quatérnios, para movimentos de rotação interpolados. Desta maneira, a adoção de eixos intermediários para executar rotações no presente trabalho, mapeadas de

movimentos de luvas de dados, sobre modelos tridimensionais, surgiu como alternativa viável para sistemas de Realidade Virtual interativos, considerando que qualquer orientação pôde ser alcançada e qualquer caminho para as mesmas pode ser adotado, tornando irrestrita a movimentação de luvas de dados e a experiência mais realista. Ao mesmo tempo, a classe *Filtro* forneceu toda a estabilidade necessária para os movimentos de manipulação e navegação descritos anteriormente. Assim, a limitação de *hardware* ao interpretar as leituras do dispositivo, pôde ser devidamente controlada, impedindo que posições e orientações errôneas fossem computadas e posteriormente utilizadas em cálculo de novas matrizes, acarretando em movimentos aleatórios.

Os resultados obtidos na etapa de validação demonstraram a efetiva qualidade da análise de imagens da ciência do solo em ambiente de realidade virtual, considerando que os usuários foram intruídos para a utilização do sistema e convidados a responderem um questionário de acordo com suas respectivas impressões. Nenhum dos usuários teve qualquer contato com a ferramenta antes dos testes aplicados.

O estudo de caso demonstrou a aplicabilidade do método em processos de visualização e análise de amostras de solos agrícolas, considerando os avanços e facilidades ao realizar inspeções não invasivas, uma vez que o *Ambiente de Realidade Virtual* possibilitou mensurar o volume de vazios das amostras (poros) e simular o trajeto do fluxo de água e solutos para a formação dos caminhos preferenciais (*fingering*).Desta maneira, tal resultado amplia a viabilidade do uso de Realidade Virtual na área de ciência de solos.

Assim, o ambiente de análise proporcionou o conhecimento do grau de eficiência e aplicabilidade das diversas ferramentas computacionais gráficas inseridas no sistema, como também dos dispositivos convencionais e não convencionais para a investigação das amostras tridimensionais.

Como trabalhos futuros sugere-se o aperfeiçoamento do ambiente de visualização e análise de imagens tomográficas da ciência do solo através de técnicas da Física Computacional, da Realidade Virtual (RV) e da Computação Gráfica (CG). Para tanto, pretende-se utilizar dispositivos não convencionais de entrada e saída de dados, processados sob arquiteturas *GPU* paralelas. (Unidade de Processamento Gráfico). Desta maneira, modelos tridimensionais reconstruídos poderão ser

importados a um ambiente de RV e tratados mediante diversos processos de CG, implementados em arquitetura concorrente e dedicada, onde o ambiente permitirá o compartilhamento de texturas através de uma abstração de memória compartilhada distribuída, permitindo assim que atributos físicos sejam extraídos e analisados em tempo real, em um ambiente de alto desempenho gráfico, somados aos processos interativos de RV.

Referências Bibliográficas

Araújo, R.B. **Especificação e Análise de um Sistema Distribuído de Realidade Virtual**. Tese de Doutorado, EPUSP, 1996.

Azuma, R., Bishop, G. ***Improving Static and Dynamic Registration in a See-Through HMD***. Proceedings of SIGGRAPH, Annual Conference Series, pp. 197-204, 1994.

Bazan, O. **Usinagem de Próteses para Cranioplastia a partir de Imagens Tomográficas**. Dissetação de Mestrado. Universidade Federal do Paraná - UFPR. Curitiba, 2004.

Begault, D. R. ***3-D Sound for virtual reality and multimedia***, Academic Press, Cambridge, MA, 1994.

Benes, J.A; Bueno, R.P. **Ambiente Virtual para Planejamento de Hepatectomia**. Trabalho de conclusão de curso. Pontifícia Universidade Católica do Rio Grande do Sul – PUC-RS. Porto Alegre, 2003.

Bolt, R.A. ***"Put-that-there": Voice and Gesture at the graphics interface***. 7th International Conference on Computer Graphics and Interactive Techniques, pp. 262-270, Washington, 1980.

Botega, L.C., Nunes, F.L.S., **Implementação de estereoscopia de baixo custo para aplicações de Realidade Virtual para treinamento medico**. Anais do XVIII Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens – SIBGRAPI. Natal, 2005.

Botega, L.C.; Cruvinel, P. E. ***Three-Dimensional Soil Physics Image Analysis Based on a Virtual Reality Environment***. IX Symposium on Virtual and Augmented Reality, Petrópolis, 2007.

Brito, J.L.N.S. e Coelho, L.C.T. ***The E-Foto Project: An Educational Digital Photogrammetric Workstation.*** In: ISPRS Commission VI Mid Term Symposium on New approaches for Education and Communication, São José dos Campos, 2002

Brooks, R. A ; Di Chro, G.*,* ***Principles of computer assisted tomography (CAT) in radiographic and radioisotopic imaging***, Physics in Medicine an Biology, v. 21 n.5, 1976, p. 689-732.

Bueno, J.M. **Reconstrução e Visualização Tridimensional de Imagens Tomográficas Baseada no uso da Transformada Rápida de Fourier**. Dissertação de Mestrado. (Orientadora Profa Dra. Agma Traina, Co-Orientador Prof. Dr. Paulo E. Cruvinel) *Reality Annual International Symposium (VRAIS'93)*, Seattle, Washington, 18-22 September, pp. 41-46, 1993.

Burdea, G., Coiffet, P. ***Virtual Reality Technology***. John Wiley & Sons, 1994.

Buxton, W. e Fitzmaurice, G.W.(1998). ***HMD's, Caves & Chameleon: A Human-Centric Analysis of Interaction in Virtual Space***. *Computer Graphics: The SIGGRAPH Quarterly,* 32(4), 64-68

Cação, G.R. **Desenvolvimento de um Algoritmo para a Reconstrução Tridimensional para imagens de um minitomógrafo baseado no método de reconstrução algébrica modificado e interpolação**. Dissertação de Mestrado. (Orientador Prof. Dr. Paulo E. Cruvinel) Universidade Federal de São Carlos - UFSCar. São Carlos, 1994.

Calife, D.; Tomoyose, A.; Spinola, D.; Bernardes, J.; Tori, R. ***Robot ARena: Infrastructure for Applications Involving Spatial Augmented Reality and Robots***. IX Symposium on Virtual and Augmented Reality, Petrópolis, 2007.

Cormack, A.M. ***Reconstruction of Densities from their Projections, with applications in Ratiological Physics.*** Phys.Med.Biol, 686-700, 1973.

Crestana, S., **A Tomografia Computadorizada com um novo método para estudos da física da água no solo**, São Carlos, USP, Tese de Doutorado, 140 páginas, 1986.

Crestana, S., Posadas, A. N. D. ***2-D and 3-D Fingering in Unsaturated Soils Investigated by Fractal Analysis, Invasion Percolation Modeling and Non-Destructive Image Processing***. *Fractals in Soil Science*, 293-331. CRC Press. Washington, EUA, 1998.

Cruvinel, P.E. **Minitomógrafo computadorizado de raios X and _ para aplicações multidisciplinares**. Tese de Doutorado. (Orientador Prof. Dr. Sérgio Mascarenhas) Universidade Estadual de Campinas – UNICAMP. Campinas, 1987.

Cruvinel, P.E.. *et* al. ***X and gamma-ray cumputadorized minitomograph scanner for soil science***. IEEE Transactions on Instrumentation and Measurement, v.39, N.5, p.745-750, 1990.

Cruvinel, P. E.; Crestana, S., ***Opportunities and use of Digital Signal Processor in Tomography dedicated to agriculture***, Anais do Workshop Brasileiro de Arquiteturas Alternativas Usando DSP's, p. 93-97, São Carlos, 1996.

Cruvinel, P. E.; Balongun, F. A. ***Compton Scattering Minitomography Scanner for Soil Science***, Advances in Agricultural Tomography, Embrapa Agricultural Instrumentation, São Carlos, Brazil, pp 24-30, 2000.

Cruvinel, Paulo E. and Balogun, Fatai A. ***Compton scattering tomography for agricultural measurements***. *Eng. Agríc.*, Apr 2006, vol.26, no.1, p.151-160.

Cruz-Neira, C., Sandin, D. J., DeFanti, T. A., Kenyon, R. V., Hart, J.C. **The CAVE audio visual experience automatic virtual environment**, *Communication of the ACM*, 35(6):64-72, 1992.

Cruz-Neira, C., Sandin, D. J., Defanti, T. A. ***Sorround-screen projection based virtual reality: the design and implementation of the CAVE***. Proceedings of the 20th Annual Conference of Computer Graphics and Interactive Techniques, pp. 135-142, 1993.

Darpa. Defense Advanced Research Projects Agency (DARPA). ***Darpa HMD Project Summaries.*** Disponível em <http://www.darpa.mil/mto/displays/hmd/projects>. Acesso em Janeiro de 2007.

Daubechies, I. ***Ten Lectures on Wavelets***, Philadelphia. SIAM, (1992).

Dayeang Co. 5DT, fifth dimension technology, ***5DT data glove specification***. Disponível em <hhtp://www.dayeang.com>. Acesso em Janeiro de 2007.

Delfino, S.; Nunes, F.L.S. **Estudo Comparativo entre Algoritmos de Segmentação para Simulação Tridimensional de Imagens Mamográficas**. Anais do II Workshop de Visão Computacional, 2006, São Carlos-SP, 2006.

E. Reality. ***P5glove specification***. Disponível em: <http://www.inition.com>, 2007.

Gattass, M., Raposo, A. B., da Silva, J. M. **Grafo de Cena e Realidade Virtual.** Monografia de conclusão de curso. Pontifícia Universidade Católica do Rio de Janeiro – PUCRIO, 2004.

Gattass, M; Biasi, S. C. **Utilização de Quatérnios para representação de rotações em 3-D**. *Reports from academic research*. Disponível em: <http://www.tecgraf.puc-rio.br/~mgattass/ > , 2002.

GeorgiaTech. ***Self-Aligning Liquid Crystal Technique Could Simplify Manufacture of Display Devices.*** Disponível em <gtresearch.gatech.edu>. Acesso em Janeiro de 2007.

Gradescki, J. ***The virtual reality construction kit***, John Wiley & Sons, 340 Pp., 1995.

Granato, L. F., **Algoritmo adaptativo para a melhoria em imagens tomográficas obtidas em múltiplas energias**, São Carlos, UFSCar, Dissertação de Mestrado, 1998, 135 páginas.

Hainsworth, J.M.; Aylmore, L.A.G., ***The use of the computed-assisted tomography to determine spatial distribution of soil water content***, Aust.Journal Soil Res. n.21, p.1435-1443, 1983.

Hearn, D. e Pauline, M. ***Computer Graphics: C Version***. Second edition prentice hall. 1997

Heeter, C. Presence: ***Teleoperators and Virtual Environments***. MIT Press, 1992.

Herman, G. T., Gordon, R. ***Three-Dimensional Reconstruction from Projections: A Review of Algorithms***, pp 111-153, 1973**.**

Hounsfield, G.N. ***Computadorized Transverse Axial Scanning (Tomography): Part I. Description of System***. British Journal of Radiology, n.46, p. 1016-1022, 1973

Hultquist, J.P.M., e Haible, E.L. Superglue: ***A programming environment for scientific visualization***, Procedure Visualization, pp. 243-250, 1992.

Immersion Corp3D. ***Interaction Overview***. Disponível em <http://www.immersion.com/3d/>. Acesso em Janeiro de 2007.

Ipolito, J e Kirner, C. **Técnicas de otimização e realismo em aplicações de Realidade Virtual**. Workshop de Realidade Virtual – WRV, pp. 91-100, UFSCar, São Carlos, 1997.

Sutherland, I. E. ***Sketchpad-A Man-Machine Graphical Communication System***. Anais do *Spring Joint Computer Conference*, Detroit, Michigan, 1963.

Jacobson, L. ***Garage Virtual Reality***, SAMS Pub., Indianapolis, IN, 1994.

Kak, A.C., Slaney, M. ***Principes of Computerized Tomographic Imaging***. IEEE Press, New York, 1988.

Kalawsky, R. S. ***The science of virtual reality and virtual environments***, Ed. Addison- Wesley, 405 Pp., 1993.

Kenner, C. ***Essential reality p5glove sumary: Dual mode driver programming***. Disponível em <http://www.geocities.com/carl_a_kenner/p5.html>, 2007.

Kirner, C., Deriggi, F., Kubo, M.M., Sementille, A. C., Brega, J, F., Santos, S. ***Virtual Environments for Shared Interactive Visualization***. Workshop of the german-brazilian cooperative program in informatics, Berlin – Alemanha, 1995.

Kirner, C; Siscoutto, R. ***Realidade Virtual e Aumentada: conceitos, projetos e aplicações. Livro do pré-simpósio***. IX Symposium on Virtual and Augmented Reality, 290Pp, 2007.

Kitware Inc. ***VTK 5.0.2 Documentation***. Disponível em <http://www.vtk.org/doc/release/5.0/html/>. 2006.

Krueger, M. W. ***Responsive environments***. NCC Proceedings, pp. 375–385, 1977

Kubo, M.M. Santos, S.G., Deriggi J.R., Kirner, C.. ***Múltiplas visões em um ambiente virtual multiusuário***. Workshop de Realidade Virtual – WRV, pp. 62-70, UFSCar, São Carlos, 1997.

Laia, M. A. M.; Cruvinel, P. E. ***Filtragem de Projeções Tomográficas da Ciência do Solo Utilizando Transformada Anscombe e Kalman***. 6º Brazilian Conference on Dynamics, Control and Their Applications, São José do Rio Preto, 2007.

Laia, M. A. M; Cruvinel, P. E. ***Filtragem de Projeções Tomográficas da Ciência do Solo Utilizando Kalman e Redes Neurais***. IEEE Latin América Transactions, pp.114-121, vol.6, nº1, 2008.

Laia, M.A.M; **Filtragem de Projeções Tomográficas da Ciência do Solo Utilizando Kalman e Redes Neurais**. Dissertação de Mestrado. (Orientador Prof. Dr. Paulo E. Cruvinel) Universidade Federal de São Carlos – UFSCar. São Carlos, 2007.

Lane, C. Display Technologies (1993). ***Encyclopedia of Virtual Environments***. Disponível em <http://www.hitl.washington.edu/scivw/EVE/I.A.1.Displays.html>. Acesso em Janeiro de 2007.

Lanier, J. ***"Virtual Reality: The promise of the future"***. Interactive Learning International, 8(4), pp. 275-279, New York, 1992.

Latta, J. N. e Oberg, D. J. ***A conceptual virtual reality model****, IEEE Computer Graphics & Applications*, pp. 23-29, Jan., 1994.

Leston, J. ***Virtual reality: the it perspective***, *Computer Bulletin*, pp. 12-13, June, 1996.

Lillersand, T., Kiefer, R. ***Remote sensing and image interpretation***. New York: John Wiley & sons, 2000.

Lima, J.; Guimarães, G.; Silva, G.; Teixeira, J.; Xavier, E.; Teichrieb, V.; Kelner, J. ***ARCam: a FPGA-based Augmented Reality Framework***. IX Symposium on Virtual and Augmented Reality, Petrópolis, 2007.

Machado, L. S. **Conceitos básicos da realidade virtual**, Monografia, INPE, Instituto Nacional de Pesquisas Espaciais, São José dos Campos/SP. Disponível em <http://www.lsi.usp.br/~liliane/conceitosrv.html>, 1995.

Machover, C; Tice, S.E. ***Virtual Reality.*** IEEE Computer Graphics & Applications, Jan. 1994

Malfatti, S.; Santos, S.; Oliveira, J.; Justel, C.; Fraga, L.; ***EnCima: A Graphics Enjine for the Development of Multimedia and Vritual Reality applications***. X Symposium on Virtual and Augmented Reality, João Pessoa, 2008.

Manssour, I.H. **Introdução à Java 3D**. 2003. Disponível em <www.inf.pucrs.br/manssour/Java3D>**.** Acesso em Julho de 2006.

Marcus, B. A., Beth A. ***EXOS Research on Master Controllers for Robotic Devices***. In Proceedings of 1991 SOARP Conference. 1991. Minas Gerais, 1996.

MayaVi Data Visualizer. ***MayaVi Documentation***. Disponível em: <http://mayavi.sourceforge.net//docs.html>. Access on June 2006.

Minatel, E.R. **Desenvolvimento de Algoritmo para Reconstrução e Visualização Tridimensional de Imagens Tomográficas com o uso de Técnicas Freqüenciais e Wavelets**. Dissertação de Mestrado. (Orientador Prof. Dr. Paulo E. Cruvinel) Universidade Federal de São Carlos - UFSCar. São Carlos, 1997.

Morie, J. F. ***Inspiring the future: merging mass communication, art, entertainment and virtual environment***, *Computer Graphics*, 28(2):135-138, May, 1994.

Nieto, A. U. Vandersteegen, P., Buggenhout, C., Verstuyft, S., Bienstman, P., Neyts, K., Baets, R. ***Increasing light extraction of a substrate emitting*** OLED using a 2D surface grating. IEEE LEOS Annual Meeting. 2006.

Oldendorf, W.H. ***Isolated Flying Spot Detection of Radiodensity Discontinuites – Dysplayng the Interhal Structural Pattern of a Complex Object***, IRE Transactions of Biomedical Eletronics, 8, p. 68-72, 1961.

Oliveira, A.C.M.T., Pavarini, L., Botega, L.C., Nunes, F. L. S., Bezzera, A. ***Virtual Reality Framework for Medical Training: Implementation of a Deformation class using Java***. ACM International Conference on Virtual Reality Continuum and Its Applications, 2006, Hong Kong.

Paradella, W.R. **Produção de Carta Topográfica através da estereoscopia de Alta Resolução do RADARSAT-1 Integrada com Dados TM-Landsat 5: Uma Avaliação para Terrenos Planos da Floresta Nacional de Tapajós**, **Revista Brasileira de Geociências,** , pp: 99-110, 2003.

Pavarini, L., Oliveira, A.C.M.T., Botega, L.C. **DefApliMed - Sistema de Deformação para Aplicações Médicas com base no Método Massa -Mola utilizando a API Java 3D**. In: SVR - *Symposium on Virtual Reality*, 2006, Belém. Anais do SVR 2006, 2006. v. 8.

Pedrotti, A. Pauletto, E. A., Crestana, S., Cruvinel, P. E., Vaz, C. M. P., Naime, J. M., Silva, A. M. **Tomografia Computadorizada aplicada a estudos de um planossolo**. Pesquisa Agropecuária Brasileira, vol. 38, numero 7. Brasília, 2003.

Perdigão, N., Manuel, J., Tavares, R. S., Martins, J.A.C., Pires, E.B., Jorge, R.M.N. **Sobre a Geração de Malhas Tridimensionais para fins computacionais a partir de Imagens Médicas**. Anais do Congresso de Métodos Numéricos em Engenharia. Granada, Espanha, 2005.

Pereira, M. F. L., Cruvinel P. E., ***Parallel DSP Architecture for Reconstruction of Tomographic Images Using WaveletsTechniques***. Anais do XIV Simpósio brasileiro de computação gráfica e processamento de imagens (SIBGRAPI), p. 384-384. Florianopolis. 2001.

Pereira, M.F.L. **Algoritmo Paralelo para Reconstrução Tridimensional de Imagens Tomográficas de amostras Agrícolas em arquitetura DSP com técnicas Wavelets**. Dissertação de Mestrado. (Orientador Prof. Dr. Paulo E. Cruvinel) Universidade Federal de São Carlos – UFSCar. São Carlos, 2001.

Pereira, M. F. L.; Cruvinel, O. E.; Costa, L. da F.; Saito, J. H.; Macedo, A. ***Uma ferramenta para análise de parâmetros físicos de solos baseada em com tomografia de raios X***. Congresso Brasileiro de Agroinformática - SBIAgro, 2007. São Pedro, SP. Campinas: Embrapa Informática Agropecuária, 2007.

Pereira, M. F. L. **Um modelo de reconstrução tomográfica 3D para amostras agrícolas com filtragem de Wiener em processamento paralelo.** Tese de Doutorado. (Orientador Prof. Dr. Paulo E. Cruvinel) Universidade de São Paulo – USP. São Carlos, 2007.

Peruzza, A.P.P.M., Nakamura, E.Y., Kirner, C. **Solução do problema de detecção de colisão em ambientes de Realidade Virtual**. Workshop de Realidade Virtual – WRV, pp. 36-43, UFSCar, São Carlos, 1997.

Pessoa, S.; Apolinário, E.; Moura, G.; Lima, J.; Bueno, M.; Teichrieb, V. ***Illumination Techniques for Photorealistic Rendering in Augmented Reality***. X Symposium on Virtual and Augmented Reality, João Pessoa, 2008.

Petrovic, A.M., Siebert, J. E., Rieke, P. E. ***Soil bulk analysis in threedimensions by computed tomographic scanning***, Soil Science Soc. Am. J., n.46, p.445-450, 1982.

Phasespace Motion Capture. ***Impulse Glove Specification***. Disponível em <http://www.phasespace.com/gloves.html>, 2008.

Pimentel, K; Teixeira, K. ***Virtual Reality – through the new looking glass***. *2nd edition*. New York, McGraw-Hil, 1995.

Pizzolato, E.; Fernandes, M.; Duarte, D. ***Speech Enable 3D Browsers: Development Issues and Software Framework***. X Symposium on Virtual and Augmented Reality, João Pessoa, 2008.

Prado, A.H.M.A; Traina, A.J.M. **Interpolação de Imagens Tomográficas através de *Matching* usando Triangulação de Delaunay**. Anais do IX SIBGRAPI – Simpósio Brasileito de Computação Gráfica e Processamento de Imagens. p.359-360. Caxambu,

Radon, J.; ***In the Determination of functions from their integral along certain manifold***, Beriche über die Verhandlungen, v. 69, p. 262-277, 1917.

Robertson, G. G., Card, S. K., Mackinlay, J. D. ***Non-immersive virtual reality****, IEEE Computer*, pp. 81-83,1993.

Romano, J. M. S. **Integração de um Dispositivo Óptico de Rastreamento a uma Ferramena de Realidade Virtual**. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro. (Orientador: Marcelo Gattass), 2004.

Sanches, S.; Rodello, I.; Brega, J.; Sementille, A. ***Um componente para Construção de Cenas com Consideração de Profundidade em Ambientes de Realidade Misturada***. X Symposium on Virtual and Augmented Reality, João Pessoa, 2008.

Santos, S.G., Kubo, M.M., Deriggi, F.V., Kirner, C. ***Multiple Views in a Multiuser Environment as support for CSCW***. Simpósio Brasileiro de Multimídia – SBMIDIA, pp 65-73, Rio de Janeiro, 1998.

Scgl. Stanford Computer Graphics Lab. ***Responsive Workbench Tutorial***. Disponível em <http://graphics.stanford.edu/projects/RWB/>. Acesso em janeiro de 2007.

Schneider, B. O. e Kirner, C. **Controle de objetos deformáveis em ambientes virtuais. Workshop de Realidade Virtual** – WRV, pp. 44-51, UFSCar, São Carlos, 1997

Selman, D. ***Java 3D Programming***. Editora *Manning Publications Company*, 2002.

Sensic. High ***Performance 3D displays: panoramic hmd's. Hmd specification***. Disponível em <http://www.sensics.com/products_specifications.html>. Acesso em Janeiro de 2007.

Shepherd, B. J. ***Rationale and strategy for VR standards***, *Proceedings of the IEEE Virtual*

Silva, A.M.M ; Andrade, M.A**.** ***Interface*** **gráfica em Java para reconstrução tomográfica em medicina nuclear**. Anais do III Congresso Latino Americano de Engenharia Biomédica. IFMBE Proceedings, v. 5. p. 1267-1269. João Pessoa. 2004.

Souza, M.A., Ricetti, F., Erthal, J. L., Pedrini, H. **Reconstrução de Imagens Tomográficas aplicada à fabricação de próteses por prototipagem rápida usando técnicas de triangulação**. Anais do II Congresso Latino Americano de Engenharia Biomédica. Havana, Cuba, 2001.

Sun, Inc. ***Java 3D 1.5.0 API Specification***. Disponível em: <http:// http://java.sun.com/products/java-media/3D/releases.html>. Acesso em Novembro de 2006.

Spencer, R. ***Information Visualization***. Addison-Wesley, 2001.

Steuer, J. ***Defining Virtual Reality: Dimensions determining telepresence***. Journal of Communication, 1992.

Takahashi, H. Shimoda, K.**,** Townes, C.H.**.. *Fluctuation in amplification of quanta with application to maser amplifiers.*** J.Phy. Soc., Japan, vol.12, pp., 1957.

Teixeira, L.; Loaiza, M.; Raposo, A.; Gatass, M. ***Um Sistema Híbrido baseado em Esferas Retroreflexivas e Características do Objeto Rastreado***. X Symposium on Virtual and Augmented Reality, João Pessoa, 2008.

Tiger. Teachers Instructional Graphics Educational Resource, *North Carolina school of science and mathematics,* ***Chemistry Distance Learning Technologie***s. Disponível em <http://www.dlt.ncssm.edu/tiger/chem1.htm>. Acesso em janeiro de 2007.

Toutin, T. Stereo ***RADARSAT for Mapping Applications***. In: Proceedings, ADRO Final Symposium, Montreal, Canadá, 1998, CD-ROM.

Tsuda, F.; Hokama, P.; Rodrigues, T.; Bernardes, J. ***Integration of jARToolKit and enJine: Extending with AR the Potential Use of a Didactic Game Engine***. IX Symposium on Virtual and Augmented Reality, Petrópolis, 2007.

Upson, C. T., Faulhaver, T., Kamins, D., Laidlaw, D., Schlegel, D., Vroom, J., Gurwitz, R., Van Dam, A. ***The application visualization system: A computacional environment for scientific visualization***, Computer Graphics and Applications, 9(4), pp. 30-42, 1989.

Venturini, Y. R., **Análise quantitativa da qualidade de imagens digitais com uso do espectro de Wiener**, São Carlos, Dissertação de Mestrado, Universidade de São Paulo - USP. São Carlos, 1995.

Vester, C e Toutin, T. ***A training package on how to use RADARSAT data in stereo***. In: ADRO, Intern. Symp. Geomatics in the era of RADARSAT, Ottawa, Canada, Proceedings, CD-ROM.

Villamil, M.B., Nedel, L. P., Freitas, C. M. D. S., Machado, R.A., Silveira, R.L., da Silva, A. M. M. **Simulação do Movimento da Mandíbula e Comportamento da Articulação Temporomandibular**. Anais do V WIM – Workshop de Informática Médica. Porto Alegre, 2005.

Von Schweber, L. & Von Schweber, E.. ***Virtual reality: Virtually here****.* PC Magazine, 168-170, 1995.

Von Schweber, L. e Von Schweber, E. **Cover story: realidade virtual**, PC Magazine Brasil, pp. 50-73, v. 5, n. 6, 1995

Vrdepot. Products: ***CyberGlove Specification***. Disponivel em: <http://www.vrdepot.com/trkglv.htm>. 2007

Vrealities. ***Head mounted displays overview: 3d stereoscopic***. Disponível em <http://www.vrealities.com/hmd.html>. Acesso em janeiro de 2007

Vresources. *Virtual Reality Resources*. ***HMD Fundamental Concepts***. Disponível em < http://vresources.org/articles/vre_articles/>. Acesso em Dezembro de 2006.

Vrlogic. Products: ***Head Mounted Displays Overview***. Disponivel em: <http://www.vrlogic.com/html/head_mounted_displays.html>. Acesso em Janeiro de 2007.

Wavefront, Tech. ***Wavefront File Format Specification***. Disponível em: <http://www.fileformat.info/format/wavefrontobj/spec/index.htm>. 2006

Web3D Consortium. ***Xj3D Developer Documentation***. Disponível em:< http://www.xj3d.org/arch.html>. 2006.

Web3D Consortium. ***Virtual Reality Modeling Language Functional Specification***. Disponível em: <http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/>. 2006.

ANEXO I

Ficha preparada para avaliação do sistema de análise de imagens tomográficas da ciência do solo ensaiadas em ambiente de realidade virtual

Esta ficha tem por objetivo auxiliar a avaliação do sistema de análise de imagens dedicado à manipulação de amostras agrícolas reconstruídas com base no método de tomografia de raios-*X*. O ambiente foi construído com o uso de ferramentas gráficas computacionais e dispositivos não convencionais de Realidade Virtual (RV), dentre eles uma luva de dados e um video-capacete para interação do usuário com o ambiente de análise.

Quanto ao grau de avaliação a ser atribuído, a escala é do menor para o maior e a aplicação desta ficha de avaliação será desenvolvida de forma supervisionada, onde será previamente explicado ao usuário, os conceitos relativos aos termos: interface, customização de amostra, customização de ambiente, obtenção de dados, detecção de colisão, imersão e interação.

1 – Quanto á interface:

A - Você pôde identificar os recursos disponíveis?

B - Você sentiu alguma dificuldade em encontrar os recursos disponíveis? Quais recursos? Quais Dificuldades?

Recursos disponíveis

Item avaliado	**Grau atingido** (1-6)

2 – Quanto à customização de amostra:

A - Você conseguiu alterar o modo de representação dos polígonos (faces, arestas, pontos), a transparência e as cores?

B - Você conseguiu alterar o formato e dimensões da amostra em teste?

Item avaliado (grau de dificuldade encontrado)	**Grau atingido** (1-6)

3 – Quanto à customização do ambiente:

A - Você conseguiu alterar a iluminação do ambiente?

B - Você conseguiu alterar a posição das câmeras do ambiente?

Item avaliado (grau de dificuldade encontrado)	**Grau atingido** (1-6)

4 – Quanto à obtenção de dados:

A - Você conseguiu exibir os dados das amostras (fronteiras, vértices, pontos, distâncias, cores, material, transparência, polígonos, coeficientes de atenuação)?

Item avaliado (grau de dificuldade encontrado)	**Grau atingido** (1-6)

5 – Quanto à detecção de colisão:

A - Você conseguiu executar uma colisão utilizando a luva ou teclado?

Item avaliado (grau de dificuldade encontrado)	**Grau atingido** (1-6)

6 – Quanto à imersão:

A - O sistema lhe proporcionou a sensação de fazer parte do ambiente de RV através dos métodos de visualização (óculos, anaglifo e realidade aumentada)?

Item avaliado (grau de dificuldade encontrado)	**Grau atingido** (1-6)

7 – Quanto à interação:

A - O sistema reagiu às suas ações? (Definição de novos parâmetros em Polígonos, Transparência, Cores, Iluminação, Transformações, Manipulação por luva, teclado e mouse).

Item avaliado (grau de dificuldade encontrado)	**Grau atingido** (1-6)

Identificação (opcional) do avaliador:
Nome:
Idade:
Sexo:
Grau de escolaridade:
E-mail:
Endereço: