TIAGO MARTINUZZI BURIOL

PROCESSAMENTO E VISUALIZAÇÃO DE CAMPOS EM AMBIENTES VIRTUAIS E SISTEMAS CAD 3D APLICADOS A PROJETOS DE ILUMINAÇÃO EM SUBESTAÇÕES

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciências, Programa de Pós - Graduação em Métodos Numéricos em Engenharia, Setor de Tecnologia e Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Dr. Sérgio Scheer

CURITIBA
2006

A todos que persistem.

AGRADECIMENTOS

Agradeço a Deus por minha saúde e capacidade intelectual. Agradeço imensamente aos meus pais Galileo Buriol e Maria Helena Buriol por terem me apoiado e incentivado a ingressar, persistir e concluir o curso de mestrado. Também à Mariana, mulher, companheira, amiga e mãe do meu filho, por todo amor, paciência e compreensão. Agradeço ao professor e amigo Sérgio Scheer, e ao colega e amigo Marlos Fabris por dividirem comigo parte de seus conhecimentos. Também à equipe da UTIG - Unidade de Tecnologia em Informações Georreferenciadas do LACTEC - Instituto de Pesquisa para o Desenvolvimento, pela concessão da bolsa e oportunidade que me foi dada.

“Imerso no acaso, de peito aberto, pretendo estar...”.
Comissário Silas

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS E ABREVIATURAS

1D	Unidimensional ou uma dimensão
2D	Bidimensional ou duas dimensões
3D	Tridimensional ou três dimensões
API	*Application Programming Interface*
CAD	*Computer Aided Design*
CAE	*Computer Aided Engineering*
CG	Computação Gráfica
E/S	Entrada e saída de dados
GUI	*Graphics User Interface*
HSV	*Hue, Saturation, Value*
IES	*Illuminating Engineering Society*
IHC	Interface Homem/Computador
LSI	Laboratório de Sistemas Interativos
LIC	*Line Integral Convolution*
MEF	Método dos Elementos Finitos
OO	Orientado a Objetos (*Object Oriented*)
P&D	Pesquisa & Desenvolvimento
RGB	*Red, Gren, Blue*
RV	Realidade Virtual
SDK	*Software Development Kit*
SG	*Silicon Graphics Inc.*
SW	*SolidWorks*
TPN	Tanque de Provas Numérico
USP	Universidade de São Paulo
VC8	*Visual C++ 2005 Express Edition*
VC	Visualização Científica
VRML	*Virtual Reality Modelling Language*
VTK	*Visualization ToolKit*
XML	*Extensible Markup Language*
X3D	*Extensible 3D*

RESUMO

Sistemas de visualização de dados científicos são, hoje, vitais em diversas áreas, como medicina e engenharia, por exemplo. Com eles é possível analisar com maior eficiência grandes conjuntos de dados complexos, provenientes de sensores ou de simulações computacionais. A tecnologia de Realidade Virtual (RV), por sua vez, representa uma forma avançada de interface em que o usuário tem a sensação de estar "dentro" de um ambiente virtual tridimensional (3D) e pode interagir com este. O uso de RV para Visualização Científica (VC) envolve uma arquitetura de sistemas complexa, mas possui grande potencial para diversas aplicações. Nesta dissertação, apresenta-se um estudo sobre processamento e pós-processamento de campos de iluminância que faz uso de técnicas de Visualização Científica (VC) e interface em RV com o fim de simular a iluminação direta em subestações de energia elétrica, provendo a visualização dos resultados numéricos em um ambiente virtual tridimensional (3D) simultaneamente à visualização de modelos CAD (*Computer Aided Design*). O estudo foi feito por meio de uma extensa pesquisa bibliográfica e do desenvolvimento de algoritmos de processamento e pós-processamento de dados. Como resultado, obteve-se um aplicativo para cálculo da iluminação direta, em malhas 2D ou 3D, e visualização em ambiente virtual 3D. Uma análise da obstrução da luz por corpos opacos pôde ser implementada por meio da integração dos algoritmos com o *software* de CAD 3D SolidWorks que, por sua vez, permite a exportação para o formato VRML (*Virtual Reality Modeling Language*). Questões relativas à simulação de iluminação, uso da ferramenta VTK (*Visualization ToolKit*), otimização de modelos CAD para uso em sistemas de *rendering* em tempo real, dispositivos não convencionais de entrada e saída de dados, e aplicações em subestações de energia elétrica são abordadas. Foi usado como conjunto de testes o modelo de dados obtidos de uma subestação real, em operação, pertencente a uma empresa do setor energético brasileiro.

Palavras-chave: Visualização Científica, Realidade Virtual, Iluminação, CAD 3D.

ABSTRACT

Nowadays visualization systems are very important in several areas, such as medicine and engineering. It is possible to analyze complex data sets obtained for sensors or computational simulations. The Virtual Reality (VR) technology represents an advanced form of interface with which the user has the sensation to be inside the virtual environment, interacting with it. The use of VR for Scientific Visualization (SciVis) involves a complex system architecture but has great potential for many applications. This dissertation presents a study about illuminance field processing and post-processing using SciVis techniques and VR interface. It aims at simulating direct illumination in electric power substations and to provide results for visualization in three-dimensional (3D) virtual environment combined with actual 3D CAD (Computer Aided Design) models. In this context, this work describes a study on SciVis techniques and tools, virtual environments and VR systems, and the development of algorithms for illuminance field processing and post-processing. A tool for direct illumination calculation has been developed for 2D and 3D meshes, considering opaques obstacles and the integration with the SolidWorks CAD software. In addition, the use of a 3D CAD system functionalities allows for another integration with virtual environment through the data exportation using VRML (Virtual Reality Modeling Language) format. Some issues about illumination processing and post-processing, the use of VTK (Visualization ToolKit), field visualization techniques, CAD models optimizations for use in real-time rendering systems, VR devices and its applications in power systems substations situations are presented. An actual data model of a Brazilian power substation has been used during this study.

Keywords: Scientific Visualization, Virtual Reality, Illumination, CAD 3D.

1 INTRODUÇÃO

Importantes empresas (petróleo, energia, etc), cada vez mais, buscam sistemas computacionais integrados, capazes de centralizar grandes quantidades de informações e prover diferentes funcionalidades. Nos setores de engenharia, as funcionalidades desejadas são de maneira geral específicas de *software* distintos, como por exemplo, *softwares* para desenho de projetos e modelagem geométrica (sistemas CAD), análise numérica (CAE), visualização de informações, acesso e gerenciamento de bancos de dados, comunicação, dentre outros. A inexistência de um sistema completo resulta em uma série de dificuldades e inconveniências no processo de transferência de dados entre diferentes formatos e/ou *softwares* como perda de tempo e de informações.

Neste cenário, a Realidade Virtual (RV) apresenta-se como uma tecnologia com grande potencial a ser usada em um sistema integrado. É considerada a mais avançada Interface Humano-Computador (IHC) conhecida, capaz de prover ao usuário a sensação de imersão, ou seja, estar “dentro” do ambiente sintético. Um ambiente virtual tridimensional ou 3D pode ser usado como interface para visualização científica, acesso à base de dados, ativação e controle de diferentes funcionalidades, permitindo dessa forma a integração de múltiplos aplicativos em um único sistema. A figura 1 ilustra o uso de ambientes virtuais com aplicação em engenharia.

FIGURA 1 - REALIDADE VIRTUAL, FOTORREALISMO E IMERSÃO

FONTE: Para as duas primeiras imagens o autor e para a terceira < http://www.avrrc.lboro.ac.uk/>.

Vale lembrar também que o grande avanço tecnológico de *softwares* e *hardwares* vivenciado nos últimos anos, faz com que, hoje, seja possível montar um

sistema de visualização utilizando plataforma baseada em computadores ditos pessoais (ou PCs), a um custo cada vez menor. Em matéria de *software*, dispõe-se de inúmeros programas gratuitos ou de código aberto e também programas comerciais com licença temporária gratuita. Quanto a *hardware*, tem-se placas gráficas programáveis que são comercializadas com implementações de bibliotecas gráficas padronizadas e possuem processador próprio, podendo também oferecer suporte a dispositivos especiais, como óculos para visão 3D. Com isso, abrem-se novas possibilidades no que diz respeito ao desenvolvimento de aplicativos gráficos interativos para PCs, ampliando seu leque de aplicações.

A busca pelo desenvolvimento de um sistema computacional completo e ideal, independente da aplicação, certamente deve contemplar a visualização interativa de grandes conjuntos de dados 3D. Em especial, nas engenharias, a visualização de campos 3D (escalares, vetoriais e tensoriais) exerce importante papel na apresentação de resultados de simulações numéricas, como aqueles obtidos pelo Método dos Elementos Finitos (MEF), por exemplo. Também é interesse de engenheiros visualizar grandes modelos digitais 3D (modelos CAD), como plantas industriais, em sistemas de RV, beneficiando-se, dessa forma, das características de imersão e interação oferecidas por esta tecnologia, e do impacto visual proporcionado.

No caso específico de subestações de energia, existe o interesse por parte de empresas e concessionárias em utilizar a tecnologia da RV para visualizar de forma exploratória o modelo digital 3D de suas plantas e instalações. Este tipo de visualização pode servir de apoio a uma série de atividades de planejamento (ampliação, sistemas de segurança, transporte de equipamentos, sistemas de iluminação, etc), treinamento e simulações.

O uso de modelos digitais 3D como interface para acessar informações diversas pode diminuir a necessidade de visitas de técnicos e engenheiros ao local. Além disso, amplia as possibilidades de transmissão de informações 3D pela *web* e, conseqüentemente, melhora a colaboração entre profissionais. Também possibilita a visualização de grandezas físicas atuantes no local (campos eletromagnéticos, por exemplo), aumentando assim, a capacidade de detectar distâncias críticas para o

transporte de cargas em áreas contendo equipamentos energizados, podendo até mesmo, reduzir eventuais interrupções de energia e outros prejuízos.

1.1 JUSTIFICATIVA

Como cita Netto et al. (2002), empresas têm utilizado a RV em campos como automação de projetos, venda e *marketing*, planejamento e manutenção, treinamento e simulação, e concepção e visualização de dados. Em um outro trabalho, Netto et al. (2005), é citado o uso de um ambiente virtual como interface 3D para manipulação de dados em redes de distribuição de energia elétrica, com a finalidade de auxiliar a tomada de decisão em um sistema para redução de perda, e facilitar a interpretação (avaliação) das soluções propostas pelo sistema. Para Corseuil et al. (2004), a visualização de grandes modelos 3D de engenharia, como uma plataforma de petróleo ou uma subestação de energia, melhora a representação, percepção e análise do espaço físico envolvido. Além disso, o uso de RV permite um ganho nos processos de análise quando é conjugada com processos de simulação.

No caso de subestações de energia elétrica, têm-se pátios lotados de cabos e outros equipamentos eletrizados, como transformadores e capacitores. Atividades de manutenção, comumente requerem a presença do operário em locais perigosos, próximo a cabos de alta tensão ou no alto de grandes torres. Em alguns casos o planejamento de consertos, transporte de equipamentos, ampliação, e muitas outras atividades também requerem a presença de engenheiros no local, envolvendo assim riscos e custos.

“Equipes enfrentam atividades de alto risco em locais com condições climáticas e de acesso adversas” (FURNAS, 2005, p.12), como colocado no artigo referindo-se aos trabalhos de recuperação das torres e das linhas de transmissão avariadas pelas tempestades ocorridas no Estado do Paraná, em 2005. Nesse episódio mobilizaram-se cerca de 200 pessoas entre diretores e engenheiros de várias áreas.

Em outro momento Furnas (2006) declara: “Com um sistema em RV, a concessionária pretende ter acesso a informações que muitas vezes não estão

disponíveis devido à impossibilidade (técnica e de segurança) de fazer esses levantamentos por meio dos métodos tradicionais".

Além da visualização do modelo CAD, é possível visualizar em RV dados de grandezas físicas diversas, como campos escalares e vetoriais, provenientes de simulações numéricas por métodos como o MEF (KLOCKE e STRAUBE, 2004; LU et al., 2001; GARCIA, 2001). A visualização combinada de modelos CAD com dados resultantes de análises numéricas em ambiente virtual 3D é um poderoso recurso a ser utilizado na obtenção de informações a partir de grandes conjuntos de dados, melhorando a compreensão desses dados, podendo reduzir custos de projeto e outros.

O desenvolvimento de sistemas interativos de visualização em RV requer conhecimento sobre elementos de Computação Gráfica (CG), Interface Humano-Computador (IHC), Engenharia de Software, sistemas CAE (*Computer Aided Engeenering*) e outros.

Assim, é importante a realização de um estudo exploratório de uma ferramenta (*toolkit*) para VC que possa ser utilizada no desenvolvimento de um sistema de RV para visualização de campos (escalares, vetoriais e tensoriais). É relevante que se considere a possibilidade de integração com algum sistema de CAD e com algoritmos de simulação numérica, buscando a implementação de funcionalidades direcionadas a alguma aplicação específica, como por exemplo, a visualização de campos eletromagnéticos 3D em subestações de energia elétrica (MARINOVA et al., 2001).

1.2 DELIMITAÇÃO DO PROBLEMA

Esta dissertação está relacionada com um projeto de Pesquisa e Desenvolvimento (P&D) que visa o desenvolvimento de um sistema de RV aplicado a atividades em subestações de energia elétrica. Neste projeto levantaram-se diversos requisitos e funcionalidades desejáveis que pudessem ser desenvolvidas e implementadas neste sistema.

Um dos requisitos levantados foi que o sistema pudesse servir como uma ferramenta de apoio em projetos de iluminação. A idéia básica era utilizar a interface

em RV para visualizar o resultado de diferentes simulações a fim de possibilitar uma análise comparativa. Outro requisito levantado, mas que não foi desenvolvido neste trabalho, embora tenha características semelhantes ao anterior, foi de poder visualizar a distribuição de campos elétricos e magnéticos atuantes na subestação. A semelhança entre esses dois requisitos reside no fato de que ambos fundamentam-se na utilização de técnicas de VC com a finalidade de representar graficamente conjuntos de dados complexos provenientes de análises numéricas.

Neste contexto, o problema consiste em viabilizar a visualização científica de campos escalares e vetoriais, invariáveis no tempo, em um ambiente virtual tridimensional. A busca da melhor solução envolve um estudo de plataformas, técnicas e ferramentas de visualização de campos, linguagens de programação, estrutura de dados, geração de geometrias, dispositivos especiais de E/S, IHC, *softwares* e linguagens para modelagem, entre outros tópicos.

1.3 OBJETIVOS

É conhecido o fato de que, em diversas empresas, a decisão de operadores e engenheiros tem grande importância, uma vez que costumam resultar em impactos substanciais no tocante a mercado e finanças. Assim, cada vez mais, respostas precisas a perguntas bastante complexas devem ser providas rapidamente, fazendo com que a gerência e a visualização de informações pertinentes tenham um papel importantíssimo.

Sistemas de RV têm sido desenvolvidos objetivando proporcionar ao usuário uma visualização sofisticada, capaz de prover a sensação de imersão através da navegação no modelo 3D, da interação em tempo real e de estímulos a múltiplos sentidos. Mas a tecnologia de RV não é caracterizada somente pela capacidade de prover imersão e interação, mas também pela sua enorme potencialidade de uso em diferentes aplicações, pois envolve soluções para problemas reais de engenharia, medicina, segurança e defesa civil, meio ambiente, e outros.

Vale citar também que, atualmente, é comum em engenharia a modelagem

matemática de problemas físicos por meio de sistemas de equações diferenciais parciais, para as quais são obtidas soluções aproximadas utilizando métodos numéricos (ANSYS, 2006). A visualização 3D dos resultados obtidos pelo MEF, por exemplo, bem como por outro método numérico qualquer, consiste em um poderoso recurso para análise e compreensão desses resultados que geralmente são conjuntos complexos de dados, cujo comportamento é muito difícil de ser observado de uma forma global.

No caso de subestações de energia elétrica, existe o interesse em visualizar campos de iluminâncias como apoio em projetos de iluminação, e também visualizar campos elétricos e magnéticos, a fim de identificar restrições no transporte de equipamentos e outras atividades.

Neste sentido, o objetivo geral deste trabalho foi utilizar o modelo digital CAD de uma subestação de energia elétrica em um ambiente virtual 3D, a fim de criar uma interface em RV para visualização de campos atuantes nesta subestação.

Um dos objetivos específicos foi verificar o potencial do uso de RV, fazendo uso da linguagem VRML (*Virtual Reality Modelling Language*, padrão ISO para transferência de conteúdo 3D na *web*), para o planejamento de sistemas de iluminação em subestações, fazendo uso de técnicas de VC na visualização de resultados de análises numéricas de iluminâncias. Técnicas de VC também podem ser aplicadas na visualização de campos elétricos e magnéticos em subestações de energia.

Outro objetivo específico deste trabalho foi realizar um estudo exploratório sobre a ferramenta VTK (*Visualization ToolKit*), instalação, utilização e recursos (classes) com potencial para serem usadas no desenvolvimento de um sistema de RV.

Espera-se que este estudo sirva como referência para trabalhos futuros, uma vez que são descritos alguns obstáculos encontrados e as soluções adotadas. O trabalho foi redigido de forma a orientar o leitor, para que este possa repetir os passos desenvolvidos nesta pesquisa. Algumas instruções para instalação da biblioteca VTK, a descrição detalhada dos algoritmos numérico e de visualização, e alguns códigos estão disponíveis. Também, procurou-se utilizar o máximo, se possível em totalidade, de *softwares* gratuitos ou de código aberto nas implementações realizadas, com objetivo de mostrar a possibilidade de desenvolvimento de sistemas a custos

reduzidos.

1.4 CONTEÚDO DA DISSERTAÇÃO

No capítulo 2 descrevem-se conceitos sobre VC apresentando algumas definições além de uma breve retrospectiva histórica do surgimento de algumas técnicas comumente utilizadas. Além disso, são apresentadas as etapas que constituem o processo de visualização e dois modelos utilizados em softwares de visualização. Por fim, lista-se algumas ferramentas computacionais utilizadas na visualização de dados científicos.

No capítulo 3 apresentam-se algumas técnicas de VC utilizadas para representação de campos escalares, vetoriais e tensoriais. Alguns dos fundamentos básicos de técnicas como mapeamento de cores e extração de isocontornos, para dados 2D e 3D, são descritos. Algumas outras técnicas mais recentes também são citadas e referenciadas.

No capítulo 4 descreve-se a tecnologia da RV, algumas características básicas, e dispositivos utilizados. Conceitos como interação e imersão são comentados bem como o formato VRML e seu sucessor, o X3D. A abordagem utilizada procura relacionar esta tecnologia com a visualização de grandes conjuntos de dados, sejam eles modelos CAD, dados científicos ou a combinação de ambos, e são citadas algumas aplicações.

O capítulo 5 dedica-se à ferramenta VTK (*Visualization ToolKit*) com a qual foram desenvolvidos os algoritmos descritos no capítulo 7. Apresenta-se esta ferramenta, algumas características técnicas, arquitetura e formatação de arquivo de dados para visualização.

No capítulo 6 relatam-se algumas possibilidades oferecidas por um sistema de visualização aplicado a atividades de uma subestação de energia elétrica. São descritos os procedimentos e programas utilizados para realizar a navegação no modelo 3D de uma subestação, as funcionalidades desenvolvidas para o processamento e pós-processamento de campos de iluminância, e também a viabilidade de uso de RV na

visualização de campos elétricos e magnéticos atuantes na subestação.

As implementações realizadas são relatadas no capítulo 7. Descreve-se detalhadamente o algoritmo desenvolvido para o processamento dos campos de iluminância e os algoritmos de visualização por mapeamento de cores e iso-superfícies. Finalmente, o capítulo 8 apresenta as conclusões do autor.

Como mencionado, à guisa de complementação são colocadas em apêndices informações sobre a instalação e o uso do software VTK e os códigos relativos aos algoritmos, bem como mais algumas imagens obtidas durante o trabalho desenvolvido.

2 VISUALIZAÇÃO CIENTÍFICA

Visualização Científica (VC) é a área da computação dedicada à visualização de dados físicos, ou científicos, geralmente provenientes de medições ou simulações numéricas, fazendo uso de Processamento de Imagens (PI) e Computação Gráfica (CG). Algumas técnicas de VC, cujos primeiros registros datam do século XII, são utilizadas até os dias de hoje, para visualização de grandes conjuntos de dados complexos, e são implementadas em muitas ferramentas computacionais para VC disponíveis atualmente.

Neste capítulo apresentam-se algumas definições e fundamentos básicos da VC. Faz-se também uma breve retrospectiva histórica sobre o surgimento de algumas técnicas e descrevem-se as etapas que constituem o processo de visualização bem como os modelos de fluxo de dados e orientado a objetos, presentes em sistemas de visualização. Por fim, são citadas algumas ferramentas computacionais de VC existentes, juntamente com breves comentários.

2.1 VISÃO GERAL E DEFINIÇÕES

Como citado por Schroeder et al. (2004), informalmente, visualização é a transformação de dados ou informações em imagens, estimulando assim, o principal sentido humano, a *visão*. Visualização Científica (VC) é a aplicação deste processo na visualização de dados científicos (VELHO e GOMES, 2001). O objetivo da VC é promover um nível mais profundo de entendimento dos dados sob investigação, confiando na habilidade poderosa dos humanos de visualizar (BRODLIE et al., 1992 apud ADAIME, 2005, p. 4). Assim, técnicas e ferramentas de visualização têm sido usadas para analisar e mostrar grandes volumes de dados multidimensionais, freqüentemente variantes no tempo, de modo a permitir ao usuário extrair características e resultados rápida e facilmente.

Uma forma de classificar as diferentes tecnologias de visualização de dados é examinar o contexto no qual eles estão inseridos. Se os dados estão no mesmo

“espaço-tempo” da natureza, ou seja, no espaço 3D mais a dimensão “tempo”, então tipicamente são usadas técnicas de VC. Se os dados existem em um espaço com muitas dimensões, ou em espaços abstratos, métodos de visualização de informações são utilizados (SCHROEDER et al., 2004).

Encontram-se aplicações da VC em muitas áreas do conhecimento: medicina, astronomia, meteorologia, engenharias e ciências em geral. Como exemplo, tem-se a visualização de dados provenientes de tomografias e ressonância magnética, animações de tornados, e pós-processamento de análises numéricas. Um caso de aplicação de VC em engenharia é a visualização de resultados obtidos por métodos numéricos (Método dos Elementos Finitos, Métodos dos Volumes Finitos, Métodos dos Elementos de Contorno, etc) (ADAIME, 2005).

Pode-se dizer que a VC é uma forma de comunicação computacional que consiste na transformação de dados, estáticos ou variantes no tempo, em representações que reflitam a informação neles contida, de forma eficiente e precisa. Sua importância está relacionada à evolução ocorrida nas tecnologias e sistemas de aquisição de dados (simulações numéricas, sensores, etc), que possibilitam cientistas e engenheiros produzirem enormes conjuntos de dados.

Algumas outras definições formais para VC podem ser encontradas. Para McCormick et al. (1987), VC é o uso de CG para criar imagens visuais que ajudam na compreensão de conceitos científicos ou resultados complexos, freqüentemente associados a representações numéricas volumosas. Wolff e Yaeger (1993, apud TAVARES, 2005), de forma muito parecida, escreveram que VC é a utilização de técnicas de CG e de processamento de imagens, para representar um conjunto de dados visualmente.

Embora a VC seja usada para, de forma visual, abstrair informações a partir de um conjunto de dados físicos, nem sempre ela é utilizada de forma eficiente. Segundo Globus e Raible (1994), a VC pode ser usada para produzir bonitas imagens, mas nem sempre essas imagens são efetivamente capazes de transmitir informações científicas relevantes. A omissão de escala de cores e suavização de geometrias, por exemplo, podem levar uma imagem, gerada com técnicas de VC, a não dizer nada

cientificamente.

Diferentes técnicas de VC podem ser mais ou menos convenientes, dependendo da natureza dos dados a serem representados (número de variáveis dependentes) e da dimensão do domínio em que se encontram (número de variáveis independentes). Por exemplo, para visualização de campos de escalares definidos em um espaço 3D, pode-se usar técnicas de extração de isosuperfícies, ou então *rendering* de volumes. Já para de campos vetoriais em 3D, técnicas como *streamlines*, baseadas em texturas ou em ícones, podem ser mais adequadas. O autor Souza (2003) escreveu que a forma de visualização mais utilizada para campos escalares é o *rendering* volumétrico. No entanto, são as técnicas de reconstrução de superfícies que se adaptam melhor às filosofias de interação, presentes nos ambientes virtuais como aqueles utilizados em RV.

A visualização computacional 3D de campos escalares, vetoriais e tensoriais, representa uma poderosa ferramenta de apoio na análise e compreensão de grandes conjuntos de dados. No entanto, a tecnologia envolvida nos sistemas de visualização encontra-se atualmente atrás das tecnologias de obtenção de dados, em uma escala crescente de evolução, fazendo com que, em alguns casos, sejam gastos muitos dias para compreender conjuntos de dados obtidos em poucas horas (KAGEYAMA e OHNO, 2005).

O processo de visualização de dados passa necessariamente por três passos fundamentais, que são: a aquisição dos dados, a transformação em uma forma apropriada para representação e a “renderização” (*rendering*) ou representação na tela do monitor ou em outro *display* (ou superfície de visualização) (SCHROEDER et al., 2004). As técnicas de visualização envolvem, portanto, algoritmos de processamento de dados que extraem os dados de interesse da amostra e os convertem em uma forma adequada para representação. Algumas formas de representação implementadas em algoritmos de visualização existem desde muitos anos antes do surgimento do computador.

A seguir apresenta-se uma breve retrospectiva histórica que cita alguns dos primeiros registros de técnicas de VC utilizadas até os dias do hoje.

2.2 BREVE RETROSPECTIVA HISTÓRICA

Segundo McCormick et al. (1987, apud Schroeder et al., 2004, p.5) e McCormick et al. (1987, apud Collins, 1993, p. 4), o painel *Visualization in Scientific Computing* do SIGGRAPH, é reconhecido como o marco inicial para os inúmeros *workshops*, conferências, livros e jornais especializados que foram surgindo neste "novo" campo da visualização de dados. Desde então, a VC é reconhecida como uma disciplina formal, com a presença de grandes conferências específicas em VC ou mesmo as de CG que dedicam parte de suas programações a temas relacionados com sistemas de visualização.

Dois períodos que anteriores, também foram marcados pelo uso de representação gráfica de dados numéricos. O primeiro teve início na metade do século XVII e durou até o início do século XX. Neste período técnicas de representação visual de dados foram desenvolvidas pelos grandes cientistas europeus como Halley, Watt, Descartes, Lambert, Playfair e von Humboldt. O segundo período teve início nos anos 60, logo após a introdução dos computadores, e novas aplicações em química, biologia e medicina passaram a utilizar técnicas desenvolvidas no primeiro período (COLLINS, 1993). As técnicas de VC conhecidas atualmente podem ser consideradas uma evolução das técnicas desenvolvidas antes da popularização dos computadores.

Como cita Collins (1993), no ano de 1137 os chineses já utilizavam um sistema de coordenadas ortogonais na confecção de mapas cartográficos. Em 1603, Johann Beyer introduziu as coordenadas esféricas, e em 1637, Rene Descartes reintroduziu as coordenadas ortogonais (cartesianas) na matemática. O traçado de linhas para representação de dados existe desde o século X, quando foi utilizado para ilustrar a inclinação das órbitas planetárias. Em 1663 Christopher Wren construiu um instrumento que mede a direção do vento, nível de chuva e temperatura que é hoje considerada a primeira invenção capaz de produzir automaticamente um gráfico.

Outros trabalhos podem ser citados, como o de Johann Heinrich Lambert, em 1779, que mostra por meio de traçado de linhas, a variação da temperatura do sol

abaixo da superfície terrestre. E em 1785 Wiliam Playfair publicou um trabalho com dados de importação e exportação, entre Inglaterra e América do Norte, representados com linhas coloridas e áreas entre gráficos pintadas. Valores medidos e “plotados” em um gráfico 2D, utilizado para representar a relação entre duas variáveis, foram usados por Edmund Halley, em 1686, para mostrar a relação entre a pressão atmosférica e a altitude. James Watt em 1764, mostrou com esta técnica a relação entre a pressão e a temperatura do vapor (COLLINS, 1993).

Em 1594, Pieter Bruinsz publicou o desenho de um rio, em que marcou com pontos os locais de mesma profundidade. Hoje, a técnica de se unir pontos de mesmo valor em um domínio de dados é conhecida como traçado de isolinhas, ou de forma generalizada, extração de isocontornos. Em 1641 Athanasius Kircher criou um mapa de isolinhas para mostrar características magnéticas em torno da Terra. Mas o primeiro trabalho “popularizando” o uso de isolinhas, segundo Collins (1993), foi o de Edmund Halley, em 1701, também descrevendo grandezas magnéticas.

Em 1741, Gottfried Hessel produziu mapas usando cores para representar regiões com populações que falam a mesma língua. O primeiro mapa que usa cores para distinguir diferentes formações rochosas foi criado em 1775, por Gottlob Glaser. Desde então, muitos trabalhos foram realizados utilizando o mapeamento por cores na representação de dados.

A representação de dados 2D através de uma superfície elevada (3D) foi usada em 1879, por Luigi Perozzo. O desenho foi chamado de “*stereogram*” e representava dados de população. A primeira representação de volumes foi gerada por Elihu Thompson em 1896, com um par de imagens estereográficas obtidas por meio de raios-X de um rato, apenas cinco meses após a invenção do raio-X. Linhas de fluxo (*streamlines*) foram usadas em 1665, por Athanasius Kircher, para representar correntes marítimas. Em 1686, Edmund Halley publicou um mapa mundial dos ventos oceânicos. Em 1883, Osborne Reynolds conduziu experimentos em que se representou, por meio de traços de partículas, o processo de transição de um escoamento laminar para turbulento por injeção de fluido em um tubo. Ícones foram usados por August Crome, em 1782, para mostrar a distribuição de produtos na

Europa. Em 1973, H. Chernoff utilizou expressões faciais para representar variáveis de uma terceira, quarta, etc, dimensões, em um plano bidimensional (COLLINS, 1993).

Collins (1993) finaliza seu artigo citando que apesar da visualização de dados ser considerada um produto da década de 80, muitas técnicas de representação de dados foram implementadas durante os anos 60, com a aplicação de computadores para visualização em cristalografia química. Schroeder et al. (2004) argumenta que as primeiras representações pictóricas datam do século XVIII, juntamente com o surgimento dos gráficos estatísticos, mas somente com a chegada dos computadores digitais e da evolução da CG que a visualização se tornou uma disciplina prática.

2.3 O PROCESSO DE VISUALIZAÇÃO POR COMPUTADOR

O processo de visualização por computador envolve etapas distintas e algumas delas podem ser agilizadas com o uso de bibliotecas gráficas e/ou ferramentas (*toolkits*) de visualização. Bibliotecas gráficas fornecem funções que, fazendo interface com o *hardware* gráfico, facilitam a tarefa de geração das imagens no monitor ou em outro dispositivo (AVEVEDO e CONCI, 2003). As ferramentas de VC fornecem um conjunto de algoritmos de visualização que e fazem interface com a biblioteca gráfica utilizada, auxiliando na tarefa de transformar os dados e também apresentá-los no dispositivo de saída.

O desenvolvimento de ferramentas de VC pode abordar as etapas envolvidas no processo de visualização por meio de diferentes métodos. Um método que tem sido comumente utilizado, tornando-se um paradigma, é o método chamado de modelo de fluxo de dados (WALTON, 1993).

No item 2.3.1 descrevem-se as etapas básicas envolvidas no processo de visualização de dados. Nos itens 2.3.2 e 2.3.3 descrevem-se o modelo de visualização por fluxo de dados e o modelo orientado a objetos, respectivamente.

2.3.1 Etapas do processo de visualização

O processo de visualização envolve as etapas de obtenção dos dados, transformação ou processamento, e a apresentação dos resultados em um dispositivo de saída, como por exemplo, o monitor.

A obtenção dos dados pode ser feita de várias formas, mediante simulações numéricas, captura ou medição utilizando algum equipamento especial, como, por exemplo, levantamento a laser (*laser scanning*), ou simplesmente através da leitura de um arquivo. A transformação (ou processamento) dos dados é feita mediante de algoritmos de visualização, como filtros e mapeadores, que convertem os dados em um formato conveniente para a visualização. Por fim, a etapa de representação envolve técnicas de *rendering*, que consistem na basicamente a maneira com que informações digitais serão mostradas no monitor (ou em outro dispositivo). Técnicas de *rendering* podem ser divididas em dois grandes grupos que são *rendering* de superfícies e de volumes.

O desenvolvimento de bibliotecas gráficas como OpenGL e DirectX tornou a etapa de representação dos dados mais fácil, uma vez que essas bibliotecas ou APIs (*Application Programming Interfaces*), contêm funções que fazem interface com o *hardware* gráfico, diminuindo a quantidade de linhas de código necessárias para a implementação de aplicativos gráficos. Além disso, como mencionado, muitas placas gráficas comerciais possuem implementações dessas bibliotecas, melhorando o desempenho final (acelerando o processamento de dados gráficos) do aplicativo.

As chamadas ferramentas, ou *toolkits*, de VC provêm rotinas e algoritmos para o processamento e também para visualização dos dados, fazendo interface com a biblioteca gráfica. Exemplos de ferramentas para VC são: AVS[1] (*Advanced Visual Systems*), IRIS[2] Explorer, OpenDX[3] e VTK[4].

Na opinião de Walton (1993), estas ferramentas apresentam vantagens e desvantagens. Uma vantagem é que, com elas, o usuário consegue obter rapidamente

---

[1] <http://www.avs.com>

[2] <http://www.nag.co.uk>

[3] <http://www.opendx.org>

[4] <http://www.vtk.org>

uma visualização, podendo, por exemplo, modificar parâmetros e visualizar novamente seus dados, várias vezes, de forma ágil. Uma desvantagem é que o usuário fica preso às funcionalidades da ferramenta.

2.3.2 Modelo de fluxo de dados

O modelo de processo de visualização chamado “fluxo de dados” (*dataflow*) influenciou o desenvolvimento de diversos sistemas de visualização (BRODLIE, 1993). Nesse modelo o sistema oferece uma variedade de módulos que recebem os dados (*input*), os transformam e reescrevem (*output*). Módulos podem ser selecionados e combinados formando um *pipeline* de visualização. As ferramentas AVS, IRIS Explorer e IBM Data Explorer são exemplos que usam o modelo de fluxo de dados.

O modelo de fluxo de dados teve origem no trabalho de Upson et al. (1989, apud Brodlie, 1993, p. 126), que definiu para o processo de visualização um modelo orientado aos dados, em que os dados são transformados por meio de passos lógicos até a representação final, conforme ilustra a figura 2.

FIGURA 2 - MODELO DE FLUXO DE DADOS

FONTE: Adaptado de Brodlie (1993).

Filtro é o passo onde os dados de interesse são extraídos (planos de corte, isocontornos), o “mapeador” (*mapper*) é o passo onde são construídas as geometrias para representar os dados extraídos, e o último passo, o “render” (ou renderizador) é onde a geometria é convertida em uma imagem e mostrada no monitor ou em outro *display*.

2.3.3 Modelo orientado a objetos

No modelo orientado a objetos (OO), a metodologia de programação é utilizada de forma a explorar propriedades comuns e hierarquias, geralmente presentes no processo de exibição, por meio do uso dos conceitos de polimorfismo e herança. Um exemplo de ferramenta que usa o modelo orientado a objetos é o VTK (BATTAIOLA e SOARES, 1998).

Sistemas OO oferecem a possibilidade de criar sistemas por meio da reutilização de componentes de *sofware*. Assim, várias bibliotecas de classes são disponibilizadas, desde estruturas padrões de dados até *solvers* matemáticos para resolução de equações e sistemas. No "desenho" usual de sistemas OO, as estruturas de dados e os métodos são encapsulados em objetos. Algoritmos (ou métodos) e conjuntos de dados (ou estruturas de dados) são encapsulados separadamente (SCHROEDER et al., 2004).

Segundo Battaiola e Soares (1998), as ferramentas OO oferecem modelos de dados que são definidos de forma extensiva como biblioteca de classes, onde cada classe implementa uma estrutura de dados. A biblioteca pode ser estendida através do uso vantajoso das propriedades de polimorfismo e herança, que também oferecem uma integração natural de módulos de visualização com linguagens de programação para a confecção de algoritmos de VC.

2.4 FERRAMENTAS DE VISUALIZAÇÃO CIENTÍFICA

De acordo com Walton (1993), a visualização de dados proporciona um ganho em compreensão, que é obtido ao se gerar uma imagem a partir de números. Em geral o usuário escreve um programa em uma linguagem de alto nível, como C ou FORTRAN, para ler ou calcular dados e executa processos (ou filtros) transformando-os em outras formas.

O programa pode conter chamadas para uma seleção de rotinas da biblioteca gráfica, que cria imagens a partir dos dados, na forma de linhas, mapas de contornos,

superfícies, volumes, etc. Ou então, o programa pode conter chamadas para funções de alguma ferramenta de visualização científica, por exemplo, o VTK, que processa filtros e mapeadores, e gera a representação da imagem no monitor ou em outro *display*.

Atualmente, muitos *softwares* e ferramentas para VC podem ser encontrados. Em Wikipedia (2006), por exemplo, na página referente ao tema *scientific visualization*, está disponível a seguinte lista, com os respectivos *links*:

HoloDraw - http://holodraw.org/

Open Source - Data Explorer http://www.opendx.org/

The Visualization Toolkit - http://www.vtk.org

Amira - http://www.amiravis.com e http://www.tgs.com

ParaView - http://www.paraview.org

VisIt - http://www.llnl.gov/visit

SCIRUN - http://software.sci.utah.edu/scirun.html

Vis5D - http://www.ssec.wisc.edu/~billh/vis5d.html

VisAD - http://www.ssec.wisc.edu/~billh/visad.html

VMD - http://www.ks.uiuc.edu/Research/vmd/

Matlab - http://www.mathworks.com/

Scilab - http://www.scilab.org/

Spotfire - http://www.spotfire.com/

Cada um destes softwares tem características próprias e nível de abstração diferente. A seguir, são descritas brevemente características de algumas destas e de outras ferramentas, mas sem pretensão de fornecer uma descrição completa, pois não faz parte dos objetivos deste trabalho.

A ferramenta Matlab, por exemplo, pode ser considerada uma linguagem de programação interpretada, uma vez que a concatenação dos comandos dados no espaço de trabalho gera um “programa”, que pode ser armazenado em um arquivo e “carregado” quando desejado (ROQUEIRO, 2006).

O *software* VisAD pode ser considerado um sistema que trabalha por mapeamento (BATTAIOLA e SOARES, 1998). Nele, o usuário pode inserir seus dados e realizar o mapeamento para uma forma adequada à visualização. Consiste em

uma biblioteca de componentes Java para visualização interativa e colaborativa de dados provenientes de análises numéricas (VISAD, 2005).

O OpenDX (*Open Visualization Data Explorer*) é uma versão de código aberto do IBM Visualization Data Explorer. Utiliza o modelo de visualização orientado a objetos, e também fornece um conjunto de ferramentas para manipulação, transformação, processamento, renderização e animação de dados.

Outros *softwares*, além dos listados acima, podem ser encontrados, tais como o AVS (*Advanced Visual Systems*). Segundo Battaiola e Soares (1998), o AVS foi um dos primeiros *softwares* desenvolvidos para atender os requisitos de facilidade de uso, completividade, expansibilidade e portabilidade. Possui uma interface de edição visual orientada a objetos o que permite que usuários "não-programadores" criem aplicações e possam visualizar facilmente seu conjunto de dados (AVS, 2006).

O Amira é um *software* profissional para visualização de dados 3D (AMIRA, 2006). Com ele, malhas volumétricas tetraédricas podem ser geradas, sendo adequado para visualização de resultados de análises por elementos finitos. Uma versão estendida deste *software* é o AmiraVR, especializado na visualização em ambientes imersivos, como grandes telas com projeção estereoscópica, e cavernas digitais conhecidas como CAVE's[5].

Uma ferramenta de construção de gráficos 3D bastante útil é o *software* GnuPlot. Trata-se de um programa portável que possui uma interface baseada em linhas de comando e é capaz de "plotar" dados 2D e 3D, incluindo a extração de contornos, superfícies e campos de vetores. Pode ser utilizado para visualização interativa de dados e funções matemáticas e também ser integrado a outras aplicações não interativas (GNUPLOT, 2006).

Uma ferramenta que vem se destacando no meio científico é o VTK (VTK, 2006). O VTK reúne muitos requisitos desejados, como ser *open source* (código aberto), portável, extensível, muito bem documentado e estar sendo utilizado por

[5] CAVE's, ou cavernas digitais, são salas especiais, em forma de cubo, onde as paredes, teto e piso são grandes telas nas quais são projetadas imagens de um ambiente virtual 3D, proporcionando ao usuário dentro desta sala, sentir-se imerso, ou seja, "dentro" do ambiente virtual .

milhares de pesquisadores e cientistas por todo o mundo. Como conseqüência desse fato, tem-se que muitas classes estão sendo adicionadas e erros são detectados e corrigidos, tornando a biblioteca cada vez mais completa e poderosa.

Neste trabalho, escolheu-se utilizar a ferramenta VTK nas implementações realizadas. Maiores detalhes sobre as características técnicas, instalação e utilização desta ferramenta são descritos no capítulo 4. O capítulo 7 apresenta algumas implementações realizadas que são exemplos de aplicação do VTK, e o apêndice 1 fornece informações úteis para sua instalação na plataforma Win32.

3 TÉCNICAS DE VIZUALIZAÇÃO CIENTÍFICA

Existem muitas técnicas de VC, como por exemplo, mapeamento de cores, isocontornos, linhas de fluxo e representação volumétrica. Algumas podem ser mais adequadas que outras, dependendo do tipo dos dados a serem representados. Dados físicos podem ser classificados como escalares (temperatura, pressão, etc), vetoriais (velocidade, campo magnético, etc) e tensoriais (tensor de tensões, tensor de difusão, etc), e são definidos em um espaço 1D, 2D ou 3D, podendo ser estáticos ou variantes no tempo.

Assim, uma forma de fazer distinção entre as técnicas de VC é separar em técnicas para visualização de dados escalares, vetoriais e tensoriais (HESSELINK, 1994). Também pode ser feita a distinção entre técnicas em função da dimensão das primitivas usadas para representar os dados: pontos, linhas, superfícies e volumes (COLLINS, 1993).

Nessa segunda categoria, a representação de superfícies é geralmente feita por meio da representação geométrica de uma malha de polígonos conectados que formam as superfícies. Em placas gráficas aceleradoras 3D são utilizados triângulos, e a quantidade máxima de triângulos mostrados em uma imagem é usada para avaliar o desempenho desse tipo de *hardware*. Souza (2003) coloca que a representação de superfícies é mais adequada a sistemas interativos, que requerem “renderização” em tempo real, como os sistemas de RV.

Ainda nas técnicas classificadas em função da dimensão das primitivas, na representação de volumes, os dados são tratados como uma matriz de elementos de volume chamados *voxels* (análogo 3D de *pixels*). O *voxel* é a representação de uma vizinhança espacial, sobre a qual determinados atributos e características são associados. A técnica de representar diretamente dados volumétricos, ou seja, sem requerer nenhum tratamento intermediário, é denominada *direct volume rendering.*

Técnicas de “renderização” híbrida usando volumes e superfícies também podem ser utilizadas. As estratégias de representação híbrida são classificadas de acordo com a necessidade ou não de converter os dados para uma única forma de

representação, (SCHIMIDT, 2000).

Uma taxonomia para as técnicas de VC é proposta por Brodlie (1992). Entre as técnicas descritas, tem-se, para representar dados escalares, os gráficos de funções de uma variável (*linegraph*), extração de contornos (*contours*), *rendering* de volumes, superfície elevada, e planos de corte (*slicing*). Para representar dados vetoriais, tem-se campos de setas (*arrow plots*), glifos (*glyphs*), trajetória de partículas (*particle tracing*), linhas de fluxo (*stream lines*) e texturas. Algumas dessas técnicas podem ser estendidas e/ou combinadas a fim de representar dados tensoriais. Exemplos de técnicas para representação de tensores são glifos e *hiperstreamlines*. Os dados também podem variar com o tempo. Nesse caso, o uso de animação pode ser conveniente (ENCARNAÇÃO, 1993).

Nos itens 3.1 até 3.3, são descritas algumas técnicas de representação de dados escalares, vetoriais e tensoriais, e conceitos relacionados. Este trabalho não lista todas as técnicas existentes tampouco detalha os algoritmos envolvidos nas técnicas apresentadas. Um conhecimento mais aprofundado das técnicas apresentadas pode ser obtido consultando as referências apresentadas.

3.1 VISUALIZAÇÃO DE CAMPOS ESCALARES

Um escalar é um componente que assume um único valor de uma escala, por exemplo, temperatura. Um campo de escalares é um arranjo de valores escalares distribuídos em um espaço 1D, 2D ou 3D.

Algumas técnicas de visualização associam os valores escalares a uma dimensão adicional, fazendo com que a representação gráfica tenha uma dimensão a mais que o espaço no qual os dados estão contidos. Por exemplo, um campo de escalares 1D pode ser representado por uma curva plana 2D. Um campo de escalares 2D pode ser representado como uma superfície no espaço 3D (COLLINS, 1993). Outras técnicas associam cores de uma escala aos valores escalares, mantendo assim a dimensão do espaço em que os dados estão definidos.

A visualização de dados definidos em um domínio 3D, ou dados volumétricos,

pode envolver algoritmos e técnicas objetivando explorar o interior desse volume. Para isso, pode ser usado *rendering* de volumes mapeados com diferentes níveis de transparência. Também é comum a extração de geometrias definindo isocontornos, como isolinhas (caso 2D) e isosuperfícies (caso 3D). O mapeamento de cores é utilizado tanto em *rendering* de superfícies quanto de volumes.

3.1.1 Campos escalares 2D

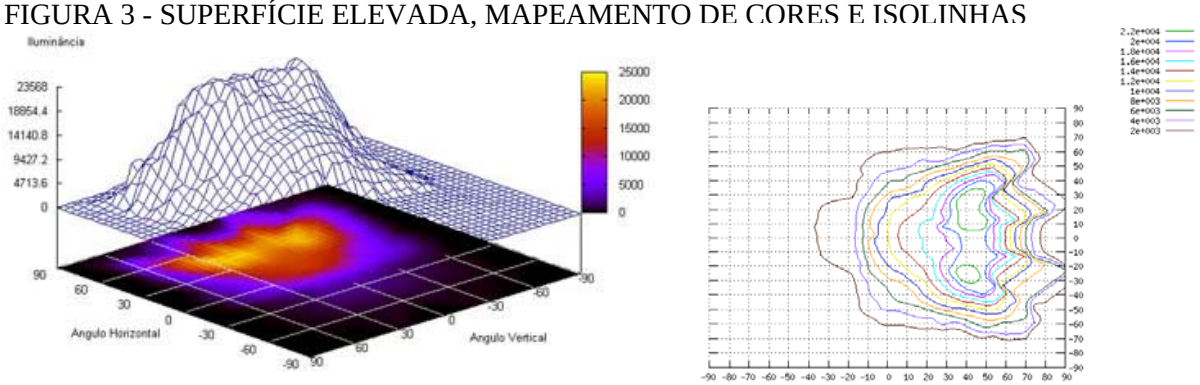
Entre as formas de representar escalares, a mais óbvia é por meio de cores (BLYTHE, 1999). A técnica de mapeamento por cores consiste na associação de cores a valores numéricos escalares. Os modelos RGB (*Red*, *Green*, *Blue*, ou vermelho, verde e azul) e HSV (*Hue*, *Saturation*, *Value*, ou matiz, saturação e intensidade) podem ser usados para a definição da escala de cores (PAVÃO et al., 2001; VELHO e GOMES, 2001). Uma maneira de inserir as cores na representação por superfícies é indexar as cores aos vértices, ou seja, diretamente nos pontos, outra forma é por meio do mapeamento de texturas, o que proporciona um ganho de desempenho na renderização. No caso de renderização de volumes, as cores são aplicadas às células (*voxels*) e diferentes níveis de transparência podem ser usados.

Na figura 3, à esquerda, são ilustradas duas técnicas de representação de campos escalares em uma mesma imagem. Os dados originais consistem em valores escalares definidos sobre um plano e estão representados por meio de uma superfície elevada (3D), ou *heigh map*, e de mapeamento de cores, ou *color map*, no plano horizontal (2D). A superfície elevada é gerada pelo mapeamento dos valores escalares para uma terceira dimensão e o mapeamento de cores relaciona os valores escalares a valores de uma escala de cores, mantendo assim a dimensão (2D) dos dados, na representação gráfica. Na figura 3 à direita, é representada a técnica de extração de isolinhas, que consiste em traçar curvas passando pelos pontos de igual valor.

No caso da amostra de dados ser volumétrica, ou seja, estar definida em um domínio 3D, o mapeamento de cores pode ser feito na superfície externa, nas células *voxels*, ou então, em planos de corte (*slices*), que correspondem ao mapeamento de

cores em planos (2D). No caso 3D, portanto, a visualização do interior do volume de dados pode ser feita utilizando-se técnicas de extração de isosuperfícies ou renderização de volumes.

FIGURA 3 - SUPERFÍCIE ELEVADA, MAPEAMENTO DE CORES E ISOLINHAS



FONTE: O autor, utilizando a ferramenta GnuPlot <http://www.gnuplot.info>.

3.1.2 Campos escalares 3D

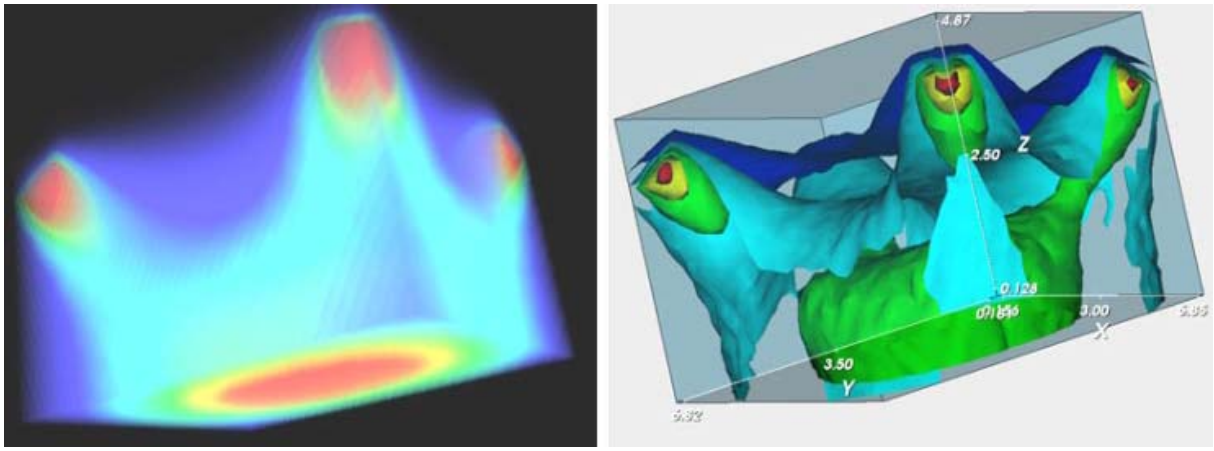
Tratando-se de dados volumétricos, a visualização do interior do volume pode ser feita por meio da extração de isosuperfícies e também por *rendering* de volume. Outra técnica é a extração de planos de corte (*slices*) seguido de mapeamento de cores ou extração de isolinhas no plano obtido.

A renderização de superfícies, como já foi dito, apresenta melhor desempenho em relação a renderização de volumes, sendo mais adequada para utilização em sistemas de *rendering* em tempo real. Isto se dá pelo fato de que o algoritmo de *Z-Buffer* (FOLEY, 1990) utilizado é um dos mais flexíveis para o tratamento de visibilidade de modelos poligonais, sendo implementado por bibliotecas gráficas como OpenGL e Direct3D e cujas funções podem ser aceleradas em uma placa gráfica. Dessa forma, a utilização de isosuperfície pode ser a solução mais adequada para visualização de campos escalares 3D em sistemas interativos. Na figura 4, à esquerda, é ilustrada a representação por *rendering* de volume e à direita por *rendering* de superfícies.

Em *rendering* de superfícies, o mapeamento de cores pode ser feito aplicando

as cores aos vértices dos polígonos (*index color*), geralmente triângulos, que formam as superfícies. No entanto, segundo Blyte (1999), o uso de *index color* é limitado sendo o mapeamento de textura a melhor maneira de atribuir cores aos valores escalares.

FIGURA 4 - *RENDERING* DE VOLUME E ISOSUPERFÍCIES



FONTE: O autor

Planos de corte são planos que atravessam o volume de dados e apresentam (por mapeamento de cores, isolinhas, etc) os valores escalares “sobre os planos”, ou seja, os valores que estão nos pontos do espaço que coincidem com cada plano. O volume de dados a ser visualizado não é contínuo, mas sim mostrado em cada *voxel*. Por esta razão interpolações são freqüentemente utilizadas para determinar valores não coincidentes com o centro dos *voxels*.

As isosuperfícies são superfícies do espaço 3D formadas a partir de pontos do volume de dados que possuem o mesmo valor.

Segundo Manssour e Freitas (2002), dentre os algoritmos de visualização de dados volumétricos por extração de superfícies, destacam-se: conexão de contornos (*boundary tesselation*) e cubos marchantes (*marching cubes*). As vantagens dessa técnica são: a velocidade para a geração e exibição da imagem final, e o pouco espaço de armazenamento requerido. Representações desse tipo são apropriadas quando existem isosuperfícies bem definidas nos dados, mas não são eficientes quando o volume é composto por muitas microestruturas, tais como os tecidos humanos em

muitas imagens médicas.

A visualização de volumes por utilização da técnica conhecida como *direct volume rendering* (visualização direta de volume) consiste em representar o volume por meio de *voxels*, que são projetados diretamente em *pixels* e armazenados como uma imagem, dispensando o uso de primitivas geométricas (MANSSOUR e FREITAS, 2002). Em outras palavras, os dados escalares são classificados a partir de uma função de transferência, que mapeia valores do campo escalar 3D em cor e opacidade, para serem visualizados na tela 2D (ESPINHA, 2005).

Alguns algoritmos envolvidos na renderização direta de volumes são *ray casting* (LEVOY, 1988 e 1990), *splatting* (WESTOVER, 1989 e 1990), *shear-warp* (LACROUTE e LEVOY, 1994), *shell rendering* (UDUPA e ODHNER, 1993), *cell-projection* (WILHELMS e GELDER, 1991) e *V-Buffer* (UPSON e KEELER, 1988).

A visualização direta de volumes pode ser feita de três formas, (KAUFMAN, 1991 apud MANSSOUR e FREITAS, 2002): *object-order* (ou *forward-projection*), que envolve o mapeamento de amostras de dados no plano da imagem; *image-order* (ou *backward-projection*), que determina para cada *pixel* do plano da imagem, quais são as amostras que contribuem no cálculo da sua intensidade; e *domain-based*, quando os dados 3D são transformados para outro domínio, como freqüência ou *wavelet*.

Recentemente, tem se popularizado o uso de mapeamento de textura 3D por *hardware* para a visualização de volumes. Uma estratégia adotada para possibilitar a visualização interativa de volumes é a utilização de *hardware* dedicado.

3.2 VISUALIZAÇÃO DE CAMPOS VETORIAIS

Visualizar campos de vetores não é uma tarefa trivial. Enquanto campos de escalares contêm um valor por ponto, campos vetoriais admitem múltiplos valores em cada ponto, geralmente 2 (domínio 2D) ou 3 (domínio 3D).

Algumas vezes pode ser conveniente simplificar os dados de forma a representar somente componentes escalares de vetores, por exemplo, representar um

campo vetorial mostrando apenas o módulo do vetor, desconsiderando a sua direção. Neste caso técnicas de visualização de campos escalares são usadas (COLLINS, 1993).

Segundo Crawfis et al. (1994), técnicas foram desenvolvidas objetivando o uso de texturas para representar coerentemente o comportamento geral de campos vetoriais, (CRAWFIS e MAX, 1992), (CABRAL, 1993). Outras técnicas, como *stream polygons* (SCHOEDER et al., 1991), *spot noise* (WIJK, 1991), *stream surfaces* (HULTQUIST, 1992) e ícones (KERLICK, 1990), são consideradas extensões das técnicas de traços de partículas e *stream lines*, e utilizam a renderização de primitivas geométricas.

Blythe (1999) agrupou as técnicas de visualização de campos de vetores em três grandes classes: baseadas em ícones, baseadas na trajetória de partículas e baseada em texturas. Abordagens mais recentes (ESS e SUN, 2006) classificam as técnicas em apenas dois grandes grupos: baseadas em geometria e em textura.

Em Groen (2006) encontra-se uma proposta de taxonomia para diversas técnicas de visualização, separadas em função do tipo de dados e do domínio em que se encontram. Entre as técnicas listadas para visualização de campos de vetores, estão: “plotagem” de setas (*arrows plots*), traço de partículas, *streamlines*, *streaklines*, *stream polygon*, *glyphs*, *stream surfaces*, *tubes* e outras.

3.2.1 Campos vetoriais 2D

A visualização de campos de vetores 2D é relativamente simples, uma vez que, as ambigüidades e o congestionamento visual que acorrem na visualização de campos de vetores 3D não ocorrem no caso 2D.

Técnicas baseadas em ícones associam um objeto geométrico em cada ponto da amostra, com a sua geometria alinhada com o sentido do vetor nesse ponto. Outros atributos, tais como o tamanho ou a cor do objeto, podem ser usados para representar uma quantidade escalar, tal como, o tamanho do vetor em cada ponto da amostra (BLYTHE, 1999).

Técnicas baseadas em traço de partículas usam primitivas gráficas para traçar “caminhos” definidos pelo campo de vetores. Porções do campo são mostradas com partículas e suas trajetórias; a posição de cada partícula ao longo do tempo também pode ser animada a fim de representar o fluxo no campo vetorial. Variações dessa técnica podem ser obtidas com o desenho de linhas (*stream/streaklines*), fitas (*ribbons*) ou tubos (*tubes*).

Collins (1993) explica que uma *streamline* é uma linha que define a trajetória de uma partícula, passando por suas diferentes posições, em um certo número de instantes, *streaklines* são semelhantes às *streamlines*, a diferença é que são linhas formadas por pontos em posições discretas.

Técnicas baseadas em texturas usam algoritmos de processamento de imagens para traçar as trajetórias definidas pelo campo vetorial. Essas técnicas fornecem bons resultados para visualização de campos vetoriais 2D ou secções retas de campos 3D, mas são insuficientes para visualizar diretamente campos vetoriais 3D (ESS e SUN, 2006).

O trabalho de Laidlaw et al. (2005), apresenta resultados de uma comparação entre técnicas utilizadas na visualização de campos vetoriais 2D. As técnicas avaliadas são: visualização por campo de setas (ícones), visualização por linhas das curvas integrais, visualização por arestas que definem as linhas de fluxo e *line-integral convolution* (LIC). O trabalho apresenta as vantagens e desvantagens de cada técnica.

3.2.2 Campos vetoriais 3D

Técnicas de visualização baseadas em geometrias (ícones, *streamlines*, etc), apresentam algumas limitações quando se trata de um domínio 3D. Como explica Adaime (2005), comumente ocorre um congestionamento visual, pois os ícones ou linhas se sobrepõem de forma desorganizada devido à projeção na imagem 2D final, tornando impossível extrair a estrutura intrínseca do campo vetorial.

Técnicas de visualização interativas têm sido desenvolvidas objetivando superar estas dificuldades. O trabalho de Helgeland e Andreassen (2004) propõe uma

técnica chamada *Seed LIC* que usa renderização de volumes e sombreamento para visualização de campos vetoriais 3D via LIC. O trabalho de Ess e Sun (2006) apresenta uma técnica recente para visualização interativa de campos vetoriais 3D, denominada *streamline splatting*. A técnica integra a geração de *streamlines* com o método de renderização de volumes chamado *splating*, no qual o volume é representado em termos de uma série de texturas definidas sobre planos de corte ortogonais à direção da vista principal.

No trabalho de Krüger et al. (2005), é descrita uma técnica baseada em sistemas de partículas, para visualização interativa de campos de fluxo 3D em malhas uniformes, onde são exploradas as características de *hardwares* gráficos. Em Shen et al. (2004), é apresentada uma técnica de visualização interativa de campos vetoriais 3D, baseada em textura, em que controles em tempo real da aparência da textura são implementados.

3.3 VISUALIZAÇÃO DE CAMPOS DE TENSORES

Técnicas de visualização de campos tensoriais são, em geral, mais sofisticadas e podem envolver a combinação de técnicas de visualização de campos escalares e vetoriais. Algumas vezes, explica Collins (1993), tensores de ordem 3 podem ser simplificados para uma ordem mais baixa (ordem 2) e um tensor de ordem 2 (tensor de tensões, por exemplo) pode ser simplificado para um vetor. Porém essas simplificações acarretam em perdas de informações ou na impossibilidade de visualizar o campo de tensores de forma global.

O desenvolvimento de técnicas para visualização de tensores é um desafio na VC, pois é difícil representá-los em uma simples imagem bidimensional. Uma alternativa é a utilização de glifos (*glyphs*) que são objetos 3D cuja geometria e atributos, como cor, direção, comprimento, e eixos representam os componentes do tensor (COLLINS, 1993).

Segundo Collins (1993), existem pelo menos três glifos que podem ser usados para representar tensores. O mais simples é o composto por três setas de cabeça dupla

ortogonais entre si, que representam os elementos de um tensor diagonal. O valor escalar é mapeado para o comprimento de cada seta e a direção da seta mostra o sinal. Outro é o elipsóide de Lamé, em que os elementos da diagonal são mapeados para os raios do elipsóide. E por fim, tem-se o *shaft-and-disk* em que a direção de achatamento, cor e comprimento de um disco representam os elementos do tensor.

Kindlmann (2004) descreve os glifos como uma técnica de visualização que converte um tensor em um objeto gráfico por meio do mapeamento dos seus autovetores para a orientação e autovalores para parâmetros da forma de uma primitiva geométrica que pode, por exemplo, ser um cubóide ou elipsóide. Segundo este autor, o uso de uma técnica chamada *superquadric tensor glyphs* elimina o problema de assimetria e de ambigüidade, presentes na representação por cubóides e elipsóides respectivamente.

Já as chamadas *hiperstreamlines* são construídas por meio da criação de *streamlines* em torno de uma das três direções do tensor, e então, realiza-se a varredura de uma primitiva geométrica ao longo das *streamlines* (DELMARCELLE e HESSELINK, 1993 apud SCHOEDER, 2004). Geralmente uma elipse é usada como a primitiva geométrica onde os dois autovetores restantes definem o maior e o menor eixo da elipse. Uma descrição mais detalhada pode ser obtida em Delmarcele e Hesselink (1993).

O trabalho de Zhang et al. (2003) descreve o uso de *streamtubes* e *streamsurfaces* na visualização 3D de tensores de difusão. A direção, o sentido, a forma e a cor da seção transversal representam informações sobre o tensor. Também é apresentado um ambiente virtual imersivo para visualização interativa.

Um tensor de segunda ordem, simétrico, como o tensor de tensões, pode ser visualizado por meio da definição de um elipsóide cujos parâmetros são relacionados com seus autovetores. O comprimento dos autovetores é determinado pelos autovalores correspondentes (SCHOEDER, 2004).

Outras técnicas para visualização de tensores, como *tensorlines* (WEINSTEIN et al., 1999), *tensor splats* (BHALERAO e WESTIN, 2003), *ellipsoids* (WESTIN, 2002), *noise field* combinado com glifos (SIGFRIDSSON, 2002), podem ser

encontradas nos respectivos trabalhos. Em Burkhard (2006) encontra-se uma compilação das técnicas mais populares para representar tensores.

4 AMBIENTES VIRTUAIS COMO INTERFACE PARA VISUALIZAÇÃO CIENTÍFICA

A utilização de sistemas gráficos 3D interativos tem trazido mudanças profundas no método de trabalho de engenheiros e pesquisadores em todo o mundo, uma vez que permite a manipulação e exploração de conteúdos 3D, fornecendo ao usuário, um meio de obter uma percepção e compreensão maior do material em estudo.

A tecnologia de Realidade Virtual (RV) pode ser considerada a mais avançada interface com o usuário conhecida, sendo capaz de fornecer ao usuário a sensação de imersão em ambientes virtuais por meio do uso de dispositivos especiais de entrada e saída de dados. Dessa maneira, o uso de RV na construção de interfaces para Visualização Científica (VC) pode ser uma escolha adequada para o desenvolvimento de sistemas de visualização (BRYSON, 1993).

Neste capítulo, são apresentados alguns conceitos e definições relacionados com a tecnologia de RV, são citados alguns dispositivos de entrada e saída de dados utilizados em RV e, por fim, reúnem-se algumas informações sobre o formato padrão VRML (*Virtual Reality Modeling Language*) e seu sucessor, o X3D.

4.1 DEFINIÇÃO E ASPECTOS GERAIS

Nos últimos anos, a utilização da computação gráfica para visualização científica e modelagem geométrica 3D tem trazido mudanças fundamentais aos métodos de trabalho, tanto na indústria como em pesquisa e desenvolvimento. A possibilidade de visualização e manipulação interativa de modelos virtuais com auxílio do computador tem revolucionado muitas atividades, pois permite a compreensão e análise de enormes quantidades de informação de natureza espacial com eficiência sem precedentes, explorando a grande capacidade humana de raciocinar e se comunicar visualmente (RAPOSO et al., 2004). Como exemplos deste fato pode-se citar o trabalho de Russo et al. (2004), que faz um panorama do uso de RV na indústria

brasileira de petróleo, e as obras de Velho e Gomes (2001) e Watt (2000), sobre computação gráfica 3D; tem-se, ainda, o trabalho de Leng (2001) que cita diversas aplicações científicas para RV.

Em RV a imersão é gerada por meio de estímulos aos múltiplos sentidos humanos e, também, da interação em tempo real com o ambiente e objetos virtuais. Sendo a visão o principal sentido humano, a visão 3D (estereoscópica) é um dos mais importantes recursos para causar a sensação de imersão. Outro recurso é a interação intuitiva com o ambiente, utilizado o conhecimento natural que o homem tem do mundo físico em que vive. A visão 3D e a interação intuitiva em sistemas de RV são possíveis pelo uso de dispositivos não convencionais de entrada e saída de dados, como por exemplo, óculos ou capacetes para visão 3D (conhecidos como HMD – *Head Mounted Displays*) e luvas de dados (conhecidas como *Data Gloves*).

As aplicações que utilizam a tecnologia de RV com finalidade de visualizar dados científicos são diversas e vão desde sistemas que demandam poucos recursos tecnológicos e que podem ser acessados pela *web*, até sistemas mais avançados que são executados em plataformas específicas (LENG, 2001). O texto de John e Leng (2001) descreve atividades na área de medicina em que a visualização de dados provenientes de escaneamentos de pacientes (tomografias) é usada em treinamentos médicos, planejamento e simulação de cirurgias. Também cita a área de Mecânica dos Fluidos Computacional, amplamente utilizada em projeto de aviões, carros e outros veículos, sendo comercialmente importante uma vez que reduz drasticamente o tempo de desenvolvimento de protótipos e melhora a qualidade aerodinâmica do veículo, que pode ser prototipado virtualmente. Aplicações em modelagem química e molecular, ciências da terra e extração de petróleo também são citadas.

Apesar das aplicações serem numerosas, não existe ainda um consenso sobre a definição formal para Realidade Virtual. Alguns autores listam elementos-chave de RV, tais como imersão, interatividade e *feedback* sensorial (SHERMAN e CRAIG, 2003, apud RAPOSO et al., 2004), e outros, descrevem RV como uma técnica avançada de interface, onde o usuário pode navegar e interagir em um ambiente sintético tridimensional, estando completa ou parcialmente imerso pela sensação

gerada por canais multisensoriais, sendo o principal a visão (BURDEA e COIFFET, 2003).

4.2 DISPOSITIVOS DE ENTRADA E DE SAÍDA DE DADOS

Como colocado anteriormente, um dos elementos chaves da RV é o estímulo aos múltiplos sentidos humanos com o objetivo de causar ao usuário a sensação de imersão; outro elemento chave é a interatividade (BURDEA e COIFFET, 2003). Para proporcionar interatividade e imersão, alguns dispositivos não convencionais de entrada e saída de dados, como óculos para visão 3D e luvas especiais são necessários. Um exemplo de uma plataforma simples para um sistema de RV pode ser, por exemplo, composta por um computador do tipo PC, *softwares* e base de dados adequados (como um navegador e "mundos VRML"), um HMD (*Head Mounted Display*) e uma luva de dados (*Data Glove*), por exemplo.

4.2.1 Dispositivos de entrada de dados

Dispositivos de entrada de dados muito utilizados são o *mouse* e o teclado do computador. No entanto, pesquisadores e desenvolvedores que trabalham com RV, procuram formas mais rápidas e naturais de comunicação com a máquina (BURDEA e COIFFET, 2003). Interfaces em RV podem fazer uso, por exemplo, de rastreadores (*trackers*), navegadores e dispositivos de entrada gestuais, para entrada de dados (Figura 5). Em Abs-Tech (2006) podem ser encontrados diversos equipamentos e dispositivos, com revenda no Brasil, e também podem ser obtidas informações sobre as características técnicas desses produtos e respectivos preços.

Rastreadores de posição 3D (3D *position trackers*) são equipamentos dotados de sensores que medem, em tempo real, mudanças na posição e orientação de objetos em um espaço 3D. Um objeto movendo-se no espaço 3D tem seis graus de liberdade, três translações e três rotações. Os rastreadores são usados para medir os movimentos da cabeça, mãos e, eventualmente, dos membros do usuário, com a proposta de

controlar a vista, locomoção e manipulação de objetos. Uma aplicação para os rastreadores é a de mapear os movimentos do usuário para um *avatar*, ou humano virtual. Outra aplicação é para atualizar (ou renderizar) a cena conforme a vista do usuário, ou seja, conforme a direção para onde ele está olhando quando usa um *display* acoplado na cabeça (HMD). Rastreadores também são usados para possibilitar uma outra modalidade sensorial, o som 3D. Diferentes modalidades de rastreadores são os rastreadores mecânicos, rastreadores magnéticos, rastreadores ultrasônicos, rastreadores ópticos, e híbridos. Uma descrição mais detalhada sobre esses dispositivos pode ser obtida em Burdea e Coiffet (2003).

FIGURA 5 - DISPOSITIVOS DE ENTRADA DE DADOS

FONTE: Abs-Tech (2006)

Navegadores, ou interfaces de navegação e manipulação são dispositivos que possibilitam uma mudança interativa na vista (ângulo de visão) que o usuário tem da cena, e na exploração, por seleção e manipulação, de um objeto virtual de interesse. A navegação/manipulação pode ser dada em função de coordenadas absolutas ou coordenadas relativas, diferenciando-se dos rastreadores, que descrevem a posição somente em função de coordenadas absolutas. Interfaces de navegação/manipulação podem ser baseadas em dispositivos rastreadores, como por exemplo, o *mouse* 3D que provê seis graus de libertade. Alguns modelos de *mouse* 3D possuem uma esfera ou um cilindro dotados de sensores que medem força e torque nas três direções. Além do

controle de movimentos, esses distositivos podem conter botões programáveis para diversas funcionalidades como, por exemplo, ligar e desligar uma simulação (BURDEA e COIFFET, 2003).

Interfaces gestuais são dispositivos que medem, em tempo real, a posição dos dedos da mão do usuário que, por meio de gestos, realiza a interação com o ambiente virtual. Alguns desses dispositivos são luvas chamadas de *data gloves*, ou luva de dados. Exemplos de dispositivos como esses são *Pinch Glove*, *5DT Data Glove*, *Didjiglove* e *CyberGlove*. Maiores informações podem ser obtidas em Burdea e Coiffet (2003).

Outros dispositivos de entrada de dados podem ser encontrados na literatura. No entanto, como o objetivo aqui não é listar todos dispositivos que existem, mas sim fornecer ao leitor uma base mínima de conhecimento sobre a tecnologia de RV, somente alguns dispositivos são mencionados.

4.2.2 Dispositivos de saída de dados

Como já foi citado, em RV busca-se estimular os múltiplos sentidos do usuário (visão, audição, tato, olfato e paladar) com a finalidade de prover a sensação de imersão. Dessa forma, pode-se classificar alguns dispositivos de saída de dados em dispositivos gráficos, sonoros e táteis, posto que os dois últimos sentidos ainda contam com um número pequeno de dispositivos no mercado.

Dispositivos gráficos de exibição ou *graphics displays* são interfaces que apresentam imagens de um “mundo” virtual a um ou mais usuários (BURDEA e COIFFET, 2003). Esses dispositivos podem ser classificados de acordo com o tipo de imagem produzida (estéreo ou mono), resolução (número de pixels), campo de visão, tecnologia utilizada (LCD - *Liquid Crystal Display* ou CRT - *Cathode Ray Tube*), tamanho e custo. O objetivo principal do estímulo visual em RV é fornecer ao usuário a sensação de tridimensionalidade. Para atingir esse objetivo, o sistema deve gerar, ao mesmo tempo, duas imagens diferentes, correspondendo às visões de cada um dos olhos. Esse tipo de geração de imagem é chamado de visão estereoscópica ou

estereoscopia.

Existem três tipos básicos de dispositivos gráficos de exibição (Figura 6): dispositivos pessoais (suportam um único usuário), baseados em monitor e baseados em projeção. Os chamados *Head Mounted Displays* (HMD's), fazem parte do primeiro tipo e podem ser no formato de óculos ou capacetes apoiados na cabeça do usuário. Alternativas para os HMD's, são os *Hand-Suported Displays* (HSD's) apoiados pelas mãos do usuário e os *Floor-Suported Displays* (FSD's) apoiados em uma estrutura semelhante a um braço mecânico provido de uma base e contra-peso. Os dispositivos baseados em monitor podem ser do tipo auto-estéreo ou então, utilizarem óculos especiais para prover a visão 3D.

FIGURA 6 - DISPOSITIVOS DE SAÍDA GRÁFICA e HMD’s

FONTE: Abs-Tech (2006)

Dispositivos baseados em projeção requerem a utilização de óculos para proverem a visão 3D. Um exemplo de aplicação para um sistema de projeção desse tipo é encontrado no laboratório Tanque de Provas Numérico, da USP - Universidade de São Paulo, em que visão estereoscópica é utilizada para visualizar resultados de simulações numéricas sobre o comportamento de estruturas em plataformas flutuantes de extração e armazenamento de petróleo.

Um sistema de projeção chamado CAVE (caverna digital) consiste de uma sala em forma de cubo em que as paredes, o chão e o teto são telas onde imagens são

projetadas. Um exemplo de sistema CAVE (*Cave Automatic Virtual Environment*) é encontrado no Laboratório de Sistemas Integráveis (LSI) da USP, constituída de 5 telas de 3x3m e um aglomerado (*cluster*) de 24 computadores tipo PC.

A visão é o principal sentido humano. Assim, a visão estereoscópica tem uma grande importância em dispositivos e sistemas RV. A estereoscopia pode ser provida utilizando diferentes princípios, por exemplo, por meio de filtros de cores (*anaglifos*) ou intercalando-se as imagens vistas por um olho e outro, simultaneamente (*visão estéreo ativa*). Maiores explicações podem ser obtidas em Raposo et al. (2004), Siscoutto et al. (2004), e outros.

Dispositivos sonoros, ou *sound displays*, 3D são sistemas de auto-falantes que emitem sons por múltiplos canais, podendo ser, por exemplo, sincronizados com a posição do usuário de forma a proporcionar a sensação de movimentação e tridimensionalidade. Em VC sons podem ser associados a variáveis.

Dispositivos táteis provêm resposta de toque (*touch feedback*) ou de força (*force feedback*). Respostas de toque podem fornecer, em tempo real, informações sobre a geometria, rugosidade e temperatura da superfície em contato. Respostas de força podem fornecer informações sobre a conformidade de uma superfície, peso e inércia de um objeto.

4.3 PADRÕES PARA DESCRIÇÃO DE AMBIENTES VIRTUAIS: VRML e X3D

VRML (*Virtual Reality Modeling Language*) é uma linguagem para modelagem de ambientes ou “mundos” virtuais 3D para distribuição na *web*. O padrão ISO que vem a substituir o VRML, chamado X3D (*Extensible 3D*), utiliza codificação XML (*Extensible Markup Language*) e incorpora os últimos avanços ocorridos em matéria de *hardwares* gráficos, compressão e segurança de dados, fornecendo melhor desempenho e impacto visual e, também, melhorando aspectos de transferência de dados em comparação ao VRML (WEB3D, 2006).

Segundo Lee e Burdea (2003), VRML não é nem uma linguagem típica de RV nem uma linguagem de modelagem, mas sim um formato de arquivo que integra

gráficos 3D com multimídia.

Pelo fato do X3D ser um padrão recente, alguns softwares de modelagem tridimensional não suportam esse formato, mas sim o VRML'97, ou VRML 2.0 versão mais atual (VRML20, 2006). Além disso, o VRML possui maior popularidade por ser utilizado há mais tempo e, conseqüentemente, existe maior quantidade de programadores experientes, documentos, exemplos e trechos de códigos disponíveis na *web*. Por estas razões, neste trabalho, o termo VRML se refere ao padrão para transferência de conteúdo 3D, podendo ser entendido como VRML ou como X3D, uma vez que a conversão de VRML para X3D é trivial.

O VRML tem diversas características que são úteis ao gerar ambientes virtuais interativos para Internet. Uma delas é descrever o mundo 3D baseado em uma estrutura de grafo de cena, o que permite criar objetos complexos baseados nos mais simples. Além disso, VRML suporta o uso de *scripts* que possibilitam animações complexas e o desenvolvimento de extensões. Também suporta cenas distribuídas, sendo possível criar um ambiente virtual a partir de arquivos diferentes, localizados em diferentes provedores, permitindo a interação multiusuários por meio de um único ambiente virtual distribuído (LEE e BURDEA, 2003).

A linguagem VRML, portanto, possui algumas vantagens que fazem dela uma opção bastante interessante no que diz respeito à visualização de conteúdo 3D de diferentes naturezas, sua versatilidade permite que seja usada com finalidades que vão desde aplicações de engenharia até entretenimento. Entre as suas maiores vantagens está o fato de ser um padrão aberto, ser possível criar cenários e "mundos" virtuais 3D usando editores de texto simples e visualizar o resultado usando um dos muitos navegadores gratuitos disponíveis hoje em dia na *web*.

4.3.1 Breve Histórico

VRML tem uma história fundamentada na colaboração de diversos pesquisadores e importantes empresas relacionadas à CG. Seu início foi em 1989 com um projeto de Rikk Carey e Paul Strass, da *Silicon Graphics Inc* (SGI). Desse projeto

nasceu o *Scenario* e, a partir daí, surge em 1992 o *Iris Inventor 3D*, uma ferramenta implementada em C++. A revisão do *Inventor*, em 1994, origina o *Open Inventor*, portável, baseado no *OpenGL*, da SG, cujo manual de referência originaria a especificação do VRML 1.0 (CARDOSO et al., 2003).

Em 1994, *Mark Pensce* e *Tony Parisi* desenvolveram o *Labyrinth*, um protótipo de navegador 3D para a *web*. Um ano depois, do conteúdo da lista de discussão do VRML, cujo ponto de partida foi o manual do *Inventor*, e respondendo à chamada para propostas de uma especificação formal para 3D na *web*, surge a primeira especificação do VRML 1.0. Após muita discussão e algumas correções, nasce a proposta de uma segunda especificação, a do VRML 2.0 (CARDOSO et al., 2003).

Aprimorada e capaz de definir novos comportamentos (com mais interação e animação) para elementos 3D, a nova proposição foi submetida à lista em 1996. As propostas apresentadas pela SG em colaboração com a Sony e com a Mitra originaram o documento de definição de VRML 2.0 que foi publicado no *SIGGRAPH'96*. Em Abril de 1997 a versão final desse texto ficou conhecida como VRML'97. A partir daí se iniciou a discussão e desenvolvimento do X3D pelo consórcio Web3D (2006).

4.3.2 Formas de interação

A linguagem VRML provê o usuário de algumas formas de interação, que promovem o aumento da sensação de imersão, e cujas possibilidades e limitações devem ser conhecidas a fim de se fazer o melhor uso possível dos recursos disponíveis. A forma de interação mais característica é a navegação no ambiente 3D. Outra forma é usando nós sensíveis a ações do usuário, podendo ser sensíveis ao movimento, ao clique ou ao arrasto do mouse.

Em relação à navegação, pode ser do tipo *Walk*, em que o usuário "caminha" pelo cenário, *Examine*, para "examinar" um objeto rotacionando-o em torno de seu centro geométrico, por exemplo, e *Fly* em que o usuário "voa" pelo cenário (VRML, 2006). O nó *Fly* provê uma navegação parecida com aquela provida pelo nó *Walk*. A

diferença é que não fornece a simulação de gravidade (movimentar-se sobre o plano do “chão”).

Outras formas de interação são obtidas usando nós chamados de sensores. São eles: *CylinderSensor*, *PlaneSensor*, *ProximitySensor*, *SphereSensor*, *TimeSensor*, *TouchSensor* e *VisibilitySensor*. Esses sensores servem para disparar eventos com base em ações do usuário, como um clique de *mouse*, a aproximação ou um *close* em determinado objeto, durante a navegação. O nó *TimeSensor* gera eventos usando contagem de tempo. Os nós *CylinderSensor*, *PlaneSensor* e *SphereSensor* permitem mover um determinado objeto em torno do eixo y, sobre o plano x-y, ou em torno de um ponto, respectivamente.

Em particular, o nó *PlaneSensor* pode ser utilizado para simular a movimentação de equipamentos pesados em uma subestação de energia e, assim, analisar visualmente possíveis colisões ou obstruções em sua passagem, considerando a espaço 3D e os objetos nele contidos. Além disso, combinando-se modelos CAD com resultados de simulações numéricas, tais como análise de campos eletromagnéticos, por exemplo, torna-se possível realizar simulações considerando campos atuantes no local, identificando distâncias críticas e outras limitações.

4.3.3 Formas de representação de dados e informações

A apresentação de dados e informações em VRML pode ser feita de três formas básicas distintas: por meio de geometrias e atributos, por meio de textos, podendo-se combinar quadros HTML (MIRANDA, 2005), e por meio de som.

O VRML possui algumas primitivas geométricas (cone, esfera, cilindro, etc) que podem ser usadas como ícones, ou glifos (*glyphs*), na representação de campos escalares e/ou vetoriais, por exemplo. A aparência dessas primitivas também pode ser ajustada pela utilização de transparências, cores, texturas e animações. Também podem ser usadas outras primitivas gráficas como linhas e pontos.

A ferramenta de visualização VTK pode ser utilizada para gerar geometrias ou outras formas de representação, a partir de um conjunto de dados, e exportar para o formato VRML.

Dispositivos de saída de dados, característicos de RV, possibilitam a visão e audição 3D além de estímulos a outros sentidos, tais como o tato. Assim, informações ou dados numéricos também podem ser mapeados para a freqüência de um pulso sonoro, por exemplo, a fim de representar dados de uma grandeza qualquer por meio de outras formas sensoriais diferentes da visão.

4.3.4 Iluminação com VRML

Não são muitos os recursos disponíveis na linguagem VRML para simular a iluminação. Um tipo especial de luz, chamada *Head Light*, é implementada no *Browser* e proporciona uma luminosidade na direção em que o usuário está olhando, como se fosse uma lanterna acoplada à cabeça ou a um capacete. Outros três nós que descrevem fontes de luz são: *Directional Light*, *Point Light* e *Spot Light*.

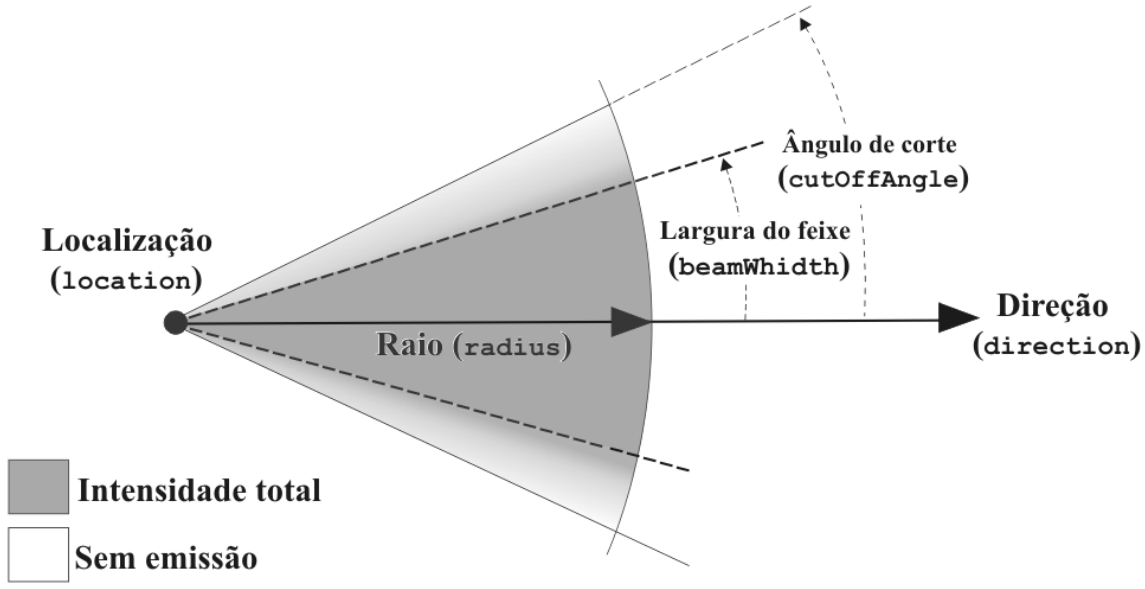
O nó *Directional Light* produz infinitos raios de luz paralelos a uma única direção, como o efeito da luz do sol na terra. O nó *Point Light* produz o efeito de um ponto de luz que emite raios luminosos de igual intensidade em todas as direções, ou seja, possui simetria radial. E o nó *Spot Light* cria o efeito de um cone de luz, cujos parâmetros de ajuste, como o ângulo de abertura, são definidos no código, podendo ser ajustados para se obter o efeito desejado.

Efeitos de reflexão de luz, utilizados na geração de imagens fotorrealísticas, não são implementados em VRML, e somente a iluminação direta está disponível. Esses efeitos são geralmente “caros” computacionalmente e sua implementação, hoje, não é usual em sistemas interativos e que requerem *rendering* em tempo real. Assim, somente a iluminação direta está disponível em VRML, o que significa que, se um objeto não for atingido diretamente pelos feixes de luz de uma fonte (sem considerar a obstrução da luz por outros objetos), ele aparecerá escuro.

Para melhorar este aspecto, pode-se usar o campo *ambientIntensity* de um nó de luz. Esse campo determina o quanto este ponto de luz contribui com a luz ambiente da cena. Sombras também não são implementadas em VRML. Dessa forma, se houver um objeto opaco entre uma fonte de luz e um segundo objeto, o segundo objeto poderá aparecer iluminado. Outra característica do VRML é que a intensidade da luz sofre um decaimento em função da distância, fazendo com que objetos mais distantes apareçam menos iluminados. Ajustes nas definições do material que compõe os objetos da cena também podem aumentar o realismo da cena.

E, por fim, a luz do tipo Spot que descreve um cone de luz definido por dois campos cujos parâmetros podem ser ajustados: *cutOffAngle* e *beamWidth* (Figura 7). O *cutOffAngle* determina o ângulo do cone externo em radianos. O *beamWidth* define o ângulo de um cone interno dentro do qual a intensidade clara é constante. Os raios claros que se encontram entre o cone interno e o cone externo tem uma intensidade que vai diminui gradativamente do interior para o exterior.

FIGURA 7 - *SPOT LIGHT* EM VRML



FONTE: Especificação do VRML 2.0

O nó *Spot Light* é o mais adequado para representar uma luminária do tipo refletor. Seus parâmetros podem ser ajustados para que o efeito luminoso fique parecido com o que se espera na realidade. Um exemplo da sintaxe do nó *Spot Light* e dos campos que podem ser ajustados, está ilustrado na Figura 8.

Os parâmetros do nó *spot light* podem ser ajustados a fim de reproduzir, de forma simplificada, um sistema de iluminação projetado. O campo *intensity* define a intensidade luminosa e pode, por exemplo, ser ajustado conforme o fluxo luminoso total de uma fonte de luz, relativo a um valor normalizado considerando-se todas as fontes luminosas usadas no projeto. O campo *ambientIntensity* define a contribuição para a iluminação total do ambiente. O campo atenuação define o decaimento da intensidade luminosa em função da distância. O campo *radius* define a distância máxima que a luz atinge, *beamWidth* e *cutOffAngle* são os ângulos mostrados na figura 7 e podem ser ajustados conforme o ângulo de abertura do facho luminoso de um refletor. A sintaxe do nó *Spot Ligth* aparece na figura 8 e os ângulos são definidos em radianos.

FIGURA 8 - SINTAXE DO NÓ *SPOT LIGTH* EM VRML

```
SpotLight {
on TRUE
intensity 1
ambientIntensity 0
color 1 1 1
location 0 0 0
direction 0 0 0
attenuation 1 0 0
radius 100
cutOffAngle 0.78
beamWidth 1.57 }
```

FONTE: O autor

Assim, a simulação de iluminação em VRML pode ser realizada ajustando-se os parâmetros disponíveis nos nós de luz a fim de tornar o efeito final da visualização o mais próximo possível da realidade. Porém, a utilização dos modelos de fontes de luz disponíveis em VRML em simulações não é adequada para uso em projetos de iluminação devido às muitas simplificações existentes. Uma solução para viabilizar o

uso de RV, por meio da linguagem VRML, para projetos de iluminação, é gerar, previamente, uma simulação numérica para os valores de iluminância no domínio desejado e, então, utilizar um ambiente virtual 3D escrito em VRML para a visualização dos resultados dessa análise numérica. Esse processo consiste na visualização de campos de escalares (iluminância em lux) em que técnicas de VC são necessárias.

4.4.5 Algumas características do VRML interessantes para uso em VC

Arquivos que simulam ambientes 3D em VRML são basicamente uma descrição textual que usa o padrão ASCII. Dessa maneira, o usuário, utilizando qualquer editor de textos ASCII pode conceber um ambiente ou um objeto, salvá-lo com a extensão *wrl*, e visualizá-lo em um navegador adequado. O navegador, ou *browser*, deve suportar o formato VRML, ou então, pode-se utilizar um *plug-in* associado ao navegador de Internet. Assim, uma vez gerado o arquivo de dados no formato VRML, torna-se possível visualizá-lo sem necessidade de *softwares* comerciais.

A linguagem VRML apresenta algumas outras características vantajosas, como a de ser de fácil aprendizado e existir abundância de material disponível para consulta, além de dispor de muitos *sofwtares* gratuitos para sua edição e visualização. Essas características, aliadas à essência 3D do VRML, fazem dessa linguagem uma opção com potencial para utilização na visualização de dados físicos distribuídos no espaço, ou seja, pode ser usada como interface de VC para visualização de campos em domínios 3D.

Outra característica interessante do VRML é a possibilidade de sua integração com páginas HTML, o que permite, por exemplo, o acesso a informações textuais na mesma tela que mostra o ambiente virtual. Páginas ou *frames* HTML podem ser ativadas ou modificadas em resposta a “cliques” de *mouse* sobre objetos da cena (interação com o ambiente). Isso faz com que muitas aplicações de visualização de informações e dados possam ser criadas. Uma aplicação integrando VRML com

HTML e banco de dados foi desenvolvida durante a pesquisa e está descrita detalhadamente em Miranda (2005).

O aplicativo desenvolvido foi utilizado para a visualização de informações sobre os equipamentos de iluminação em uma subestação de energia elétrica, mas pode ser adaptado para outras aplicações. De maneira geral, qualquer objeto (equipamento, luminária, etc) em uma cena 3D pode receber um nó *Anchor* que acessa uma página HTML, a qual contém informações armazenadas em um banco de dados SQL *Server*, e a página pode ser atualizada dinamicamente via rotinas da linguagem de programação ASP.Net.

5 O VTK (*THE VISUALIZATION TOOLKIT*)

Entre as ferramentas de Visualização Científica que existem, uma se destaca por ser amplamente utilizada por cientistas do mundo todo, inclusive no desenvolvimento de aplicativos comerciais, também por ser *open source* e, principalmente, por seus autores publicarem livros que permitem um aprendizado completo do seu uso e funcionamento (KITWARE, 2004). Essa ferramenta é o *Visualization ToolKit*, ou VTK. Este capítulo dedica-se a essa ferramenta.

5.1 O QUE É VTK

O *Visualization ToolKit* é um sistema de software de código aberto para computação gráfica 3D, processamento de imagens e visualização, usado por milhares de pesquisadores e investigadores em todo o mundo inclusive no desenvolvimento de aplicações comerciais (SCHROEDER et al., 2000). Consiste numa biblioteca de classes em C++ e várias camadas de interfaces interpretadas, incluindo Tcl/Tk, Java, e Python, suportando uma ampla variedade de algoritmos de visualização para diferentes tipos de dados. Também são implementadas técnicas de modelagem avançadas como modelagem implícita, redução de polígonos, suavização de malhas e triangulação de Delaunay.

O projeto e a implementação da biblioteca foram fortemente influenciados pelos princípios de orientação a objetos, e o software já foi instalado e testado em quase todas as plataformas baseadas em Unix e Windows. O VTK pode ser obtido por meio de *download* no site *www.vtk.org* ou então por meio do CD que acompanha o livro de Schroeder et al. (2004).

VTK provê uma variedade de representações de dados incluindo conjuntos de pontos desorganizados, dados poligonais, imagens, volumes, e também malhas estruturadas, retilíneas e não-estruturadas. VTK possui leitores/importadores e escritores/exportadores para trabalhar com dados de diferentes aplicações. O modelo

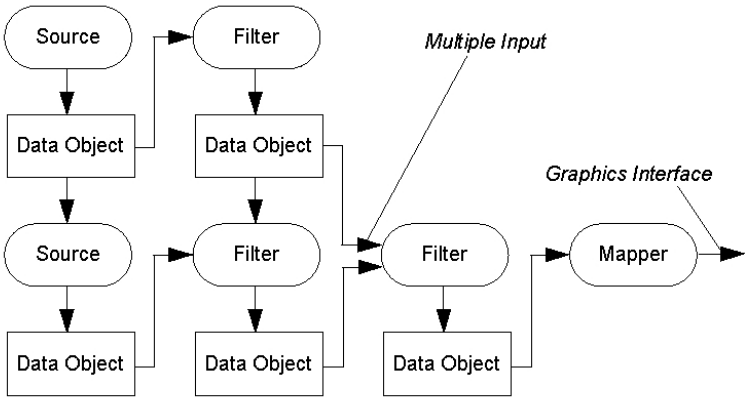
de renderização do VTK suporta as abordagens 2D, poligonal, volumétrica e baseada em texturas que podem ser usadas de forma combinada (KITWARE, 2005).

5.2 ARQUITETURA: MODELO GRÁFICO E *PIPELINE* DE VISUALIZAÇÃO

Como explica Schroeder et al. (2004), o VTK é composto por duas partes principais, um núcleo compilado, escrito em C++, e uma camada interpretada gerada automaticamente que suporta atualmente as linguagens Tcl, Java e Python. No núcleo em C++ são implementadas as estruturas de dados, algoritmos e funções de sistema. A camada interpretada permite que, rapidamente, sejam criadas aplicações utilizando ferramentas de GUI (*Graphical User Interface*) tais como Tcl/Tk, Python/Tk, ou Java AWT. Assim, enquanto o núcleo compilado provê velocidade e eficiência, a camada interpretada oferece flexibilidade e extensibilidade.

O VTK possui dois subsistemas principais: o modelo gráfico e o *pipeline* de visualização. O modelo gráfico (ilustrado na Figura 9) forma uma camada abstrata sobre a linguagem gráfica OpenGL, assegurando a portabilidade entre plataformas.

FIGURA 9 - MODELO GRÁFICO



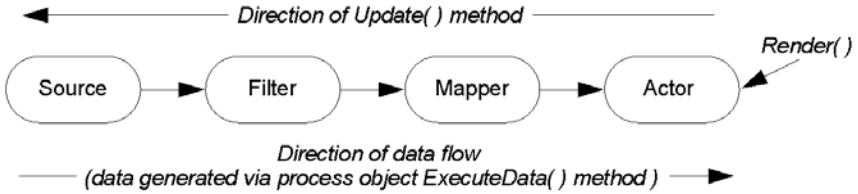
FONTE: Schroeder et al. (2004).

Os nomes das classes no modelo gráfico foram adaptados da indústria cinematográfica, por exemplo, luzes, câmeras e atores. Essas e outras classes são

instanciadas pelo usuário para criar uma cena. Segundo Schroeder et al. (2004) existem sete classes básicas para a renderização de uma cena. São elas: *vtkRenderWindow*, *vtkRenderer*, *vtkLight*, *vtkCamera*, *vtkActor*, *vtkProperty* e *vtkMapper*.

O *pipeline* de processamento transforma os dados em formas que podem ser mostradas no subsistema gráfico (Figura 10). Por exemplo, pode-se ler uma nuvem de pontos desorganizados, criar uma malha via trangulação de Delaunay, e mostrar a malha usando renderização de superfícies.

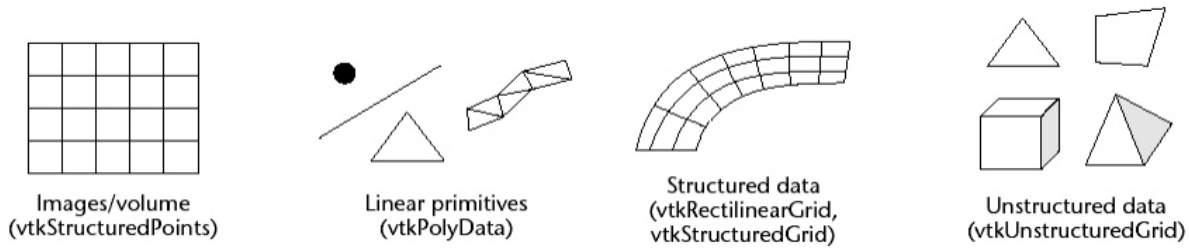
FIGURA 10 - *PIPELINE* DE VISUALIZAÇÃO



FONTE: Adaptado de Schroeder et al. (2004).

O *pipeline* é construído pela conexão de objetos de dados e filtros (*SetInput*/*GetOutput*): os objetos de dados fornecem acesso aos dados e os filtros realizam operações sobre esses dados. Os objetos de dados suportados pelo VTK são mostrados na figura 11.

FIGURA 11 – OBJETOS DE DADOS DO VTK



FONTE: Schroeder (2000).

Conjunto de dados, ou *Data Sets*, são uma especialização de objetos de dados

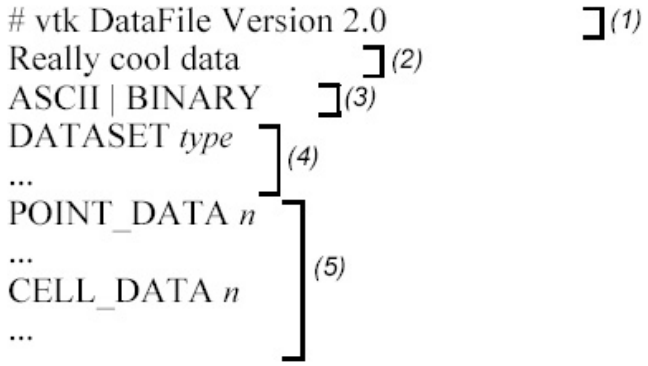
possuindo estruturas geométrica e topológica, e também, atributos associados em conformidade com sua geometria e/ou topologia. Os objetos de processos, comumente chamados de filtros, no VTK podem ser classificados em três categorias diferentes: *sources*, *filters* e *mappers*. As fontes (*sources*) não admitem entradas (*Inputs*), apenas saídas (*Outputs*), os filtros (*filters*) admitem uma ou mais entradas e uma ou mais saídas, e os mapeadores (*mappers*) finalizam o *pipeline* acoplando-se ao modelo gráfico.

5.3 O FORMATO *LEGACY* DO VTK

O VTK possui um conjunto de rotinas capazes de ler e escrever os formatos mais populares de arquivos de dados, oferecendo assim uma forma simples para viabilizar a transição de dados entre diferentes softwares. O VTK também possui seus próprios formatos de arquivos: o *Legacy* e o baseado em XML. O *Legacy* é mais simples, podendo ser facilmente escrito tanto manualmente quanto via programação. O formato baseado em XML é mais flexível, suporta o acesso aleatório, entrada e saída de sistemas paralelos e tem maior capacidade de compressão.

O formato de arquivo *Legacy* do VTK consiste em cinco partes básicas, como indicadas na Figura 12.

FIGURA 12 – AS CINCO PARTES DO ARQUIVO *LEGACY* DO VTK



FONTE: Kitware (2004).

As cinco partes básicas são descritas a seguir:

Parte 1 - Identificador: identifica o arquivo e informa a versão do VTK.

Parte 2 - Cabeçalho: consiste em uma cadeia de caracteres (*string*) (de no máximo 256 caracteres, finalizada com /n) que pode ser usado para descrever qualquer informação pertinente.

Parte 3 - Formato: descreve o tipo de dados – ASCII ou BINÁRIO.

Parte 4 - Estrutura dos dados: descreve a Topologia\Geometria dos dados, ou seja, se são pontos estruturados, *grid* não estruturada, etc.

Parte 5 - Atributos: relacionados a células ou a pontos, que podem ser, por exemplo, escalares, vetores, tensores, normais, coordenadas de texturas, etc.

Um exemplo de arquivo do tipo Legacy do VTK é mostrado abaixo:

```
# vtk DataFile Version 3.0
vtk output
ASCII
DATASET STRUCTURED_POINTS
DIMENSIONS 41 21 9
SPACING 0.5 0.5 0.5
ORIGIN 0 0 0
CELL_DATA 6400
POINT_DATA 7749
SCALARS scalars float
LOOKUP_TABLE default
9.66094 11.3903 14.7769 16.1594 18.0373 21.7153 25.4946 27.0943 29.556
31.7627 33.0606 38.8882 40.1404 40.6551 41.7893 43.2789 41.7745 40.7437
42.8493 45.2166 47.5835 50.6087 53.1031 58.7691 66.6661 65.5652 68.202
78.7795 72.6687 73.6285 73.4933 78.8409 77.6152 80.1229 75.4688 72.3009
69.9846 59.3487 55.6119 54.0945 49.5653 10.3909 12.7939 15.5356 18.0909
23.3049 25.6245 30.3735 32.359 35.9408 40.7255 42.1185 48.4489 44.9994
48.1471 49.128 47.7495 48.6963 48.0272 47.7874 50.242 51.6888 63.8216
...
```

O VTK suporta cinco diferentes estruturas de dados: pontos estruturados, grade (*grid*) estruturada, grade retilínea, grade não-estruturada e dados poligonais. Dados com topologia implícita (estruturados) variam recebendo acréscimo em x, depois em y, e então, em z.

Resultados de simulações, se armazenados em arquivos com o Formato *Legacy*, poderão ser facilmente interpretados por algoritmos que utilizam esta ferramenta. Maiores detalhes podem ser obtidos em Kitware (2004).

6 APLICAÇÕES EM SUBESTAÇÕES

Sistemas de RV aplicados a atividades em subestações de energia elétrica são pouco comuns, embora seja possível imaginar muitos aspectos nos quais esta tecnologia poderia trazer benefícios. Alguns trabalhos foram publicados em Arroyo e Los Arcos (1999), Helwig (2002) e Geus e Dometerco (2004), e em Orteng (2006) encontram-se vídeos com o modelo digital 3D de uma subestação. As aplicações que podem ser implementadas vão desde a simples navegação exploratória no modelo digital 3D da subestação, até a integração de diversas funcionalidades usando o ambiente virtual como uma interface 3D. Exemplos de funcionalidades são o processamento e visualização interativa de grandezas físicas existentes no ambiente, acesso a informações em base de dados, simulações de transporte de cargas e planejamento de ampliação.

Assim, algumas funcionalidades aplicadas a atividades em subestações de energia elétrica, que poderiam ser providas por um sistema de RV são:

- Planejamento e simulação de iluminação;
- Visualização de campos elétricos e magnéticos;
- Planejamento para mudança de leiaute ou ampliação e infra-estrutura;
- Simulação de movimentação de equipamentos;
- Acesso a informações diversas em bases de dados.

Descobrir novas aplicações para a tecnologia de RV, como afirmam Burdea e Coiffet (2003), depende essencialmente da imaginação de seus programadores.

6.1 NAVEGAÇÃO NO MODELO 3D

A visualização de grandes modelos geométricos digitais, provenientes de projeto auxiliado por computador (CAD), em sistemas de RV é interesse de muitas empresas e tema de trabalhos e pesquisas em todo o mundo, como pode ser observado nos trabalhos de Berta (1999), Zong et al. (2002), Corseuil et al. (2004), e muitos

outros. O impacto visual e o caráter imersivo, proporcionado por sistemas de RV, atraem a atenção de seus usuários e melhoram a percepção do conteúdo 3D em estudo.

Muitos sistemas CAD permitem a exportação para o formato VRML, como por exemplo, o Microstation (Bentley) e o Solidworks (Solidworks). O *software* AutoCAD (Autodesk) não exporta diretamente para VRML, mas utilizando outro programa, como o 3D Studio Max (Discreet), por exemplo, em um processo intermediário, pode-se converter um modelo do AutoCAD (arquivo DWG) para o formato VRML.

Em VRML é possível "navegar" no modelo 3D interativamente, de forma exploratória utilizando um programa interpretador que pode ser um *plug-in* para o navegador de Internet ou um *browser* compatível com VRML. Muitos *plug-ins* e *browsers* estão disponíveis gratuitamente na *web*, e alguns exemplos são o Octaga[6], OpenWorld[7] e Xj3D[8].

Entre as vantagens de se usar VRML para visualizar modelos digitais 3D estão o impacto visual proporcionado, podendo ser usadas texturas e outros elementos com *background* (pano de fundo) e *fog* (efeito de poeira ou fumaça suspensa no ar), a facilidade de transferência e visualização dos arquivos 3D entre diferentes sistemas, a possibilidade de melhorar a comunicação e, conseqüentemente, a colaboração entre profissionais, e também permitir a visualização com dispositivos especiais para visão 3D.

Algumas dificuldades no uso operacional de RV para a visualização de grandes modelos CAD são descritas no trabalho de Corseuil et al. (2004). Entre elas, tem-se o baixo realismo apresentado pela maioria dos sistemas CAD, que não têm suporte ao uso de textura. Além disso, aplicar texturas individualmente é muito oneroso pelo tamanho dos modelos. Outra dificuldade é o baixo desempenho obtido, segundo Corseuil et al. (2003), pelos algoritmos de otimização utilizados em seu trabalho, que ainda não são suficientes para modelos com as dimensões apresentadas.

---

[6] < http://www.octaga.com>

[7] < http://www.openworlds.com>

[8] < http://www.xj3d.org>

Além disso, superfícies complexas, como superfícies NURBS (*NonUniform Rational BSplines*), não são devidamente tratadas. As malhas geradas para este tipo de superfícies são ineficientes para utilização em sistemas que requerem *rendering* em tempo real.

6.2 PLANEJAMENTO DA ILUMINAÇÃO

O adequado dimensionamento da iluminação em subestações de energia é importante por melhorar as condições de trabalho, reduzir os riscos de acidentes e o consumo de energia. Um método preciso e muito utilizado em cálculo luminotécnico é o método ponto-a-ponto, ou método das iluminâncias. Com ele é possível calcular a iluminância resultante em cada ponto de interesse, ou em uma malha de pontos, e traçar as curvas 'isolux' em um plano de trabalho (MOREIRA, 1999). Esse método geralmente requer o uso de aplicativos computacionais para gerar os valores de iluminância em malhas 2D ou 3D definidas em um espaço 3D. Técnicas de VC podem ser usadas para representar graficamente esses dados de forma a promover um ganho em sua compreensão.

Nos últimos anos grandes empresas do setor de materiais de iluminação vêm investindo na elaboração de programas computacionais de qualidade, voltados para projetos de iluminação. Muitas dessas ferramentas, como o DIALux[9] e o Calculux[10], são de distribuição gratuita e se mostram à altura de renomados programas comerciais. Esses programas fazem uso de bases de dados contendo curvas fotométricas digitalizadas em padrões internacionais de produtos (lâmpadas e luminárias) de diversos fabricantes.

No entanto, segundo Marinoski et al. (2003), no cenário nacional, os profissionais do ramo ainda têm pouco conhecimento e prática na aplicação dessas ferramentas. Este fato pode ser atribuído à dificuldade de aprendizado e de utilização

[9] <http://www.dialux.com>

[10] <http://www.lightingsoftware.philips.com>

dos aplicativos ou à impossibilidade de sua integração com sistemas do tipo CAD, por exemplo.

No caso específico da empresa brasileira de energia elétrica relacionada ao estudo aqui relatado, os projetos de iluminação das subestações eram realizados sem a utilização de um *software* específico. O processo demandava grande esforço e tempo, e não fornecia uma visualização 3D dos resultados. Em linhas gerais, os cálculos eram realizados com auxílio de planilhas eletrônicas e a representação dos resultados era feita por meio de diagramas bidimensionais e relatórios.

O método ponto-a-ponto, utilizado por essa empresa, requer o conhecimento de informações fotométricas sobre as fontes de luz utilizadas no projeto. Essas informações são geralmente fornecidas pelo fabricante por meio de gráficos nos catálogos dos produtos. Deste modo, o projetista precisava consultar os catálogos inúmeras vezes, de forma exaustiva.

Um sistema de RV aplicado ao planejamento de iluminação pode prover uma interface intuitiva, de fácil utilização, e uma visualização 3D interativa e imersiva dos campos de iluminância. Além disso, pode-se combinar a visualização de dados físicos resultantes de outras análises numéricas, com dados de modelos geométricos digitais (modelo CAD) (BURIOL, 2006). Por exemplo, pode-se visualizar a geometria de um transformador de potência e o campo eletromagnético a sua volta simultaneamente, aumentando assim a capacidade de abstração de informações, por parte do usuário (projetista), sobre o fenômeno em estudo.

Formatos digitais padronizados, como o padrão IES[11] (*Illuminating Engineering Society*), para transferência de informações fotométricas de fontes de luz possibilitam que a obtenção da intensidade luminosa, emitida em cada direção por uma determinada fonte, seja realizada automaticamente e computada diretamente no algoritmo de cálculo.

Ferramentas para VC, como o VTK, podem ser utilizadas gratuitamente no desenvolvimento de aplicativos gráficos para a visualização de campos de iluminâcia.

[11] <http://www.iesna.com>

Técnicas de VC, como mapeamento de cores, extração de isosupertícies, dentre outras, podem ser utilizadas na visualização 3D interativa de resultados numéricos. O VTK também realiza a exportação para o formato VRML.

Assim, é possível desenvolver algoritmos para processar e visualizar campos de iluminância 3D para qualquer projeto de iluminação utilizando um ambiente virtual escrito em VRML como interface.

6.3 VISUALIZAÇÃO DE CAMPOS ELÉTRICOS E MAGNÉTICOS

Existe uma preocupação, tanto das empresas de energia e telecomunicações quanto dos órgãos de saúde, com questões relativas à influência dos campos elétricos e magnéticos nos seres vivos. Onde existe corrente elétrica há campos eletromagnéticos. Os trabalhadores do setor elétrico, operadores de radar, rádio, ondas curtas, microondas e telefonia celular estão expostos a seus efeitos (SINDIPETRO, 2005).

Os campos eletromagnéticos de alta freqüência, produzidos por correntes elétricas de linhas de transmissão e de distribuição, são potencialmente perigosos. Trabalhadores em unidades desse tipo sofrem longos períodos de exposição, situação que pode provocar danos à sua saúde e prejuízo à empresa. Considerando que, segundo MTE (2005), em 2001 o maior volume de trabalhadores concentrou-se na distribuição de energia elétrica, cujo número de empregados das concessionárias e suas prestadoras de serviços totalizavam 350.000 trabalhadores, eventuais acidentes podem assumir grandes proporções.

Além disso, como também pode ser visto em MTE (2005), riscos atribuídos à ação de campos eletromagnéticos são relacionados também à possibilidade de ocorrências de curtos-circuitos ou mau funcionamento do sistema elétrico, originando incêndios, explosões ou acidentes ampliados. Sendo Assim, é importante o conhecimento da distribuição dos campos elétricos e magnéticos em uma subestação de energia elétrica, tanto para promover a segurança das pessoas que trabalham em subestações, quanto para evitar danos em equipamentos eletrizados.

Neste sentido, a importância de se visualizar os campos eletromagnéticos em um ambiente virtual, no qual pode-se ver os equipamentos e construções que coexistem no mesmo espaço físico em que atua o campo, é a de conhecer melhor a sua distribuição. Já que é invisível ao ser humano, apesar de nocivo e poderoso, é de grande valor poder visualizá-lo, ou melhor, visualizar o resultado de uma análise numérica, analítica, ou medições de campos eletromagnéticos.

A análise computacional da distribuição de campos eletromagnéticos por meio de métodos numéricos é bastante difundida na engenharia. Neste tipo de análise a visualização dos dados constitui um passo muito importante. No entanto, os métodos de visualização não têm explorado totalmente a extraordinária evolução da computação gráfica. Isso se dá devido a fatores como: dificuldades específicas das linguagens de programação, concentração de esforços dispensada na obtenção dos dados e não na exploração das possibilidades de visualização, além do alto custo de um bom pacote de software comercial, como constatou Pavão et al. (2001). Exemplos de trabalhos descrevendo a utilização de RV para visualização de campos eletromagnéticos são descritos em Huang et al. (1996) e Souza et al. (2000).

7 IMPLEMENTAÇÕES REALIZADAS

Neste capítulo são descritos os resultados experimentais obtidos, em que se buscou desenvolver um aplicativo com interface de visualização em RV, para planejamento da iluminação em subestações. Vale salientar que a abordagem utilizada neste trabalho procura enfatizar o uso de Realidade Virtual (RV) como interface para Visualização Científica (VC), mais especificamente para a visualização de campos de grandezas físicas, integrada (ou combinada) à visualização de modelos geométricos de engenharia (modelos CAD – *Computer Aided Design*). Os algoritmos desenvolvidos durante este trabalho realizam o processamento e o pós-processamento de campos de iluminância. No entanto, esses algoritmos de visualização poderiam ser utilizados para visualizar valores de intensidade de campos elétricos ou magnéticos, por exemplo.

7.1 COMENTÁRIOS INICIAIS

Um dos focos deste trabalho foi o uso de técnicas de VC (e da ferramenta VTK), na visualização de campos, combinadas com a visualização de modelos CAD de engenharia, utilizando como interface um ambiente virtual 3D escrito em VRML (ou X3D). Em outras palavras, objetivou-se experimentar o uso de RV para visualização simultânea de dados científicos e dados geométricos, e possíveis aplicações em engenharia, mais especificamente em subestações de energia elétrica. Usou-se como modelo para testes uma subestação cuja planta digital 3D foi modelada a partir de um levantamento a laser, o que permite um alto grau de detalhamento das geometrias obtidas. Os dados de campos escalares utilizados foram obtidos por simulações numéricas de iluminação.

A empresa financiadora do projeto não possuía um *software* para simulação de iluminação. Por essa razão, desenvolveu-se o programa chamado *ProcessaIES* que realiza os cálculos da iluminação direta por meio do método ponto-a-ponto. Anteriormente a este desenvolvimento, alguns programas para projetos de iluminação foram analisados, mas nenhum deles possuía a versatilidade e a facilidade de uso

desejada. O aplicativo desenvolvido processa o cálculo do campo de iluminâncias, em uma malha ortogonal escolhida pelo usuário, e gera um arquivo com a estrutura adequada para ser visualizado com um aplicativo baseado no VTK (*Visualization Toolkit*).

Entre as características favoráveis e específicas às necessidades do projeto que o aplicativo *ProcessaIES* possui, destacam-se: o uso do método ponto-a-ponto para cálculo da iluminação direta, a possibilidade de montar qualquer esquema de iluminação (independente de quantidade, localização, posicionamento ou modelo dos equipamentos utilizados), a escolha das dimensões da malha de pontos (2D ou 3D) a ser utilizada, e a geração do arquivo no formato VTK como saída, o que amplia as possibilidades de visualização desses dados. O algoritmo desenvolvido também foi integrado ao *software* de CAD 3D SolidWorks 2006, o que possibilitou implementar uma análise de obstruções por corpos opacos, fornecendo uma segunda simulação considerando a complexa geometria dos equipamentos que compõem a subestação.

No item seguinte são descritos em detalhes os passos da implementação para o cálculo numérico das iluminâncias, bem como o funcionamento do aplicativo *ProcessaIES*.

7.2 VISUALIZAÇÃO DO MODELO CAD DA SUBESTAÇÃO EM VRML

Como apresenta o trabalho de Corseuil et al. (2004), encontram-se algumas dificuldades no processo de visualização de grandes modelos digitais, provenientes de sistemas CAD (*Computer Aided Design*), em sistemas de RV. Isso se dá devido ao fato de que sistemas CAD e de RV foram desenvolvidos com diferentes propósitos. Enquanto o primeiro exige precisão na geometria, o outro requer desempenho em renderização. Isso acarreta que um modelo CAD pode possuir grande número de polígonos comprometendo a renderização em tempo real quando visualizado em RV. Dificuldades como essa foram encontradas durante este trabalho e são descritas no que segue, bem como, os artifícios utilizados para superá-las.

O modelo digital, utilizado neste trabalho, de uma subestação de energia elétrica foi gerado a partir de um escaneamento a laser e posterior modelagem geométrica tridimensional. Ambos foram feitos por uma empresa contratada[12] pela concessionária de energia à qual pertence a subestação. Entre as vantagens de tecnologias como a de escaneamento a laser estão: a precisão, a rapidez e a segurança do operador, uma vez que ele não precisa se aproximar dos equipamentos eletrizados. Outra vantagem é que possibilita a integração dos dados coletados com sistemas CAD.

No caso da subestação estudada neste projeto, as plantas existentes antes do levantamento tinham sido concebidas em 2D, no ano de 1981. O escaneamento a laser foi a forma mais eficiente encontrada para obter o modelo 3D digitalizado, em um formato compatível com as ferramentas existentes atualmente.

O produto do escaneamento é uma nuvem de pontos altamente densa, o suficiente para gerar geometrias com uma precisão na ordem de milímetros. A partir da nuvem de pontos, com a utilização do *software* Cyclone[13], foi realizada a modelagem geométrica 3D e conversão para o formato DWG do AutoCAD 2004 (Autodesk). Esse formato foi escolhido pela empresa contratada por ser um padrão adotado internamente.

Devido ao tamanho excessivo, dividiu-se o modelo em quatro partes:

1 - Área de Capacitores.dwg (33.171 KB)

2 - Área de Trafos.dwg (63.823 KB)

3 - Pátio 138 KV.dwg (155.527 KB)

4 - Pátio 345 KV.dwg (120.948 KB)

As primeiras análises dos dados revelaram algumas peculiaridades que acabaram se tornando obstáculos no processo de conversão para outros formatos. Diversas entidades gráficas uni e bidimensionais, como *polylines* e *3dFaces*, não foram corretamente interpretadas no processo de exportação, necessitando de

---

[12] <http://www.santiagoecintra.com.br>

[13] < http://www.leica.loyola.com/products/hds/cloudworx.html>

correções (inversão de normais, acoplamento (*tesselation*) de polígonos, redução do número de polígonos, etc) antes de gerar o modelo VRML.

Foram testados, para geração do modelo VRML, os *softwares* 3D Studio Max 7 (Discreet), Microstation (Bentley) e SolidWorks (SolidWorks). Todos os três permitem a exportação para o formato VRML97, podendo ser usados para este fim sem maiores dificuldades a não ser pelo tamanho do modelo.

Observou-se que a entidade gráfica *3DSolid* do AutoCAD podia ser interpretada corretamente pelo *software* SolidWorks, não acarretando em erros de importação. Assim, foi feita uma segunda modelagem em que se gerou o modelo completo utilizando somente a entidade *3DSolid*. A importação dos arquivos DWG no SolidWorks pôde ser realizada somente em pequenas partes. Então, partes do modelo foram separadas em *layers*, seguindo uma lógica de integridade (por exemplo, *transformador01*, *tranformador02*, *poste*, *torre*, etc), para serem importadas no SolidWorks e exportadas para VRML em arquivos separados (por exemplo, *transf_01.wrl*, *tranf_02.wrl*, etc).

Essa estrutura possibilitou a criação de uma base de dados geométricos de equipamentos e objetos que compõem a subestação. O modelo da subestação foi obtido agrupando os arquivos VRML por meio de nós *Inline* (nó da linguagem VRML que insere na cena o conteúdo de um outro arquivo VRML) em um único arquivo que carrega os demais.

O *software* SolidWorks possui um recurso que possibilita ajustar a qualidade da representação do modelo 3D no ambiente de trabalho. O ajuste controla a quantidade de polígonos (resolução) que formam algumas primitivas gráficas, como cones e cilindros, por exemplo. Ao exportar para VRML, essas primitivas mantêm o número de polígonos referentes ao nível de qualidade definido no SolidWorks. Esse recurso permite uma otimização do modelo no processo de conversão. Na resolução mais baixa do SolidWorks, a base de um cilindro possui 5 arestas. No modelo DWG original, foram encontrados cilindros com até 72 arestas na base.

A figura 13 ilustra o efeito da redução de polígonos em objetos 3D.

FIGURA 13 – REDUÇÃO DE POLÍGONOS

FONTE: O autor.

Para melhorar o desempenho de navegação no modelo também foram utilizadas outras técnicas de otimização, como nós LOD[14] (*Level Of Details*) e, para aumentar o realismo do modelo, foram usadas texturas e "pano de fundo" (*BackGround*). Com o modelo VRML foi possível navegar interativamente na planta digital 3D da subestação, utilizando, neste trabalho, o navegador *Octaga*.

Não foi objetivo deste trabalho, avaliar questões relativas ao desempenho da aplicação em relação à navegação e renderização. Esse desempenho pode ser medido por meio da taxa de quadros por segundo (*frame rate*) apresentada. É desejável que a aplicação seja capaz de apresentar em torno de 30 quadros por segundo. A taxa varia em função do equipamento utilizado (*software* e *hardware*) e também em função do tamanho do modelo (número de polígonos). No caso apresentado nesse trabalho o tamanho do modelo da subestação varia também em função da maneira é feita a conversão do formato DWG para VRML, por exemplo, é possível ajustar as primitivas geométricas (cones, cilindros, esferas, etc.) para diferentes resoluções.

Nos diversos testes realizados estimou-se que, convertendo o modelo original com todas as primitivas de seção circular (cones, cilindros e esferas) com resolução tal

[14] LOD ou *Level of Details* é um recurso do VRML que permite implementar uma variação no nível de detalhes de objetos e/ou geometrias usado para melhorar o desempenho da navegação. A idéia básica é mostrar o objeto com uma maior resolução quando o usuário encontra-se "perto" deste e em menor resolução caso contrário.

que a seção circular seja convertida como um hexágono, o modelo completo teria um número de triângulos na ordem de um bilhão.

7.3 PROCESSAMENTO PONTO-A-PONTO

Entre os métodos utilizados em luminotécnica, o método ponto-a-ponto é o método mais indicado no dimensionamento da iluminação direta em áreas externas, explica Moreira (1999). Esse é o caso de uma subestação. Por meio desse método, obtém-se a iluminância realizada por uma ou mais fontes de luz, em qualquer ponto desejado, ou em uma malha de pontos.

Para aplicar o método ponto-a-ponto é preciso conhecer as características fotométricas de cada fonte de luz utilizada. Essas características são geralmente fornecidas pelos fabricantes. Para cada fonte de luz utilizada, os dados fotométricos que serão utilizados no cálculo estão contidos em um arquivo IES (*Illuminating Engineering Society*) correspondente. Um arquivo IES possui o formato que é padrão internacional para transferência de informações fotométricas digitalizadas. Também são necessárias as informações geométricas do sistema de iluminação, ou seja, o ponto $L(X_L, Y_L, Z_L)$ em que cada luminária está localizada, o ponto $D(X_D, Y_D, Z_D)$ para onde está focalizada e o(s) ponto(s) $P(X_P, Y_P, Z_P)$ no qual se deseja saber a iluminância (MOREIRA, 1988). Dessa maneira, um programa para processar o método ponto-a-ponto deverá receber essas informações como entrada.

No programa *ProcessaIES*, a entrada de dados é feita por meio de um arquivo no formato de texto ASCII contendo as informações sobre o sistema de iluminação projetado. As informações contidas nele são: as dimensões da malha estruturada (retilínea) que será gerada, o número de divisões em cada direção (x-y-z), a origem da malha, o número de fontes de luz diferentes utilizadas no projeto, o número total de pontos de luz, a lista de arquivos IES das diferentes luminárias utilizadas, a lista das luminárias com a posição e orientação de cada uma e o índice do arquivo IES correspondente. O formato do arquivo de entrada a ser montado está esquematizado na figura 14 e um exemplo de arquivo de entrada na figura 15.

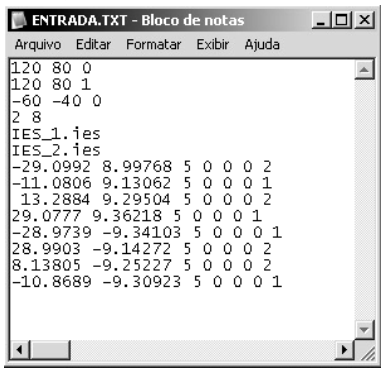
FIGURA 14 - FORMATO DO ARQUIVO DE ENTRADA

```
<Dimensão em X> <Dimensão em Y> <Dimensão em Z>
<Divisões em X> <Divisões em Y> <Divisões em Z>
<X da origem> <Y da origem> <Z da origem>
<# de fontes de luz diferentes> <# total de fontes de luz>
<nome do arquivo IES de índice 1>
<nome do arquivo IES de índice 2>
<nome do arquivo IES de índice 3>
...
<X da luz 1> <Y da luz 1> <Z da luz 1> <X do foco> <Y do foco> <Z do foco> <Índice do seu IES>
<X da luz 2> <Y da luz 2> <Z da luz 2> <X do foco> <Y do foco> <Z do foco> <Índice do seu IES>
<X da luz 3> <Y da luz 3> <Z da luz 3> <X do foco> <Y do foco> <Z do foco> <Índice do seu IES>
<X da luz 4> <Y da luz 4> <Z da luz 4> <X do foco> <Y do foco> <Z do foco> <Índice do seu IES>
...
```

FONTE: O autor.

Ao ser executado, o programa *ProcessaIES* calcula a iluminância resultante da iluminação direta em cada ponto da malha escolhida e escreve um arquivo com formato VTK (ver item 5.2.2) que poderá ser interpretado por um aplicativo de visualização baseado no *Visualization Toolkit* (VTK).

FIGURA 15 - EXEMPLO DE UM ARQUIVO DE ENTRADA



FONTE: O autor.

7.3.1 Detalhes do algoritmo ponto-a-ponto

O método ponto-a-ponto consiste em se obter o valor da iluminância total que incide em cada ponto de interesse por meio da aplicação da lei de Lambert (inverso do quadrado da distância). A implementação do método requer o conhecimento da distribuição fotométrica de cada fonte de luz cuja parte da intensidade luminosa emitida atinja o ponto diretamente.

A distribuição fotométrica de uma fonte de luz pode ser fornecida por meio de curvas e diagramas impressos no catálogo do produto ou por meio de arquivos digitais padronizados.

A iluminância resultante em um ponto *P* sobre uma superfície qualquer é dada pela equação:

$$E_{total} = \sum_{i=1}^{N} E_i \ , \tag{1}$$

onde:

N = número de fontes de luz;

E_i = iluminância realizada pela fonte de luz *i* no ponto *P*.

A iluminância E_i realizada por cada fonte de luz (Figura 16), individualmente, é dada pela equação:

$$E_i = \frac{I \cos\gamma}{d^2} \ , \tag{2}$$

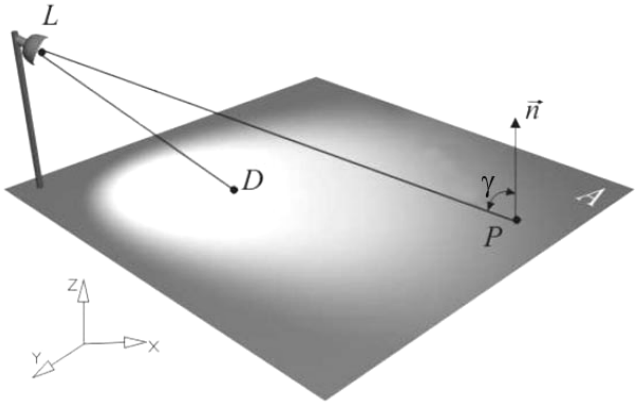
onde:

I = intensidade luminosa que a fonte de luz *i* emite na direção do ponto *P*;

γ = ângulo entre a normal da superfície considerada e a direção do ponto *P*;

d = distância entre a fonte de luz e o ponto *P*.

FIGURA 16 - ESQUEMA DE MONTAGEM DE UMA FONTE DE LUZ



FONTE: O autor.

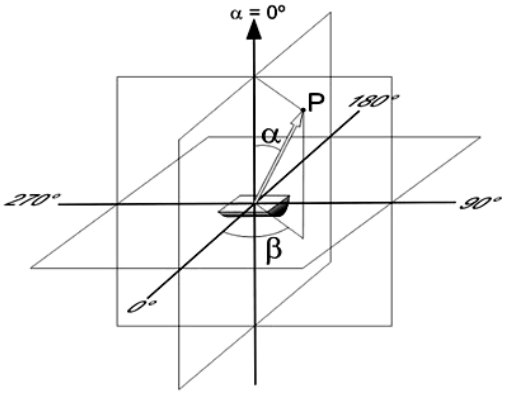
A intensidade *I* que uma fonte de luz emite em uma determinada direção é dada pela distribuição fotométrica dessa fonte de luz, e varia para cada modelo de luminária e lâmpada utilizadas. A distribuição fotométrica de uma fonte de luz é obtida por meio de medições em laboratório e geralmente é fornecida pelo seu fabricante em diferentes formas: tabelas, diagramas ou gráficos. Uma das dificuldades em automatizar os cálculos das iluminâncias é obter os dados para entrar com as distribuições de intensidades no sistema.

Geralmente, os dados fotométricos de uma luminária são disponibilizados por meio de curvas definidas sobre dois ou três planos específicos podendo ser imprecisos e de difícil utilização, principalmente quando a distribuição se dá sem simetria. A forma mais conveniente de entrar com dados fotométricos é através da utilização de arquivos IES. No entanto, nem todos os fabricantes de luminárias possuem os dados dos seus produtos nesse formato (Figura 17).

Em um arquivo IES, a direção é dada em coordenadas esféricas e, portanto, definida por dois ângulos como mostrado na figura 17.

Assim, dada uma luminária, com localização e direcionamento definidos, e possuindo seu arquivo IES, basta obter os ângulos correspondentes e consultar o arquivo IES para saber a intensidade luminosa emitida na direção de um determinado ponto. No caso da direção do ponto não estar definida no arquivo, pode-se realizar uma interpolação.

FIGURA 17 - SISTEMA DE COORDENADAS E EXEMPLO DE ARQUIVO IES

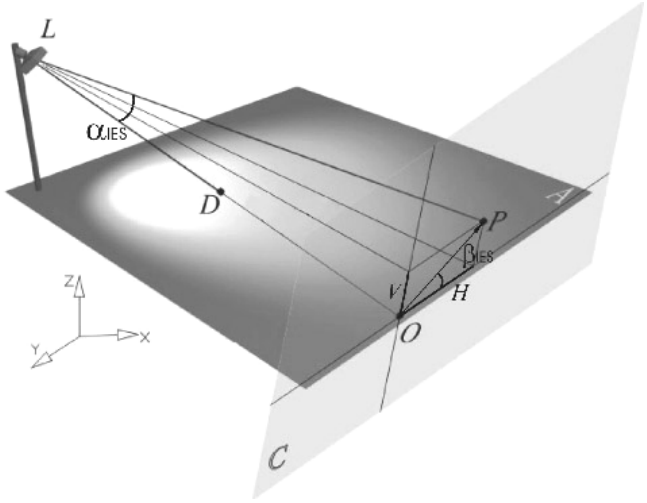



```
IESNA:LM-63-1995
[TEST]
[DATE] 02-01
[MANUFAC] GE LIGHTING
[LAMPCAT] 12749
[LAMP] MXR100/U/PAR/WFL
TILT=NONE
1 5700 1.00 29 1 1 1 0 0 0
1 1 0
0 1.25 2.5 3.75 5 6.25 7.5 8.75 10 12.5 15 17.5 20 22.5 25
27.5 30 35 40 45 50 55 60 65 70 75 80 85 90
0
5000 5003 5012 5025 5038 5052 5060 5063 5052 5002 4895
4720 4474 4151 3767 3338 2892 2055 1438 1019 725 514 398
326 240 147 80 41 25
```

FONTE: O autor.

A figura 18 ilustra um esquema que permite o cálculo dos ângulos α_{IES} e β_{IES} para entrada no arquivo IES.

FIGURA 18 – ESQUEMA PARA OBTER OS ÂNGULOS α_{IES} E β_{IES}.



FONTE: O autor.

Primeiro encontra-se o plano *C*, ortogonal à direção do foco, e que passa pelo ponto *P*. O ponto *O* é a intersecção deste plano com a reta que dá a direção *LD* de foco. Sobre o plano *C* e passando pelo ponto *O*, encontra-se o segmento de reta horizontal *H* e ortogonal a ele o segmento *V*. O ângulo entre *H* e o vetor *OP* é o β_{IES}. O ângulo entre as retas LO e LP é o ângulo α_{IES}.

Como algoritmo para um aplicativo que realize o cálculo da iluminação direta em um ponto deve-se seguir os seguintes passos:

1. Escolher um conjunto lâmpada/luminária na base de dados;
2. Entrar com o ponto $L(X_L, Y_L, Z_L)$ em que a luminária está instalada;
3. Entrar com o ponto $D(X_D, Y_D, Z_D)$ de foco;
4. Voltar para 1 até que todas as luminárias sejam instaladas;
5. Entrar com o ponto $P(X_P, Y_P, Z_P)$ onde se deseja saber a iluminância;
6. Obter os ângulos α_{IES} e β_{IES} correspondente à primeira luminária;

7. Consultar o arquivo IES da luminária e obter o valor de intensidade *I* referente à direção do ponto *P*;
8. Calcular a iluminância por meio da equação (2);
9. Repetir os passos 6 até 9 para cada luminária *i=2, 3, 4,..., n*;
10. Somar os valores de iluminância exercidas por cada luminária no ponto *P* usando a equação (1). O resultado é a iluminância total no ponto *P*, dada em lux.

7.4 VISUALIZAÇÃO DE ARQUIVOS IES

Como já foi citado, um dos padrões existentes de formato de arquivo para distribuição de informações fotométricas de fontes de luz (lâmpada/luminária) é o padrão IES (*Illuminating Engineering Society*). Esse padrão é aceito por muitos *softwares* para projeto de iluminação como o *Calculux* e o *DIALux*, e também por alguns *softwares* que utilizam algoritmos como o *Ray Tracing* para renderização realística de efeitos de luz, sendo um exemplo o programa *3D Studio 7*.

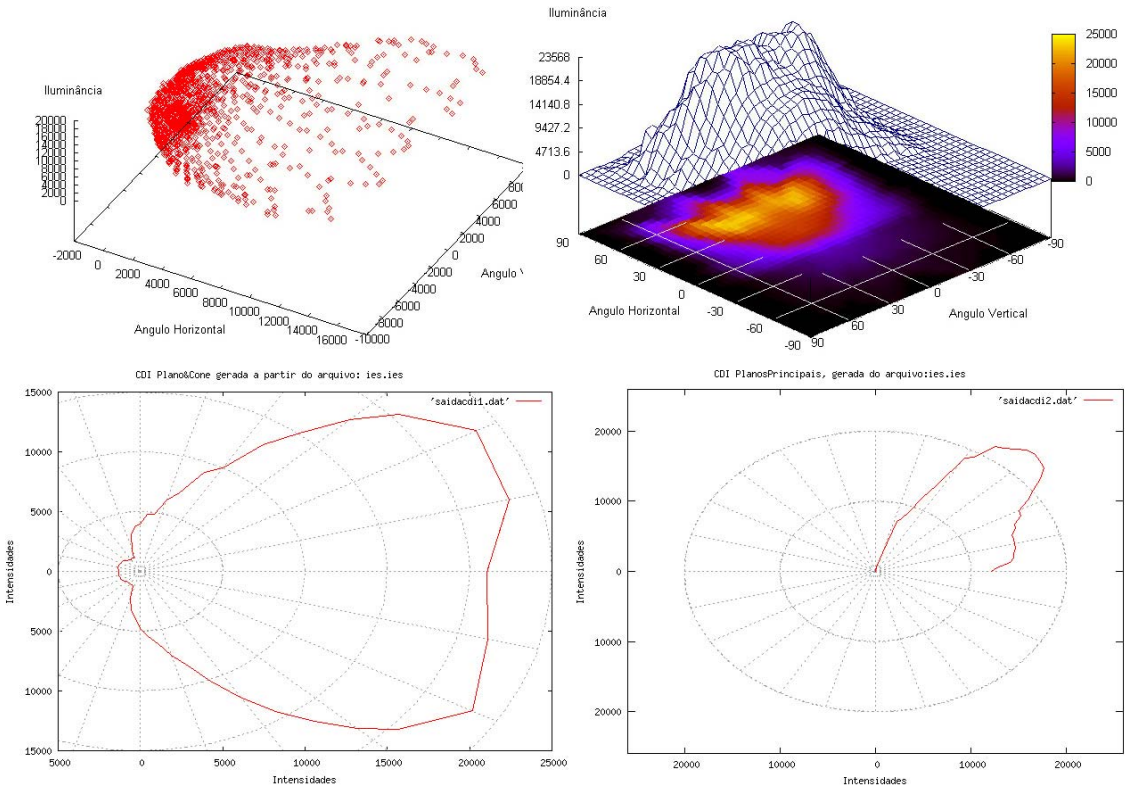
Um dos interesses de projetistas, ao escolher uma luminária adequada às suas necessidades, é poder ver como se dá a distribuição fotométrica de diferentes luminárias a fim de escolher, fazer uma comparação ou simplesmente ter uma visão geral de como se dá a distribuição de intensidades de uma determinada luminária. Essa distribuição é 3D mas geralmente, mesmo com o uso de *softwares*, a visualização é feita em 2D, apenas para planos específicos.

O arquivo IES utiliza o sistema de coordenadas esféricas para definir o "volume" de distribuição de intensidades luminosas. Nos catálogos de luminárias, esse volume de intensidades é comumente transcrito para um sistema de coordenadas ortogonais e faz-se uso de isolinhas e planos de corte. Neste trabalho, desenvolveu-se um aplicativo chamado *InterpretaIES*, o qual efetua a leitura de um arquivo IES e apresenta a visualização 3D dos dados contidos nesse arquivo. Os dados são mostrados no sistema de coordenadas cartesianas e esféricas. O programa também gera as curvas

fotométricas para os planos principais e um diagrama de distribuição de intensidades que utiliza representação por meio de isolinhas.

O aplicativo *InterpretaIES*, portanto, lê o arquivo IES, faz as transformações de coordenadas e a extração dos planos de corte, escreve um *script* com os ajustes de parâmetros adequados, e faz chamada ao programa GnuPlot para prover a visualização 3D interativa. O *script* contém linhas de comando para o programa GnuPlot. A figura 19 mostra as "saídas" do programa, ou melhor, mostra as diferentes formas de visualização 2D e 3D providas.

FIGURA 19 - VISUALIZAÇÃO DE UM ARQUIVO IES



FONTE: O autor.

7.5 VISUALIZAÇÃO DE CAMPOS DE ILUMINÂNCIA

A visualização de campos de iluminância é um caso de visualização de campos escalares estáticos. Dentre as técnicas de visualização científica para

visualização de campos escalares 3D está a obtenção de isosuperfícies, podendo ser associadas ao uso de transparências, extração de planos de corte e correspondente mapeamento de cores ou extração de isolinhas, ícones pontuais e renderização de volumes.

No caso específico do programa *ProcessaIES*, os dados contidos no arquivo de saída estão definidos sobre um malha estruturada e ortogonal, as células utilizadas são do tipo *voxel* ou *pixel* do VTK. Estruturados dessa forma é possível, com o uso do VTK, gerar texturas a partir de malhas 2D. O uso de textura, ao invés de cores indexadas aos vértices de polígonos, pode melhorar o desempenho do aplicativo.

Nos itens seguintes, é descrita a experiência com o uso da biblioteca VTK para visualização dos campos de iluminância e geração dos arquivos de saída em VRML e JPEG. São relatadas as dificuldades encontradas e algumas conclusões parciais a respeito da implementação dos algoritmos para visualização. Alguns aspectos relativos ao desenvolvimento de ambientes de visualização interativos também são abordados.

7.5.1 Mapeamento de cores

Desenvolveu-se um algoritmo, utilizando-se a ferramenta VTK, para ler e prover a visualização, por mapeamento de cores, dos dados escalares que estão armazenados em um arquivo no formato *Legacy* do VTK gerado pelo aplicativo *ProcessaIES*. O algoritmo de visualização também permite a exportação para os formatos VRML, o que viabiliza a visualização em sistemas de RV, e para o formato JPEG (imagem 2D), com a finalidade de armazenar a barra de cores. O código completo encontra-se no apêndice 2, e alguns trechos desse código foram adaptados de Adaime (2005).

O aplicativo para visualização do mapeamento de cores desenvolvido, chamado aqui de *ColorMap*, foi escrito na linguagem C++ utilizando o *Microsoft Visual C++ 2005 Express Edition*, disponibilizado gratuitamente durante um ano. A versão do VTK utilizada foi a 4.2 e a sua instalação na plataforma Win32 é descrita no apêndice 1, para que o leitor interessado possa mais facilmente repetir esta

experiência. Outras informações que podem auxiliar na instalação do VTK podem ser obtidas em Li (2005).

As classes do VTK utilizadas estão ilustradas no trecho de código que aparece na figura 20.

FIGURA 20 – CLASSES DO VTK USADAS

```
#include "vtkActor.h"
#include "vtkCubeAxesActor2D.h"
#include "vtkCamera.h"
#include "vtkOutlineFilter.h"
#include "vtkPoints.h"
#include "vtkRenderWindow.h"
#include "vtkRenderWindowInteractor.h"
#include "vtkRenderer.h"
#include "vtkStructuredPoints.h"
#include "vtkStructuredPointsToPolyDataFilter.h"
#include "vtkDataSetMapper.h"
#include "vtkDataSetToPolyDataFilter.h"
#include "vtkLookupTable.h"
#include "vtkPointData.h"
#include "vtkVRMLExporter.h"
#include "vtkScalarBarActor.h"
#include "vtkStructuredPointsWriter.h"
#include "vtkStructuredPointsReader.h"
#include "vtkTextProperty.h"
#include "vtkVRMLExporter.h"
#include "vtkWindowToImageFilter.h"
#include "vtkJPEGWriter.h"
```

FONTE: O autor.

Para ilustrar a utilização da ferramenta desenvolvida foi montado um esquema de iluminação composto de dois refletores e lâmpadas de marca Philips, cujos arquivos IES foram fornecidos pelo próprio fabricante mediante solicitação via *e-mail*. São eles o refletor Philips Tempo3 Simétrico e o Philips Tempo3 Assimétrico, com a lâmpada SON-T 400W, de vapor de sódio em alta pressão. Gerou-se uma simulação para uma malha 2D com 5151 pontos e os resultados foram visualizados por meio de mapeamento de cores.

O arquivo de entrada de dados utilizado para esta simulação é mostrado na figura 21.

FIGURA 21 – ARQUIVOS DE ENTRADA

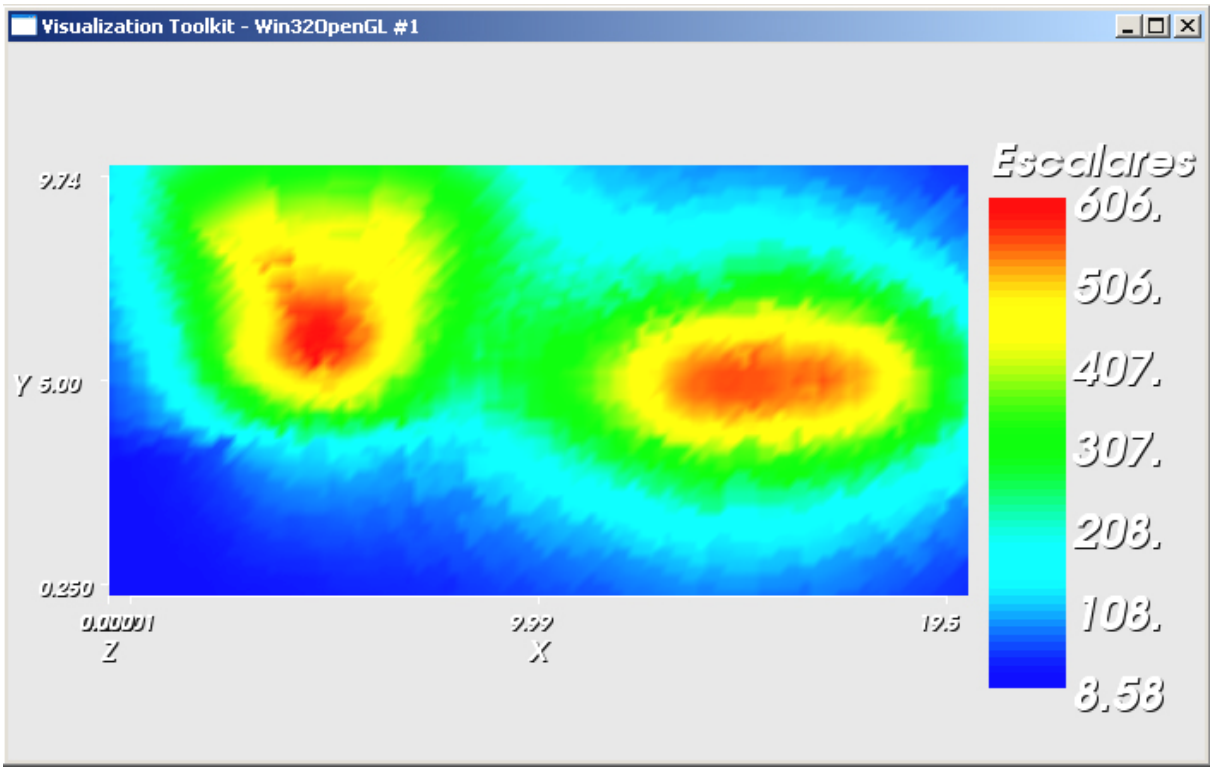
```
20 10 0
100 50 10
0 0 0
2 2
SWF330_400KA_45_01xSON-T400WCON.ies
SWF330_400KS1xSON-T400WCON.ies
5 5 5 5.01 5 0 1
15 5 7 15.01 5 0 2
```

FONTE: O autor.

O arquivo contém as informações da malha (dimensões, número de divisões e origem) descritas nas três primeiras linhas. A quantidade de pontos de luz e de modelos de arquivos IES utilizados estão descritos na quarta linha. Seguem a lista com os nomes dos arquivos e a lista de pontos de luz (com as coordenadas de posição, de ponto de foco e o índice do IES correspondente).

Os resultados obtidos da visualização das iluminâncias, para o esquema descrito na figura 21, são mostrados na figura 22.

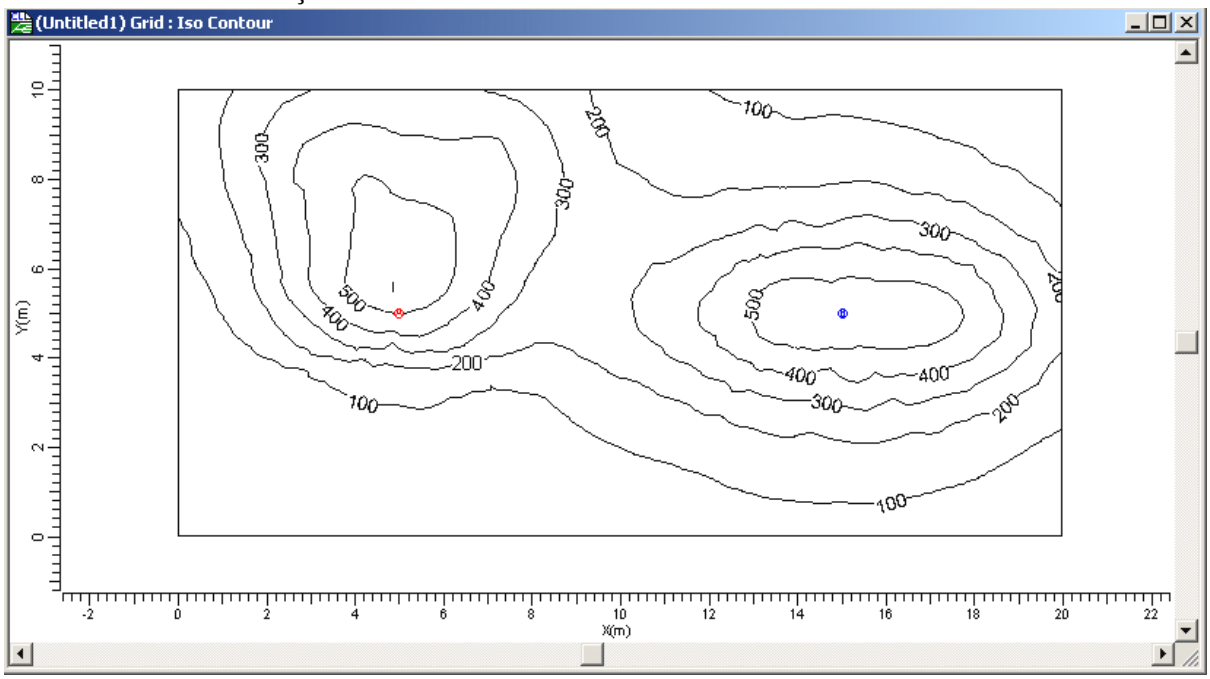
FIGURA 22 – VISUALIZAÇÃO POR MAPEAMENTO DE CORES



FONTE: O autor.

Com a finalidade de validar o algoritmo *ProcessaIES*, gerou-se uma simulação, com os mesmos dados de entrada, utilizando o programa *CalculuX*. O resultado é mostrado na Figura 23. O programa *CalculuX* não fornece a visualização dos resultados por meio de mapeamento de cores, tampouco permite uma interação com os dados representados, somente fornece um diagrama de isolinhas e imagens estáticas podendo ser em perspectiva. O aplicativo *ColorMap* provê a visualização por mapeamento de cores e pode ser adaptado para gerar as isolinhas. Além disso, também provê a visualização 3D interativa, a geração de uma textura e a exportação para o formato VRML, ampliando as possibilidades de visualização em diferentes sistemas.

FIGURA 23 – SIMULAÇÃO COM O PROGRAMA CALCULUX

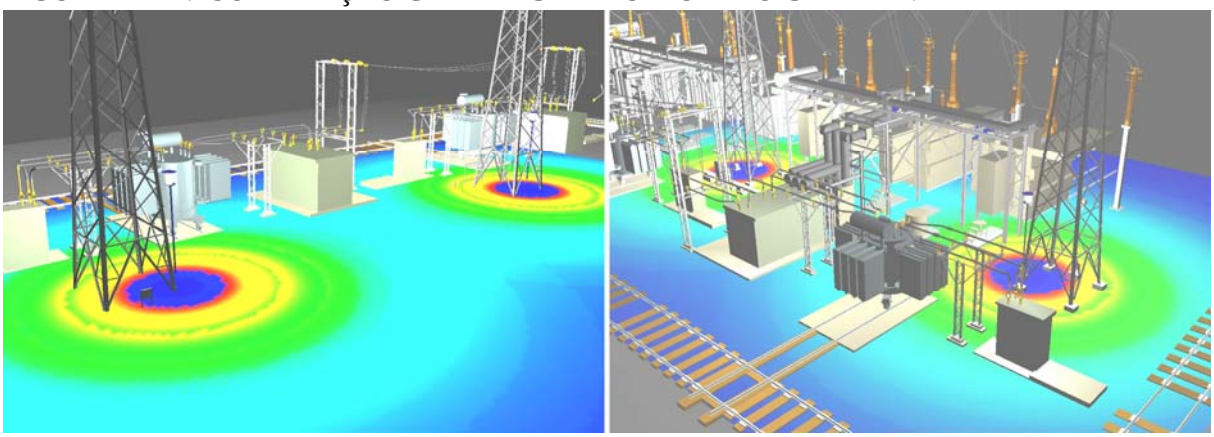


FONTE: O autor.

Para construir a visualização do modelo CAD simultaneamente à visualização dos dados de iluminância, basta exportar os planos, ou malhas 2D, gerados na simulação para o formato VRML, com os pontos de luz posicionados corretamente, e adicioná-los ao modelo da subestação por meio de um nó *Inline* do VRML. A classe *vtkVRMLexporter* do VTK faz a exportação para VRML do conteúdo 3D gerado pelo algoritmo *ColorMap*. Também é possível gerar uma textura a partir da malha com os

valores escalares obtidos. O uso de textura diminui a quantidade de polígonos a serem renderizados na cena melhorando o desempenho final da visualização. O procedimento para obtenção da textura é descrito no item 7.2.3. Exemplos foram gerados utilizando-se o modelo 3D de uma subestação de energia elétrica (Figura 24).

FIGURA 24 – VISUALIZAÇÃO CIENTÍFICA E DO MODELO CAD EM VRML



FONTE: O autor

7.5.2 Isosuperfícies

Outro aplicativo desenvolvido com a ferramenta VTK, chamado *Isosup3D*, realiza a leitura do arquivo de dados 3D, a extração de isosuperfícies, a exportação para o formato VRML, e provê a visualização dos resultados obtidos. O código completo encontra-se no apêndice 3. Alguns trechos de código foram adaptados de Adaime (2005). Este aplicativo também foi escrito na linguagem C++, compilado utilizando o *Microsoft Visual C++ 2005 Express Edition* (MICROSOFT, 2005) e a versão do VTK foi a 4.2.

As classes do VTK utilizadas estão ilustradas no trecho de código que aparece na figura 26.

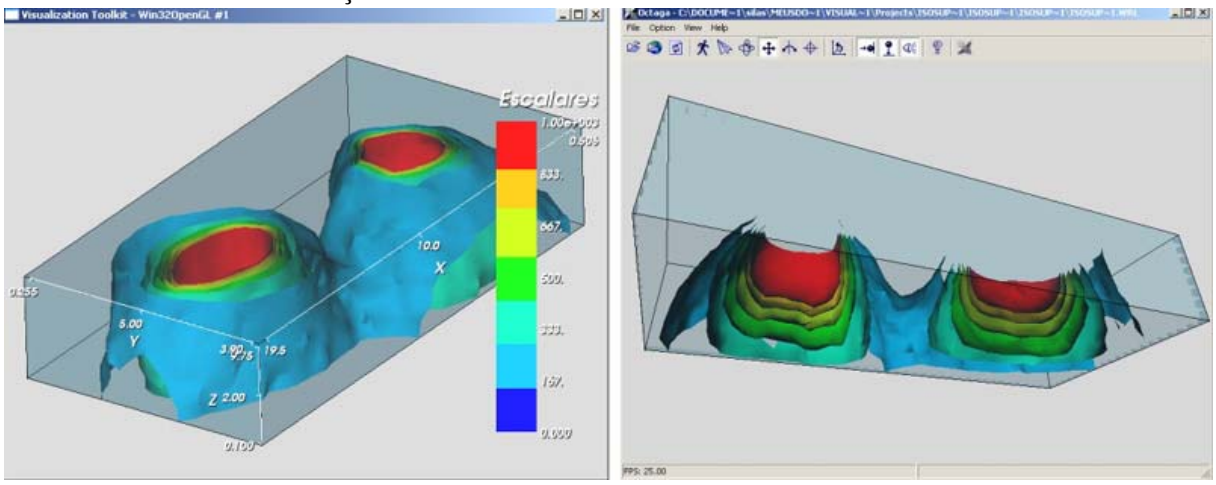
FIGURA 26 – CLASSES DO VTK UTILIZADAS

```
#include "vtkActor.h"
#include "vtkCamera.h"
#include "vtkCubeAxesActor2D.h"
#include "vtkPolyData.h"
#include "vtkPolyDataMapper.h"
#include "vtkRenderWindow.h"
#include "vtkRenderWindowInteractor.h"
#include "vtkRenderer.h"
#include "vtkUnstructuredGrid.h"
#include "vtkContourGrid.h"
#include "vtkContourFilter.h"
#include "vtkDataSetMapper.h"
#include "vtkDataSetToPolyDataFilter.h"
#include "vtkLookupTable.h"
#include "vtkProperty.h"
#include "vtkScalarBarActor.h"
#include "vtkStructuredPoints.h"
#include "vtkStructuredPointsToPolyDataFilter.h"
#include "vtkStructuredPointsWriter.h"
#include "vtkStructuredPointsReader.h"
#include "vtkTextProperty.h"
#include "vtkUnstructuredGridWriter.h"
#include "vtkUnstructuredGridReader.h"
#include "vtkPolyDataNormals.h"
#include "vtkStripper.h"
#include "vtkVRMLExporter.h"
```

FONTE: O autor.

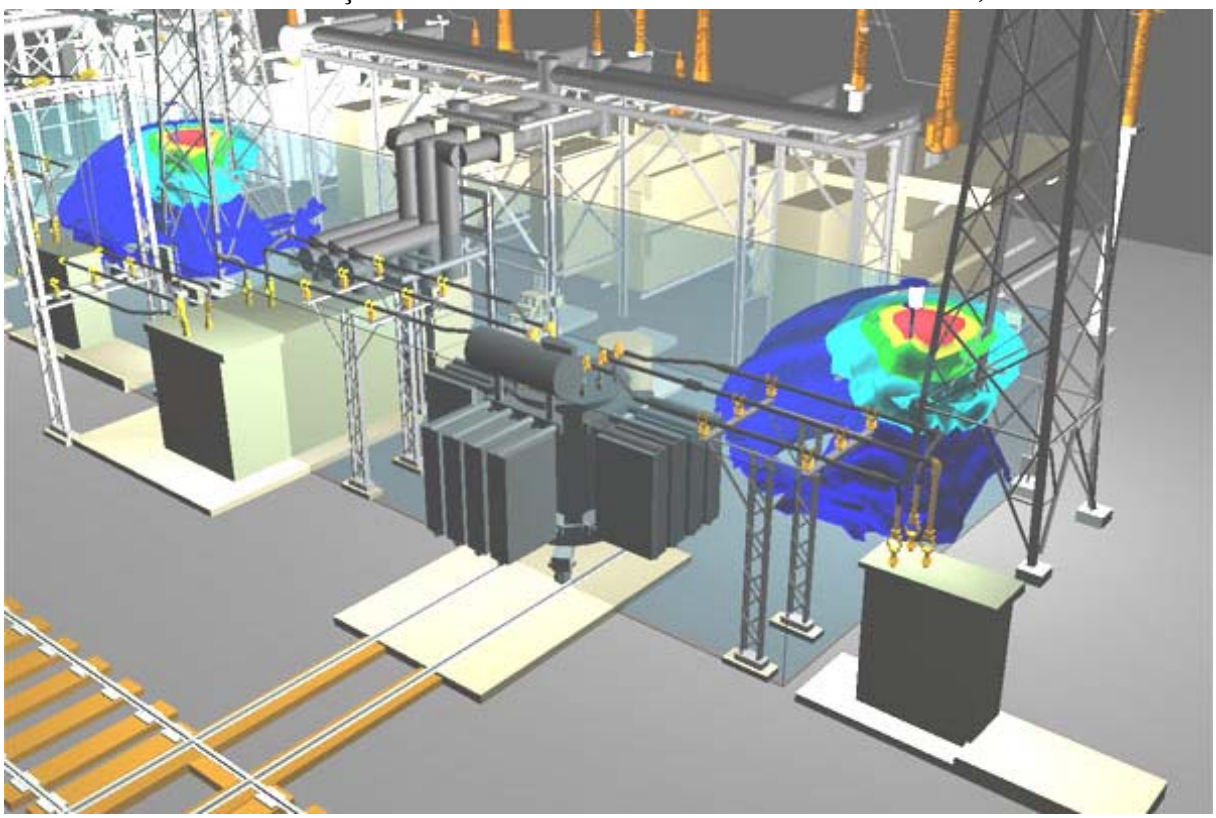
Novamente, para ilustrar a utilização da ferramenta desenvolvida, foi gerado um esquema de iluminação utilizando os mesmos dados de entrada utilizados na simulação descrita no item 7.4.1 (Figura 21), mas agora com uma malha 3D. Então, gerou-se a simulação para uma malha 3D com 7749 pontos e os resultados, como já foi dito, foram visualizados por maio da extração de isosuperfícies e mapeamento de cores (Figuras 27e 28).

FIGURA 27 – VISUALIZAÇÃO DE ISOSUPERFÍCIES



FONTE: O autor

FIGURA 28 – VISUALIZAÇÃO DAS ISOSUPERFÍCIES E DO MODELO CAD, EM VRML



FONTE: O autor.

7.5.3 Integração com o SolidWorks

O processamento numérico para o cálculo das iluminâncias, como já foi citado, requer algumas informações de entrada. Essas informações são sobre as características técnicas dos equipamentos utilizados, configuração (ou geometria) do esquema projetado, ou seja, a localização de cada fonte de luz e seu posicionamento, e também as características da malha, o que fornecerá as coordenadas dos pontos em que será calculada a iluminância. Entretanto, diversos navegadores VRML não suportam a captura dos dados (coordenadas de pontos), necessários ao cálculo das iluminâncias pelo método ponto-a-ponto, relativos à montagem do esquema de iluminação.

Assim, vislumbrou-se a utilização de um sistema CAD 3D como interface para a criação do esquema de iluminação, ou seja, para a escolha, inserção e

direcionamento das fontes de luz. O ambiente CAD 3D permite visualizar o modelo da subestação e, interativamente, criar o esquema de iluminação, “capturando” as coordenadas dos pontos no espaço 3D em que serão instaladas as fontes de luz, e também, determinar a direção de foco de cada luminária.

Por meio da customização de um sistema de CAD 3D, foi possível criar funcionalidades para acessar e editar uma base de dados com modelos de fontes de luz (conjunto lâmpada/luminária), e assim inserir e direcionar diversas luminárias no modelo da subestação de forma intuitiva, fazendo uso de funcionalidades do CAD, como ferramentas para extração de distâncias.

O software CAD 3D escolhido foi o SolidWorks, e para a integração foi escolhida a linguagem de programação C# em ambiente .NET. Foram adotadas quatro subdivisões para o software desenvolvido. A primeira, cobrindo o cadastro de informações acerca dos conjuntos lâmpada-luminária em banco de dados SQL. A segunda, compreendendo a montagem do esquema que será carregado no algoritmo ponto-a-ponto, incluindo a definição das coordenadas e escolha dos conjuntos utilizados. A terceira subdivisão trata do processamento do método ponto-a-ponto e a quarta e última subdivisão cobre o pós-processamento (visualização) dos resultados no ambiente CAD 3D e exportação para o formato VRML.

Com relação à primeira divisão, foi estendida a GUI (*Graphics User Interface*) do software SolidWorks (SW) utilizando a API (*Application Programming Interface*) de desenvolvimento do pacote CAD 3D, disponível com o *software*. Foram desenvolvidas ‘páginas’ (interfaces adicionais do SolidWorks) para o cadastro separado de lâmpadas, luminárias, conjuntos lâmpada-luminária e também informações adicionais, como lista de fabricantes (ambivalentes quanto a lâmpadas e luminárias), lista de bases (compatível com o soquete da luminária) e lista de tipos de lâmpadas.

A montagem do esquema, que consiste na segunda divisão, utilizou os recursos do CAD 3D para a captura dos pontos. Foi criada uma ‘página’ para a escolha do conjunto lâmpada-luminária, definição do nome, e escolha do ponto de origem e ponto de foco. Dentro dessa página é possível salvar e carregar esquemas de

iluminação. A captação dos pontos é feita de forma automática e gera as coordenadas que serão utilizadas no algoritmo ponto-a-ponto. O esquema então é salvo em um arquivo XML e pode ser carregado sempre que necessário. Um exemplo do arquivo que armazena o esquema de iluminação é mostrado na figura 29.

FIGURA 29 – ARQUIVO XML PARA ARMAZENAMENTO DO ESQUEMA DE ILUMINAÇÃO

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
- <file version="1.04" offline="False">
    <simulation>Apresentação</simulation>
    <lights count="2" />
  - <light omni="True" enabled="True" web="True">
      <name>Luz1</name>
      <part>ND</part>
      <model>CW-250</model>
      <position>11,8438868228314 -17,5568378378784 4,77116310967438</position>
    </light>
  - <light omni="True" enabled="True" web="True">
      <name>Luz2</name>
      <part>ND</part>
      <model>CW-250</model>
      <position>23,7518922407892 0,609433914421743 4,78999533167973</position>
    </light>
  </file>
```

FONTE: O autor.

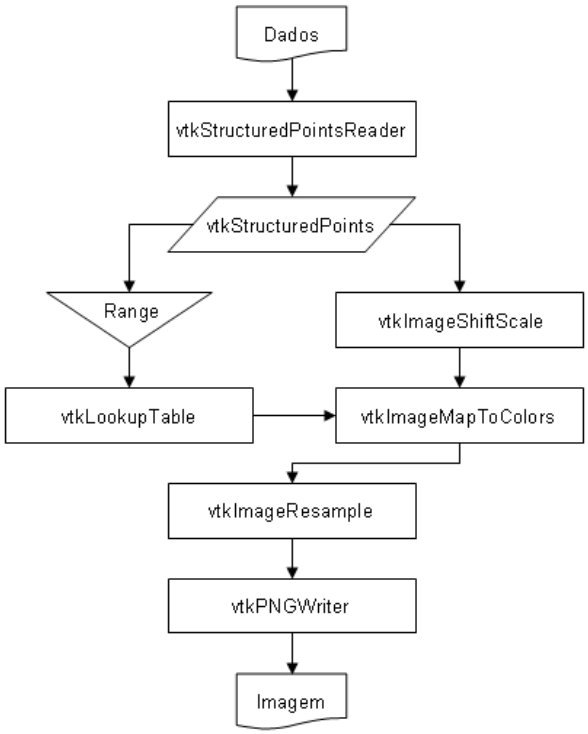
A terceira etapa consiste na transferência das coordenadas das posições e de direções de foco das fontes de luz utilizadas no projeto, obtidas a partir do *software* SolidWorks, para o algoritmo do método ponto-a-ponto, o qual é executado de forma transparente ao usuário. Os resultados são salvos como um arquivo VTK para posterior visualização. A última etapa consiste na leitura do arquivo VTK e apresentação, ou visualização, dos resultados sobre o plano selecionado por meio de um mapeamento de cores e geração de uma textura.

Para a geração da textura foram utilizadas as classes do VTK mostradas na figura 30, que ilustra o fluxo de dados até a geração da textura.

A utilização do *software* CAD 3D veio a ser uma alternativa à entrada dos dados na forma de um arquivo ASCII, não sendo, entretanto, essencial. O algoritmo de processamento numérico pode ler as informações de entrada em um arquivo ASCII (ou XML, no caso de estar integrado ao SolidWorks) e gerar o arquivo VTK. O algoritmo de visualização, baseado no VTK, gera o mapeamento de cores e/ou a extração de

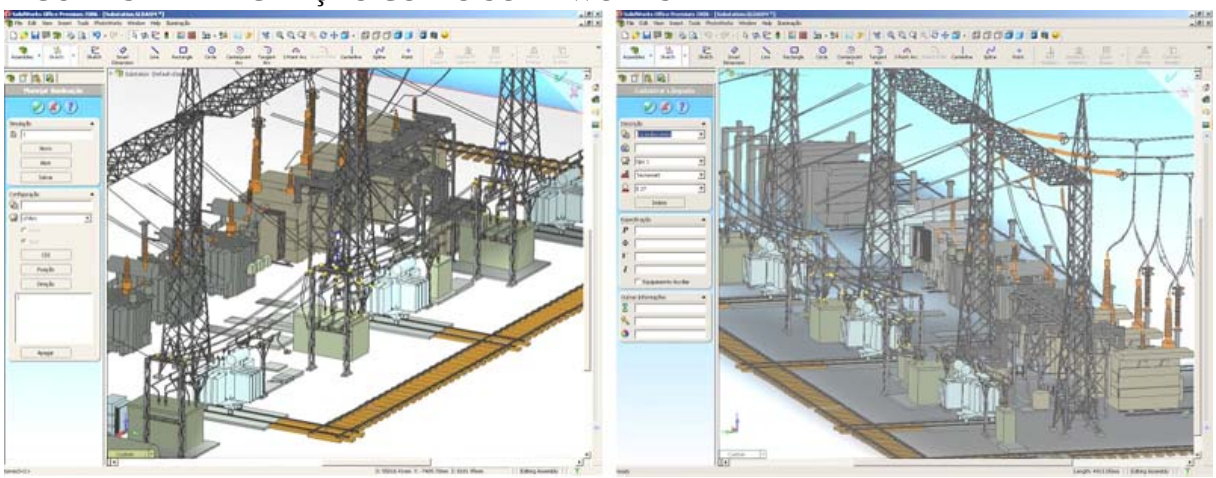
isocontornos e também exporta o resultado para o formato VRML, podendo, dessa forma, ser visualizado em um navegador adequado ou em um sistema de RV.

FIGURA 30 – FLUXO DE DADOS PARA A GERAÇÃO DA TEXTURA



FONTE: O autor

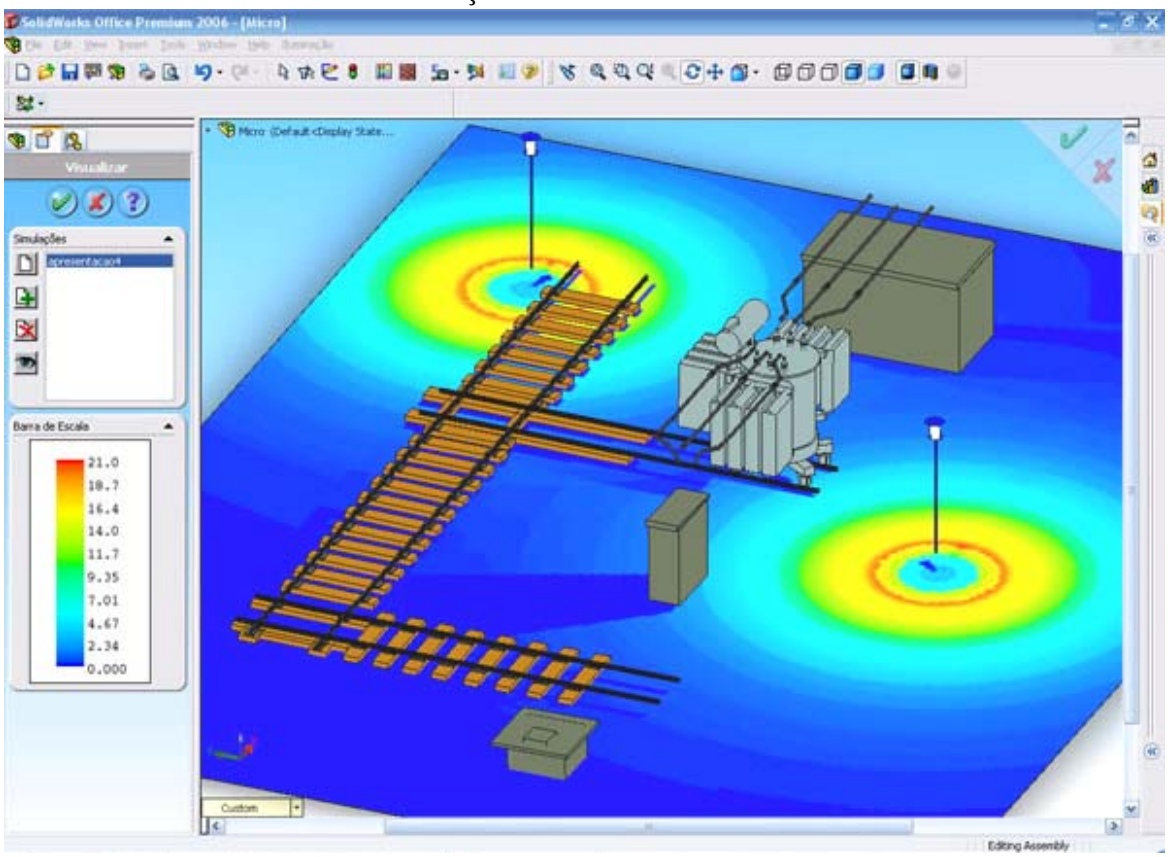
FIGURA 31 – INTEGRAÇÃO COM O SOLIDWORKS



FONTE: O autor

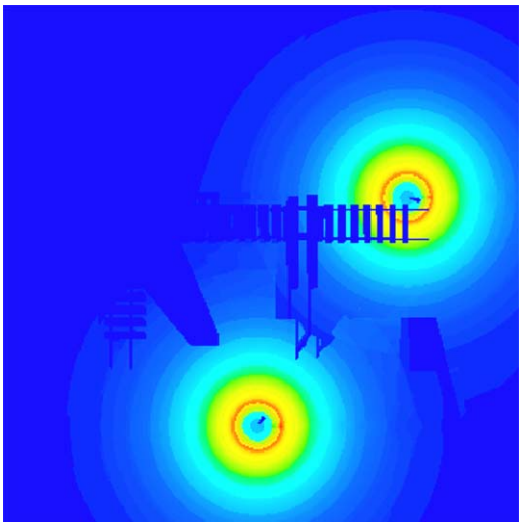
Também foi possível, utilizando métodos da API (*Application Programming interface*) do SolidWorks, realizar a simulação levando em consideração a existência de corpos opacos que impedem a passagem da luz em algumas regiões da malha. Os métodos utilizados foram o *RayIntersections* e o *GetRayIntersectionPoints* que permitem, dados um ponto e uma direção, verificar se existe algum objeto nessa direção. Com esses métodos foi possível analisar se há ou não algum objeto entre o ponto de instalação da fonte de luz e o ponto da malha no qual está sendo calculada a iluminância. Caso haja um objeto, a influência dessa fonte de luz será desconsiderada, ou seja, somente será computada a influência das fontes de luz cujos raios luminosos incidem diretamente no ponto. Um exemplo de uma simulação realizada e visualizada no ambiente do CAD, considerando as obstruções, é mostrada na figura 32 e a textura gerada é mostrada na figura 33.

FIGURA 32 – EXEMPLO DE SIMULAÇÃO REALIZADA NO SOLIDWORKS



FONTE: O autor

FIGURA 33 - TEXTURA GERADA



FONTE: O autor

Uma das dificuldades encontradas foi o longo tempo de processamento necessário para realizar a simulação considerando a análise de obstruções. A malha utilizada na simulação da figura 31 tinha 169744 pontos e levou aproximadamente 1,5 horas em um computador PC com processador AMD 2800+ e 512 Mb de memória RAM. Otimizações no algoritmo, com a finalidade de reduzir o tempo de processamento, estão sendo trabalhadas atualmente. Uma idéia que está sendo implementada é desconsiderar a análise de interferência em pontos que estão a uma distância tal que torna a iluminância nesse ponto próxima de zero (ou menor que um valor arbitrado, como por exemplo, 0.01% da iluminância máxima contida no arquivo IES).

Mais testes e a adoção de uma métrica adequada para medir o desempenho desses algoritmos se fazem necessários para fazer uma avaliação mais precisa.

8 RESULTADOS EXPERIMENTAIS

No decorrer deste trabalho desenvolveu-se um aplicativo para processamento e pós-processamento de campos de iluminância na forma de três algoritmos, denominados: *ProcessaIES*, *ColorMap* e *Isosup3D*. O primeiro, *ProcessaIES*, realiza o processamento numérico dos campos de iluminâncias, em malhas estruturadas 2D ou 3D, para qualquer sistema de iluminação projetado, não possuindo limitação de número ou modelo de fontes de luz. O segundo algoritmo, chamado *ColorMap*, provê a visualização dos resultados gerados pelo algoritmo *ProcessaIES* por meio do mapeamento de cores, e exporta para os formatos VRML e JPEG. E finalmente o algoritmo *Isosup3D* provê a visualização dos dados de iluminância por meio da extração de isosuperfícies e exporta as geometrias geradas para VRML. Estes algoritmos constituem uma ferramenta de apoio a projetos de iluminação que utiliza, como interface de visualização científica, um ambiente virtual 3D no qual podem ser inseridos modelos provenientes de sistemas CAD.

Um quarto algoritmo, denominado *InterpretaIES*, foi implementado com a finalidade de visualizar dados da distribuição fotométrica de uma determinada fonte de luz contidos em um arquivo padronizado IES (*Illuminating Engineering Society*). Assim, possuindo o arquivo IES de uma luminária, é possível visualizar interativamente em 3D os dados desse arquivo, ou seja, a distribuição de intensidades luminosas dessa luminária.

Utilizou-se durante os experimentos, o modelo 3D de uma subestação de energia elétrica que foi obtido a partir de um escaneamento a laser e posterior modelagem geométrica, resultando em um modelo no formato DWG do *software* AutoCAD 2004. A partir do modelo no formato DWG gerou-se o modelo em VRML. Técnicas de otimização para melhorar o desempenho na visualização de grandes modelos, como a utilização de diferentes níveis de detalhes e redução do número de polígonos, foram implementadas. Com o modelo em VRML, otimizado, foi possível visualizar os dados obtidos de campos de iluminâncias, simultaneamente ao modelo geométrico da subestação. A visualização foi feita por meio da navegação exploratória

e interativa no ambiente virtual 3D.

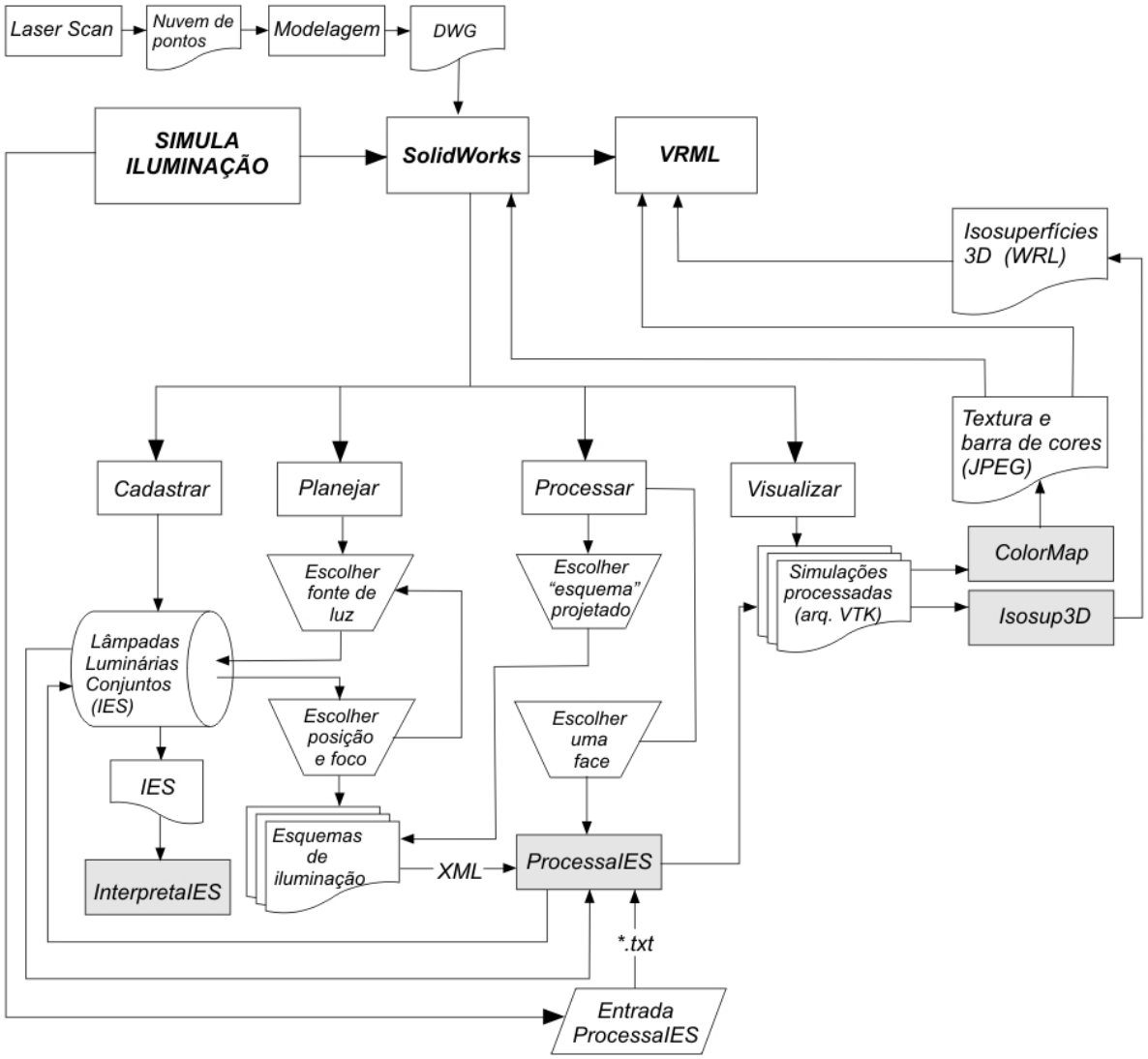
Em uma etapa posterior, implementou-se a integração dos algoritmos de processamento e pós-processamento de campos de iluminância com o *software* de CAD 3D SolidWorks 2006 (SW). Por meio da utilização de métodos da API (*Application Programming Interface*) do SW, foi possível realizar simulações da iluminação direta considerando a interferência de corpos opacos na direção do raio de luz. Uma limitação encontrada na análise de interferências foi o longo tempo de processamento necessário.

Para possibilitar a criação do sistema de iluminação, ou montagem do esquema (escolha, posição e direcionamento das fontes de luz), no ambiente de trabalho do SW, criou-se uma base de dados contendo informações de diferentes produtos (lâmpada/luminária) disponíveis no mercado e customizou-se o programa de forma a disponibilizar um menu implementado com essas funcionalidades. Feito isso, o sistema de iluminação pôde ser projetado interativamente no próprio ambiente CAD (escolha, inserção e direcionamento de pontos de luz), e os dados de iluminância resultantes puderam ser visualizados no ambiente CAD e, também, exportados para VRML para visualização em sistemas de Realidade Virtual.

A figura 34 descreve o esquema de funcionamento dos algoritmos desenvolvidos e sua utilização. A ilustração consiste em um diagrama que mostra a arquitetura da integração dos algoritmos desenvolvidos com o sistema de CAD 3D SolidWorks. Pode-se observar que a simulação da iluminação pode ser feita de duas formas distintas. A primeira delas é fornecendo ao algoritmo de processamento numérico, *ProcessaIES*, as informações de entrada por meio de um arquivo texto no padrão ASCII (*.txt). A segunda é gerando o “esquema” de iluminação (modelo, localização e posição de foco das fontes de luz) no ambiente de trabalho do CAD interativamente, selecionando as luminárias e inserindo-as em pontos escolhidos no modelo 3D (da subestação, por exemplo). O “esquema” montado então é armazenado em um arquivo XML. Essas duas maneiras geram os mesmos resultados. A diferença é que na primeira não faz uso do o SolidWorks para realizar a simulação e a visualização é feita em VRML.

Uma vez processada a simulação é gerado um arquivo VTK (*Legacy*) com os resultados. Esse arquivo pode ser facilmente interpretado por sistemas ou algoritmos de visualização baseados no VTK, como os algoritmos *ColorMap* e *Isosup3D*. Dessa maneira, a visualização desses dados pode ser feita de duas formas distintas: no próprio ambiente de trabalho de SolidWorks ou em um ambiente virtual em VRML.

FIGURA 34 – DIAGRAMA QUE ILUSTRA OS PROCESSOS PARA UMA SIMULAÇÃO



FONTE: O autor.

9 CONCLUSÕES

Concluiu-se que os objetivos foram atingidos, uma vez que os programas para visualização desenvolvidos puderam ser utilizados para visualizar os campos de iluminância juntamente com os modelos CAD da subestação. O uso de sistemas gráficos 3D, como softwares de CAD 3D e RV, possibilita uma visualização interativa e, conseqüentemente, uma percepção mais aprofundada e intuitiva sobre conteúdos 3D. Em particular, o uso da linguagem VRML permite uma visualização e a edição de conteúdo 3D utilizando-se *softwares* (navegador e editor de texto ASCII) de fácil acesso e aquisição, sendo muitos deles gratuitos. Assim, dados científicos e modelos CAD podem ser combinados em um modelo VRML a fim de se obter um ambiente de visualização e interação de apoio a aplicações diversas, como por exemplo, em projetos de iluminação.

Foi possível concluir também que é viável o uso da linguagem VRML para simulação de sistemas de iluminação aplicada a projetos de iluminação. No entanto, duas formas distintas de simulação podem ser geradas. A primeira, utilizando os nós de fontes de luz disponíveis em VRML, fornece uma visualização simplificada e qualitativa do efeito da iluminação causada. A segunda, utilizando técnicas de visualização científica para mostrar os resultados de uma simulação numérica, fornece uma visualização qualitativa e quantitativa do campo de iluminâncias obtido, podendo, portanto, ser utilizada efetivamente em projetos de engenharia.

Verificou-se que o VTK possibilita exportar conteúdo 3D para o formato VRML e, assim, essa ferramenta pode ser usada no desenvolvimento de aplicativos para visualização de resultados numéricos, utilizando interface em RV. O fato de se ter desenvolvido os aplicativos mostrados neste trabalho utilizando a ferramenta VTK, de código aberto, e a versão gratuita do *Microsoft Visual C++ Express Edition*, demonstra também que é possível construir ferramentas com aplicações efetivas em problemas de engenharia, a custos reduzidos, ao menos no que tange à aquisição de *softwares*.

Sobre as implementações realizadas, foi possível concluir que o algoritmo *ProcessaIES* possibilita uma forma simples e eficiente de se calcular a iluminação direta em malhas 2D ou 3D para qualquer projeto de iluminação. O surgimento de formatos de arquivos padrões para distribuição de informações fotométricas de fontes de luz torna o processo de cálculo relativamente simples de ser implementado computacionalmente. A saída do algoritmo, na forma de um arquivo *Legacy* do VTK, permite que os resultados possam ser visualizados em qualquer aplicativo ou algoritmo baseado no VTK. Algumas vantagens desse algoritmo são: a facilidade de uso, a possibilidade de utilizar qualquer fonte de luz, bastando possuir o arquivo IES correspondente, a possibilidade de escolher a densidade e dimensão da malha e a geração da saída como um arquivo do VTK. Uma desvantagem é a falta de uma interface interativa.

Em relação ao algoritmo *ColorMap,* pôde-se concluir que a utilização da ferramenta VTK possibilita um poderoso recurso para implementação de algoritmos de visualização e conversão entre diferentes formatos de forma a ser possível gerar o mapeamento de cores e converter a saída em uma textura (caso 2D) que pode ser visualizada em diferentes sistemas como CAD e de RV. Isso possibilita o desenvolvimento de aplicativos para visualização interativa de dados e também para processamento e pós-processamento. Uma vantagem desse algoritmo é a exportação da malha com mapeamento de cores para o formato VRML, possibilitando assim a sua visualização em ambientes virtuais 3D combinada com a visualização de modelos CAD.

A implementação do algoritmo *Isosup3D* permitiu concluir que sistemas de visualização de dados 3D constituem poderosas ferramentas de exploração do interior de amostras de dados volumétricos, e novas aplicações para esses sistemas, nos vários ramos da engenharia, tendem a surgir a cada dia. Em relação à simulação de iluminação, geralmente os dados de iluminância são calculados sobre um plano. A geração de isosuperfícies para visualizar os valores de iluminância horizontal em uma malha 3D (ou em um volume de dados) permite uma nova forma de observar a distribuição desses campos e, dessa forma, permite obter uma maior compreensão da

distribuição dos dados. Uma desvantagem desse algoritmo é não possuir uma interface interativa para ajuste dos parâmetros dos filtros envolvidos, tais como o número de superfícies geradas ou o valor para uma determinada superfície, que são descritos no código e são portanto fixos.

A integração desses algoritmos com o programa SolidWorks foi a alternativa adotada para se obter uma interface 3D intuitiva, em que o sistema de iluminação pudesse ser "montado" (ou projetado) interativamente, utilizando o ambiente de trabalho desse *software*. A implementação foi feita de forma que o usuário pudesse escolher uma fonte de luz em uma lista pré-definida, em uma base de dados, inseri-la no modelo 3D (de uma subestação, por exemplo), e escolher a direção de foco. Estas etapas são repetidas até que todas as luminárias estejam inseridas no projeto. Então o usuário escolhe uma face do modelo onde deseja visualizar o campo de iluminâncias. Como resultado desta implementação obteve-se um *Add-in* para o SolidWorks que realiza o processamento e o pós-processamento da iluminação direta em modelos 3D. Também foi possível, utilizando métodos da API (*Application Programming Interface*) do SolidWorks, implementar a interferência de corpos opacos na direção dos raios de luz. No entanto, o algoritmo desenvolvido até o momento requer um longo tempo de processamento funcionando apenas para pequenas partes da subestação. Mais testes são necessários para fazer uma avaliação mais detalhada.

Em relação à integração CAD e RV utilizando o CAD SolidWorks, chegou-se à conclusão de que uma forma viável de fazer a conversão entre os formatos é importar pequenas partes do modelo DWG (por exemplo, um transformador ou uma torre), previamente separadas em diferentes camadas (*layers*), ajustar o SolidWorks para representar as primitivas gráficas com baixa resolução (para fins de otimização), e exportar como um arquivo VRML (por exemplo, *transformador05.wrl* ou *poste07.wrl*). Assim, cria-se uma base de dados com geometrias de equipamentos e objetos que compõem a subestação. Para reconstruir o modelo em VRML, basta carregar os arquivos (geometrias de equipamentos e objetos) desejados em um único ambiente por meio de um "arquivo principal" que invoca os outros arquivos usando o nó *Inline* do VRML.

Vale citar que, neste trabalho, usaram-se técnicas de visualização de campos escalares de iluminâncias. No entanto, os mesmos algoritmos podem ser usados para visualizar campos escalares de intensidade de campos elétricos e magnéticos. Embora campos eletromagnéticos sejam campos vetoriais, se considerada apenas a sua intensidade e não a direção do campo em cada ponto, o caso passa a ser de visualização de campos escalares. A ferramenta VTK, utilizada no desenvolvimento deste trabalho, também possui classes que implementam técnicas de visualização de campos vetoriais e tensoriais. Trabalhos futuros podem explorar o desenvolvimento de aplicações para visualização de campos eletromagnéticos por meio do uso de técnicas de visualização de campos vetoriais.

9.1 SUGESTÕES PARA TRABALHOS FUTUROS

Trabalhos futuros podem concentrar-se no desenvolvimento de interfaces para sistemas interativos de visualização que permitam, no ambiente virtual, extrair planos de corte e gerar isosuperfícies, por exemplo, oferecendo uma forma de ajustar diversos parâmetros definidos nos algoritmos de visualização de forma interativa e em tempo real. Também pode ser explorada a utilização de dispositivos especiais de entrada e saída de dados, como luvas de dados (*datagloves*).

Outra sugestão para trabalhos futuros está relacionada ao desempenho do aplicativo e tamanho dos modelos utilizados. Melhorar o desempenho do algoritmo de análise de interferência por corpos opacos ou avaliar o melhor algoritmo dentre os existentes torna-se relevante para permitir o uso operacional de aplicativos como o desenvolvido neste trabalho. Também é relevante encontrar formas de converter modelos CAD em modelos de RV utilizando-se da melhor forma possível das técnicas de otimização existentes, ou então, desenvolvendo novas técnicas.

Em uma perspectiva mais otimista, pode ser sugerido o desenvolvimento de sistemas interativos com interface em RV, em que o processamento e a visualização de dados seriam realizados em tempo real. Esse tipo de sistema requer o estudo de uma arquitetura mais sofisticada, pois envolve um alto custo computacional, tanto em

processamento numérico quanto gráfico. Pode-se citar, como exemplo relacionado ao trabalho apresentado nesta dissertação, uma funcionalidade do sistema em que o usuário pudesse, no ambiente virtual, mover uma fonte de luz e observar em tempo real as mudanças geradas no campo de iluminâncias. O mesmo pode ser pensado para análise de outras grandezas físicas, como temperaturas ou campos eletromagnéticos, também pode ser investigada a viabilidade de utilizar processamento pelo MEF.

REFERÊNCIAS

ABS-TECH. Absolute Technologies Home Page. Disponível em: <http://www.abs-tech.com/>. Acesso em: 21 jan. 2006.

ADAIME, L. M. **Aplicação do *Visualization Toolkit* para pós-processamento de análises pelo método dos elementos**. 141 p. Dissertação (Mestrado em Métodos Numéricos em Engenharia) Universidade Federal do Paraná, Curitiba, 2005.

AMIRA. Amira Home Page. Disponível em: <http://www.amiravis.com/>. Acesso em: 5 fev. 2006.

ANSYS. Ansys Home Page. Disponível em: <http://www.ansys.com>. Acesso em: 11 fev. 2006.

ARROYO, E.; LOS ARCOS, J. L. Los. A Virtual Reality Application to Electrical Substations Operation Training. **IEEE International Conference on Multimedia Computing and Systems**, v. 1, n. 1, p. 835-839, 1999.

AVS. AVS Home Page. Disponível em: <http://www.avs.com>. Acesso em: 5 fev. 2006.

AZEVEDO, E.; CONCI, A. **Computação Gráfica:** teoria e prática. Rio de Janeiro: Elsevier, 2003.

BATTAIOLA, A. L.; SOARES, L. P. Estudo e Uso Exploratório de Ferramentas de Visualização Científica. In: SEMANA DE INFORMÁTICA DA UFBA, 7., 1998, Salvador. **Anais...** Salvador: UFBA, 1998. p. 16-30.

BERTA, J. Integrating VR and CAD. **IEEE Computer Graphics and Applications**, v. 19, n. 5, p. 14-19, 1999.

BHALERAO, A.; WESTIN, C. Tensor Splats: Visualising Tensor Fields by Texture Mapped Volume Rendering. In: CONFERENCE ON MEDICAL IMAGE COMPUTING AND COMPUTER-ASSISTED INTERVENTION - MICCAI'03, 6., 2003, Montreal, **Anais...** Berlin: Springer / Heidelberg , 2003. p. 294-302.

BLYTHE (1999). Advanced Graphics Programming Techniques Using OpenGL. Tutorial apresentado no SIGGRAPH 1999. Disponível em : <http://www.opengl.org/resources/tutorials/sig99/advanced99/notes/node283.html>. Acesso em: 20 dez. 2005.

BRYSON, S. Virual environments in scientific visualization. In: EARNSHAW, R. A.; WATSON, D. (Eds.) **Animation and Scientific Visualization**: tools & applications. Academic Press, 1993. p. 113-122.

BRODLIE, K. A classification scheme for scientific visualization. In: EARNSHAW, R. A.; WATSON, D. (Eds.) **Animation and Scientific Visualization**: tools & applications. Academic Press, 1993. p. 125-140.

BRODLIE, K. et al. **Scientific Visualization, techniques and applications**. Springer-Verlag, 1992.

BURDEA, G.; COIFFET, P. **Virtual Reality Technology**. New Jersey: Wiley-Interscience, 2003.

BURIOL, T. et al. Realidade Virtual Aplicada a Visualização de Campos de Iluminância e Integração com CAD 3D, In: SYMPOSIUM ON VIRTUAL REALITY, 8., 2006, Belém. **Anais...** Belém: SBC, 2006, submetido.

BURKHARD. Página do pesquisador Burkhard Wünsche, universidade de Auckland. Disponível em: <http://www.cs.auckland.ac.nz/~burkhard/PhD/introduction.html>. Acesso em: 9 fev. 2006

CABRAL, B. Imaging vector fields using line integral convolution. In: International Conference on Computer Graphics and Interactive Techniques, 20., Florida. **Anais...**, New York: ACM Press, 1993 p. 263-270.

CARDOSO, A. VRML: a web em 3D. In: CARDOSO, A.; TEIXEIRA, C. A. C.; LAMOUNIER Jr., E. (org.). **Ambientes Virtuais :** projeto e implementação. 01. ed. Porto Alegre, 2003. p. 01.

COLLINS, B. M. Data visualization: has it all been seen before? In: EARNSHAW, R. A.; WATSON, D. (Eds.) **Animation and Scientific Visualization**: tools & applications. Academic Press, 1993. p. 3-28.

CORSEUIL, E. T. L. et al. A VR tool for the visualization of CAD models. In: SYMPOSIUM ON VIRTUAL REALITY, 7., 2004, São Paulo. **Anais...** Porto Alegre: SBC, 2004. p.327-338.

CORSEUIL, E. T. L. et al. Buscando o uso operacional de realidade virtual em grandes modelos de engenharia. In: SYMPOSIUM ON VIRTUAL REALITY, 6., 2003, Ribeirão Preto. **Anais...** Porto Alegre: SBC, 2003. p.187-198.

CRAWFIS, R.; MAX, N; BECKER, B. Vector field visualization. **IEEE Computer Graphics and Applications**, v. 14, n. 5, p. 50-56, 1994.

CRAWFIS, R.; MAX, N. Direct volume visualization of three-dimensional vector field. In: WORKSHOP ON VOLUME VISUALIZATION, 1992, San Diego. **Anais...** New York: ACM Press, 1992. p. 55-60.

DELMARCELE, T.; HESSELINK, L. Visualizing second-order tensor fields with hyperstreamlines. **IEEE Computer Graphics and Applications**, v. 13, n. 4, p. 25-33, 1993.

ENCARNAÇÃO, J. L. Advanced research and development topics in animation and scientific visualization. In: EARNSHAW, R. A.; WATSON, D. (Eds.) **Animation and Scientific Visualization:** tools & applications. Academic Press, 1993. p. 37-73.

ESPINHA, R. S. L. **Visualização interativa de malhas não estruturadas utilizando placas gráficas programáveis**. 86 p. Dissertação (Mestrado em Informatica) Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2005.

ESS, E.; SUN, Y. Visualizing 3D vector fields with splatted streamlines. In: IS&T AND SPIE SYMPOSIUM, Visualization and Data Analysis, 18., 2006, San Jose. **Anais...** San Jose: IS&T/SPIE, 2006. Disponível em: <http://www.cs.purdue.edu/homes/sun/CIRL/Publication/Ess06_SPIE_VDA__vectorVis.pdf>. Acesso em: 10 fev. 2006.

FOLEY, J. et al. **Computer Graphics:** principles and practice. Addison-Wesley, 1990.

FURNAS. Revista Furnas, n. 326, Novembro de 2005. Disponível em: <http://www.furnas.com.br/noticias_ConsRevistaFurnas.asp?codigo=42&mes=11&ano=2005>. Acesso em: 14 dez. 2005.

FURNAS. Home page de Furnas Centrais Elétricas. Disponível em: <http://www.furas.com.br/inovacao_ped_0203_42.asp>. Acesso em: 7 fev. 2006.

GARCIA, F. L. S. Visualização de problemas eletromagnéticos 3D baseados em MEF utilizando VRML. In: SYMPOSIUM ON VIRTUAL REALITY, 4., 2001, Florianópolis. **Anais...** Florianópolis : UFSC, 2001. p. 377-378.

GEUS, K.; DOMETERCO, J. H. Visualização qualitativa em engenharia utilizando Realidade Virtual. **Revista Espaço Energia – COPEL**, Curitiba, v1, n.1, 2004.

GLOBUS, A.; RAIBLE, E. Fourteen ways to say nothing with scientific visualization. **IEEE Computer Graphics and Apllications**, v. 27, n. 7, p. 86-88, 1994

GROEN. Homepage of Pieter. W. Groen.
Disponível em: <http://www.nat.vu.nl/~pwgroen/scivis/tax.html>. Acessado em: 6 fev. 2006.

GNUPLOT. GnuPlot Home Page.
Disponível em: <http://www.gnuplot.info>. Acessado em: 5 fev. 2006

HELGELAND, A; ANDREASSEN, O. Visualization of vector fields using Seed LIC and volume rendering. **IEEE Transaction on Visualization and Computer Graphics**. v. 10, n. 6, p. 673-682, 2004.

HELWIG, L. D. Using web-based immersive graphical interfaces to access power substation design and standards information. DistribuTECH, Miami, 2002. Disponível em:<http://axis1.bizland.com/papers/TD2002_Paper.pdf>. Acesso em: 20 ago. 2005.

HESSELINK, L.; POST, F.H.; van WIJK, J.J.. Research issues in vector and tensor field visualization. **IEEE Computer Graphics and Applications**, v. 14, n. 12, p.76-79, 1994.

HUANG, M. et al. Virtual Reality visualization of 3D electromagnetic fields. Argonne National Laboratory, Argonne, 1996. Disponível em: <http://citeseer.ist.psu.edu/111951.html>. Acesso em: 21 fev. 2006.

HULTQUIST, J. P. M. Constructing stream surfaces in steady 3D vector fields. In: CONFERENCE ON VISUALIZATION, 3., 1992, Boston. **Anais...** Boston: IEEE CS Press, 1992, p. 171-178.

JOHN, N. W., LENG, J. Scientific applications of visualization, Virtual Reality and high performance visualisation computers. **UKHEC – UK High Performance Computing**, Manchester, p. 1-9, 2001.

KAGEYAMA, A., OHNO, N. **Tutorial Introduction to Virtual Reality**: What Possibility are offered to our field? Cornell University Library, Kyoto, v.1, n.1, p.127-136, 2005. Disponível em: <http://arxiv.org/abs/physics/0512066> Acesso em: 5 jan.2005.

KAUFMAN, A. E. **Volume Visualization**. Los Alamitos: IEEE CS Press, 1991.

KERLICK, G. D. Moving iconic objects in scientific visualization. In: IEEE CONFERENCE ON VISUALIZATION, 1., 1990. **Anais...** IEEE CS Press, 1990. p. 124-130.

KINDLMANN, G. Superquadric tensor glyphs. In: IEEE TVCG/EG SYMPOSIUM ON VISUALIZATION, 2004. **Anais...** IEEE TCVG Press, 2004. p. 147-154.

KITWARE. **The VTK user's guide**: install, use and extend The Visualization Toolkit. 2004.

KITWARE. VTK home page. Disponível em: <http://www.vtk.org>. Acesso em: 19 jan. 2005.

KLOCKE, F.; STRAUBE, A. M. Virtual process engineering: an approach to integrate VR, FEM, and simulation tools in the manufacturing chain. **Mécanique & Industries**, Courtaboeuf, n. 5, p. 199-205, 2004.

KRÜGER, J. et al. A particle system for interactive visualization of 3D flows. **IEEE Transactions on Visualization and Computer Graphics**, v. 11, n. 6, p. 744-756, 2005.

LACROUTE, P.; LEVOY, M. Fast volume rendering using a Shear-Warp factorization of the viewing transformation. In: SIGGRAPH'94, Orlando, 1994. **Anais...** Orlando: ACM Press, 1994. p. 451-458.

LAIDLAW D. H. et al. Comparing 2D vector field visualization methods: a user study. **IEEE Transactions on Visualization and Computer Graphics**, v. 11, n. 1, p. 59-70, 2005.

LEE, C. S.; BURDEA G. C. **Virtual Reality technology laboratory manual**. Rutgers-The State University of New Jersey, 2nd ed, U.S.A, John Wiley & Sons, 2003.

LENG, J. Scientific examples of Virtual Reality and visualization applications. **UKHEC – UK High Performance Computing**, Manchester, p. 1-13, 2001.

LEVOY M. Volume Rendering - Display of surfaces from volume data. **IEEE. Computer Graphics & Applications**, v. 8, n. 3, p. 29-37, 1988.

LEVOY M. Efficient Ray-Tracing of volume data. **ACM Transactions on Graphics**, v. 9, n. 3, p. 245-261, 1990.

LI. Jing Li Home Page. Disponível em: <http://www.cs.auckland.ac.nz/~jli023/>. Acessado em: 8 dez. 2005.

LU, L.; CONNEL, M.; TULLBERG, O. The use of Virtual Reality in interactive Finite Element Analysis by Java 3D API. In: CONFERENCE AT CHALMERS, Gothenburg. **Anais...** Gothenburg, 2001.

MANSSOUR, I. H., FREITAS, C. M. D. S. Visualização Volumétrica. **Revista de Informatica Teórica e Aplicada**, Porto Alegre, v. 9, n. 2, p. 97-126, 2002.

MARINOSKI, D. L.; WESTPHAL, F. S.; LAMBERTS, R. Desenvolvimento de um algoritmo de cálculo luminotécnico para ambientes internos através do método ponto-a-ponto. In: ENCONTRO NACIONAL SOBRE CONFORTO NO AMBIENTE CONSTRUÍDO, 7., 2003, Curitiba. **Anais...** Curitiba: ENCAC-COTEDI, 2003. p. 1066-1073.

MARINOVA, I.; ENDO, H.; SAITO, Y. Electromagnetic field visualization by image processing. In: INTERNATIONAL SYMPOSIUM ON ELECTRICAL APPARATUS AND TECHNOLOGIES, 7., 2001, Plovdiv. **Anais…** Plovdiv: SIELA, 2001. p. 1-6.

MCCORMICK, B. H.; DEFANTI, T. A.; BROWN, M. D. Visualization in Scientific Computing. **Computer Graphics** (edição especial), v. 21, n. 6, 1987.

MICROSOFT. *Site* do Visual C++ 8.0 2005 *Express Edition*.
Disponível em: < http://msdn.microsoft.com/vstudio/express/visualc/default.aspx>.
Acessado em: 4 jan. 2006.

MIRANDA, M. F. Interface 3D aplicada à visualização de informações de uma subestação de energia armazenadas em banco de dados. In: WORKSHOP DE APLICAÇÕES DE REALIDADE VIRTUAL, 1., 2005, Uberaba. **Anais...** Uberaba: UFU, 2005.

MOREIRA, V. de A. Método preciso para cálculo de iluminação por computador. In: SEMINÁRIO NACIONAL DE DISTRIBUIÇÃO DE ENERGIA ELÉTRICA, 10., 1988, Rio de Janeiro. **Anais...** Rio de Janeiro: Eletrobrás, 1988.

MOREIRA, V. de A. **Iluminação elétrica**. EdgarBlücher Ltda, São Paulo, 1999.

MTE. Ministério do Trabalho e do Emprego. Segurança e Saúde no Trabalho. Manual do Setor Elétrico e Telefonia. Disponível em:
<www.mte.gov.br/Empregador/segsau/Conteudo/969.pdf >. Acesso em: 1° dez. 2005.

NETTO, A. V.; MACHADO, L. S.; OLIVEIRA, M. C. F. **Realidade Virtual:** fundamentos e aplicações. Visual Books, 2002.

NETTO, A. V., GOUVEIA, J. D., CATERIANO P. S. H. Interface 3D para manipulação de dados em redes de distribuição de energia elétrica. **INFOCOMP Journal of Computer Science**, Lavras, v. 4, n. 4, p. 73-81, 2005

ORTENG. Disponível em: <http://www.ortengrj.com.br/>. Acesso em: 21 jan. 2006.

PAVÃO, A. C., POUZADA, E. V. S., MATHIAS M. A. Electromagnetic field visualization through VTK software. In: MICROWAVE AND OPTOELECTRONICS CONFERENCE, 2001, Belém. **Anais...** Belém: SBMO/IEEE MTT-S, 2001. p. 21-24.

RAPOSO, A. B. et al. Visão Estereoscópica, Realidade Virtual, Realidade Aumentada e Colaboração. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 24., 2004, Salvador. **Anais...** Porto Alegre: SBC, 2004. p. 289 - 331.

ROQUEIRO. Página eletrônica do professor Nestor Roqueiro da UFSC.
Disponível em: <http://www.das.ufsc.br/~nestor/cursos/matlab/matlab/matlab.html>.
Acessado em: 5 jan. 2006.

RUSSO, E. E. R. et al. A Realidade Virtual na Indústria de exploração e produção de petróleo. In: Realidade Virtual: Conceitos e Tendências – Livro do Pré-Simpósio SVR 2004, São Paulo: Editora Mania de Livro, 2004. p.283-288.

SCHIMIDT, A. E. F. **Visualização tridimensional combinada de dados volumétricos e modelos poligonais usando o algoritmo *Shear-Warp***. 121 p. Tese

(Doutorado em Informática), Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2000.

SCHOEDER, W.; VOLPE, C. R.; LORENSEN, W. E. T. The stream polygon: a technique for 3D vector field visualization. In: IEEE CONFERENCE ON VISUALIZATION, 2., 1991, San Diego. **Anais...** San Diego: IEEE, 1991. p. 126-132.

SCHROEDER, W.; MARTIN, K.; LORENSEN, B. **The Visualization Toolkit:** an object-oriented approach to 3D graphics. 3rd. ed. Kitware, Inc, 2004.

SCHROEDER W. J.; AVILA, L. S.; HOFFMAN, W. Visualizing with VTK: a tutorial. **IEEE Computer Graphics and Applications**, v. 20, n. 5, p. 20-27, 2000.

SHEN, H. W.; LI G. S.; BORDOLOI, U. D. Interactive Visualization of Three-Dimensional Vector Fields with Flexible Appearance Control. **IEEE Transactions On Visualization And Computer Graphics**, v. 10, n. 4, p. 434-445, 2004.

SHERMAN, W. R. e CRAIG, A. B. **Understanding Virtual Reality**: Interface, Application, and Design. Morgan Kaufmann, 2003.

SIGFRIDSSON, A. et al. Tensor field visualization using adaptative filtering of noise field combined with glyph rendering. In: IEEE VISUALIZATION, 13., 2002, Los Alamitos. **Anais...** Los Alamitos: IEEE, 2002. p.461-469.

SISCOUTTO, R. A. et al. Estereoscopia. In: Realidade Virtual: Conceitos e Tendências - Livro do Pré-Simpósio SVR 2004, Editora Mania de Livro, 2004. p.179-201.

SINDIPETRO. Sindicato dos Trabalhadores da indústria de petróleo. Cartilha: Meio Ambiente, Saúde e Trabalho. Disponível em: <www.mte.gov.br/Empregador/segsau/Conteudo/969.pdf>. Acesso em: 30 nov. 2005.

SOUZA, C. A. S. de. **Implementação de uma estrutura de dados para visualização científica**. 92p. Mestrado (Ciências da Computação e Matemática Computacional) Universidade de São Paulo, São Carlos, 2003.

SOUZA, A. L. et al. Uma biblioteca VRML para a visualização de campos eletromagnéticos. In: ENCONTRO DE ENSINO DE ENGENHARIA, 5., 2000, Itaipava. **Anais...** Rio de Janeiro: Eletrobrás, 2000.

TAVARES, J.; BARBOSA, J. Notas da disciplina de Visualização Científica 2004-2005, Faculdade de Engenharia da Universidade do Porto. Disonível em: <http://paginas.fe.up.pt/~tavares/ensino/VISCI/visci.html>. Acesso em:30 dez. 2005.

UDUPA, J. K., ODHNER, D. Shell Rendering. **IEEE Computer Graphics & Applications**, v. 13, n. 6, p. 58-67, 1993.

UPSON, C.; KEELER, M. V-Buffer: Visible Volume Rendering. **SIGGRAPH Computer Graphics**, New York, v. 22, n. 4, p. 59-64, 1988.

UPSON, C. et al. The Application Visualization System: a computational environment for Scientific Visualization. **IEEE Computer Graphics & Applications**, v. 9, n. 4, p. 30-42, 1989.

VELHO, L.; GOMES, J. **Sistemas gráficos 3D**. São Paulo: IMPA, 2001

VISAD. VisAD Home Page. Disponível em: <http://www.ssec.wisc.edu/~billh/visad.html>. Acesso em: 5 jan. 2006.

VRML20. VRML 2.0 Especification. Disponível em: <http://www.graphcomp.com/info/specs/sgi/vrml/spec/part1/nodesRef.html>. Acesso em: 20 mar. 2006.

VTK. The Visualization Toolkit. Disponível em:< http://www.vtk.org/index.php>. Acesso em: 03 jan. 2006.

WALTON, J. Get the picture: a new direction in data visualization. In: EARNSHAW, R. A.; WATSON, D. (Eds.) **Animation and Scientific Visualization**: tools & applications. Academic Press, 1993. p. 29-36.

WATT, A. **3D Computer Graphics**, 3th. ed., Addison Wesley, 2000.

WEB3D. Open Standards for Real-Time 3D Communication. Disponível em: <http://www.web3d.org/ >. Acesso em: 30 jan. 2006.

WEINSTEIN, D. M.; KINDLMANN, G. L.; LUNDBERG, E. C. Tensorlines: Advection-Diffusion based Propagation through Diffusion Tensor Fields. In: IEEE VISUALIZATION, 10., 1999, Los Alamos. **Anais...** Los Alamos: IEEE**,** 1999. p. 249-254.

WESTIN, C. F. Processing and visualization for diffusion tensor MRI. **Medical Image Analysis**, v. 6, p. 93-108, 2002.

WESTOVER, L. Interactive Volume Rendering. In: WORKSHOP ON VOLUME VISUALIZATION, 1989, North Carolina. **Anais...** University of North Carolina Press, 1989. p. 9-16.

WESTOVER, L. Footprint Evaluation for Volume Rendering. In: INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 17., 1990, Dallas. **Anais...** New York: ACM Press, 1990. p. 367-376.

WIJK, J. J. V. Spot noise-texture synthesis for data visualization. In: INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 18., 1991, Richland. **Anais...** New York: ACM Press,1991. p. 309-318.

WIKIPEDIA . The Free Encyclopedia.
Disponível em: <http://en.wikipedia.org/wiki/Scientific_visualization>
Acessado em: 6 jan. 2006

WILHELMS, J.; GELDER, A. V. A coherent projection approach for direct volume rendering. **ACM SIGGRAPH Computer Graphics**, New York, v. 25, n. 4, p. 275-284, 1991.

WOLFF, R. S.; YAEGER, L. **Visualization of Natural Phenomena**. Springer. Bk&CD-Rom, 1993.

ZHANG, S.; DEMIRALP, C.; LAIDLAW, D. H. Visualizing diffusion tensor MR images using streamtubes and streamsurfaces. **IEEE Visualization and Computer Graphics**, v. 9, n. 4, p. 454-462, 2003.

ZONG, Y., MUELLER, W. e MA, W. A model representation for solid modeling in a virtual reality environment. In: PROCEEDINGS OF THE SHAPES MODELING INTERNATIONAL, **Anais...**, IEEE, 2002. p.183-190

APÊNDICES

APÊNDICE 1 – ALGORITMO PONTO-A-PONTO

```
!!!!!!!!!!!!!!! ProcessaIES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Programa que lê um arquivo de entrada, devidamete formatado,
! processa os cálculos de iluminação direta, e gera um arquivo
! de saída no formato Legacy do VTK
! FORTRAN 6
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

real*4, allocatable, dimension(:,:)::LUZ,AngH,AngV,EE_IES,LP,LC,Dist,Gama
real*4, allocatable, dimension(:)::IES,alt,larg,prof,Lm,mult,EE,Dist2,
real*4, allocatable, dimension(:)::EEE,maxH,maxV,indV,indH
real*4 dimX,dimY,dimZ,divX,divY,divZ,OX,OY,OZ,OOX,OOY,OOZ,
real*4 deltaX,deltaY,deltaZ,
real*4 mod_w,mod_d,mod_h,cos_fi,dXp,hXw,
real*4 cos_alfaIES,cos_betaIES,alfaIES,betaIES,PI,maxwv
integer, allocatable, dimension(:) ::NL_IES,TipFot,Unid,NumAngH,NumAngV
integer i,j,k,kk,NIES,NL,NP,NV,NC,n, a,b,c,d,e,f,g,h,yy,zz, NPX,NPY,NPZ
character (150), allocatable, dimension(:) :: arqIES
character (150) frase, cab, input, plano
1 format (20(F15.4), 1/) !formato: (Coluna(FloatEspaços.CasasDec)
2 format ("8",8(I12))
3 format ("# vtk DataFile Version 3.0")
4 format (12(F8.2), 1/)
5 format ("#VRML V2.0 utf8")
6 format ("4",8(I12))
PI=3.14159265358979323846264338327 95
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
write(*,*) "ProcessaIES.exe_STRUCTURED_POINTS"
write(*,*) ""
write(*,*) ""
write(*,*) "Entre com o nome do arquivo de entrada: "
write(*,*) ""
read(*,*) input
!!!!!!!!!!!!!!!LEITURA DO ARQUIVO: "entrada.txt"!!!!!!!!!!!!!!!!!!!!!!!!!!!!
open (2, file=input)
read(2,*)dimX, dimY, dimZ
read(2,*)divX, divY, divZ
read(2,*)OX, OY, OZ
read(2,*)NIES,NL
allocate (LUZ(NL,7))
allocate (arqIES(NIES))
 do i=1,NIES,1
  read (2,*) arqIES(i)
 end do
read (2,*) ((LUZ(i,j),j=1,7),i=1,NL)
close(2)
!!!!!!!!!!!!!!!!PROCESSA INFORMAÇÕES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
NV = divX*divY*divZ
deltaX=dimX/divX
deltaY=dimY/divY !espaçamento entre nós no eixo Y
deltaZ=dimZ/divZ !espaçamento entre nós no eixo Z
NPX=divX+1
NPY=divY+1
NPZ=divZ+1
if(dimX==0)NPX=1
if(dimY==0)NPY=1
if(dimZ==0)NPZ=1
NP=NPX*NPY*NPZ
allocate (LP(NP,3)) !LP=lista de pontos: matriz de NP por 3 elementos
```

```
allocate (LC(NV,3)) !LC=lista de células: NVx3 (cada voxel e seus vértices)
allocate (NL_IES(NIES))
allocate (Lm(NIES))
allocate (mult(NIES))
allocate (NumAngH(NIES))
allocate (NumAngV(NIES))
allocate (TipFot(NIES))
allocate (Unid(NIES))
allocate (alt(NIES))
allocate (larg(NIES))
allocate (prof(NIES))
allocate (Dist(NP,NL))
allocate (Gama(NP,NL))
allocate (EE(NP))
allocate (EEE(NP))
allocate (Dist2(NL))
allocate (indH(NIES))
allocate (indV(NIES))
allocate (maxH(NIES))
allocate (maxV(NIES))
!!!!!!!!!!!!!!!!!ALOCA E ZERA AS MATRIZES!!!!!!!!!!!!!
allocate (AngH(NIES,100))
allocate (AngV(NIES,100))
allocate (EE_IES(NIES,10000))
do j=1,NIES,1
do i=1,100,1
AngH(j,i)=0
AngV(j,i)=0
end do
do i=1,10000,1
EE_IES(j,i)=0
end do
end do
!!!!!!!!!!!!!!!!!!LÊ OS ARQUIVOS IES!!!!!!!!!!!!!!!!!
do k=1,NIES,1
open (4,file=arqIES(k)) !Lê IES->k=indice referente ao ies

do i=1,50,1
read(4,*)cab
if (cab=="TILT=NONE") exit
end do

read(4,*)NL_IES(k),Lm(k),mult(k),NumAngV(k),NumAngH(k),TipFot(k),Unid(k),alt(k),lar
g(k),prof(k)
read(4,*)
read(4,*)(AngV(k,i),i=1,NumAngV(k)) !k é o indice do IES, e i do angulo
read(4,*)(AngH(k,i),i=1,NumAngH(k))
read(4,*)(EE_IES(k,i),i=1,(NumAngH(k)*NumAngV(k)))

do i=1,(NumAngH(k)*NumAngV(k)),1
EE_IES(k,i)=EE_IES(k,i)*mult(k)
end do

!obtenção do maior ângulo horizontal e vertical
do j=2,NumAngH(k),1
if (AngH(k,j)>AngH(k,j-1)) maxH(k)=AngH(k,j)
indH(k)=j
end do
```

```
do j=2,NumAngV(k),1
if (AngV(k,j)>AngV(k,j-1)) maxV(k)=AngV(k,j)
indV(k)=j
end do

!!cria os valores dos angulos horizontais, omitidos devido à simetria
if(maxH(k)==180) then
  do j=1,indH(k)-1,1
  AngH(k,indH+j)=360-AngH(k,indH-j)
   do i=1,indV(k),1
   EE_IES(k,((indH(k)-1+j)*indV)+i)=EE_IES(k,((indH(k)-1-j)*indV)+i)
   end do
  end do
 NumAngH(k)=(2*NumAngH(k))-1
 maxH(k)=360
 indH(k)=NumAngH(k)
end if
close(4)
end do

!!!!!!!!!!!!!!!!!!!!INICIA GERAÇÃO DO ARQUIVO.VTK!!!!!!!!!!!!!!!!!!
open(1, FILE="arquivo.vtk")!ABRE O ARQUIVO1 PARA COMEÇAR A GERAÇÃO
        !CABEÇÁRIO
       write(1,3)
       write(1,*)"Pontos Estruturados"
       write(1,*)"ASCII"
              !ESCREVE GEOMETRIA/TOPOLOGIA
       write(1,*)"DATASET STRUCTURED_POINTS"
       write(1,*)"DIMENSIONS",NPX, NPY, NPZ
       write(1,*)"ORIGIN",OX,OY,OZ
       write(1,*)"SPACING",deltaX,deltaY,deltaZ

       n=0
       do k=0,NPZ-1,1
       do j=0,NPY-1,1
       do i=0,NPX-1,1
       n=n+1
       LP(n,1)= i*deltaX+OX !INDEXAÇÃO DOS NÓS DA MALHA: coord X
       LP(n,2)= j*deltaY+OY !INDEXAÇÃO DOS NÓS DA MALHA: coord Y
       LP(n,3)= k*deltaZ+OZ !INDEXAÇÃO DOS NÓS DA MALHA: coord Z
    !write(*,*) LP(n,1),LP(n,2), LP(n,3)
       end do
       end do
       end do

!!!!!!!!!!!!!!!!!LISTA DE ESCALARES - PROCESSAMENTO OU CÁLCULO NUMÉRICO!!!!!!!!
write(1,*)""
write(1,*)"POINT_DATA", NP
write(1,*)"SCALARS scalars float"
write(1,*)"LOOKUP_TABLE default"
do i=1,NP,1 !!!!!!!!!vai de ponto em ponto, começando pelo primeiro
    EE(i)=0 !zera a iluminância no ponto
       EEE(i)=0
       do j=1,NL,1 !!!!!!!!!vai de luz em luz
!cálculo da distância ponto(i)-luz(j)
Dist(i,j)= sqrt(((LP(i,1)-LUZ(j,1))**2)+((LP(i,2)-LUZ(j,2))**2)+((LP(i,3)-
LUZ(j,3))**2))
```

```
if (Dist(i,j)<1) Dist(i,j)=1
!cálculo da distância entre o ponto de luz e o ponto de foco
Dist2(j)=sqrt(((LUZ(j,1)-LUZ(j,4))**2)+((LUZ(j,2)-LUZ(j,5))**2)+((LUZ(j,3)-
LUZ(j,6))**2))
if (Dist(i,j)<1) Dist(i,j)=1
Gama(i,j)= acos(abs((LP(i,3)-Luz(j,3))/ Dist(i,j)))
sigma=(((LP(i,1)-LUZ(j,1))*(LUZ(j,4)-LUZ(j,1)))+((LP(i,2)-LUZ(j,2))*(LUZ(j,5)-
LUZ(j,2)))+((LP(i,3)-LUZ(j,3))*(LUZ(j,6)-LUZ(j,3))))/(Dist2(j)**2)
       OOX=LUZ(j,1)+(sigma*(LUZ(j,4)-LUZ(j,1)))
       OOY=LUZ(j,2)+(sigma*(LUZ(j,5)-LUZ(j,2)))
       OOZ=LUZ(j,3)+(sigma*(LUZ(j,6)-LUZ(j,3)))
mod_h=sqrt(((LUZ(j,2)-LUZ(j,5))**2)+((LUZ(j,4)-LUZ(j,1))**2))
mod_w=sqrt(((LP(i,1)-OOX)**2)+((LP(i,2)-OOY)**2)+((LP(i,3)-OOZ)**2))
dXp=((LUZ(j,4)-LUZ(j,1))*(LP(i,1)-LUZ(j,1)))+((LUZ(j,5)-LUZ(j,2))*(LP(i,2)-
LUZ(j,2)))+((LUZ(j,6)-LUZ(j,3))*(LP(i,3)-LUZ(j,3)))
hXw=((LUZ(j,2)-LUZ(j,5))*(LP(i,1)-OOX))+((LUZ(j,4)-LUZ(j,1))*(LP(i,2)-OOY))
cos_alfaIES=(dXp/(Dist(i,j)*Dist2(j)))
if (cos_alfaIES==0) cos_alfaIES=.000000001
if (cos_alfaIES>=1) cos_alfaIES=.999999999
if (cos_alfaIES<=-1) cos_alfaIES=-.999999999
alfaIES=acos(cos_alfaIES)*(180/PI)
cos_betaIES=(hXw/(mod_h*mod_w))
if (mod_h*mod_w==0) cos_betaIES=.000000001
if (cos_betaIES==0) cos_betaIES=.000000001
if (cos_betaIES>=1) cos_betaIES=0.999999999
if (cos_betaIES<=-1) cos_betaIES=-0.999999999
betaIES=acos(cos_betaIES)*(180/PI)
if ((LP(i,3)-OOZ)<=0) betaIES=360-betaIES
betaIES=360-betaIES
ind_V=1
ind_H=1
do k=1,NumAngV(LUZ(j,7)),1 !consulta aos vetores do arquivo IES de cada luz
if (alfaIES<=AngV(LUZ(j,7),k)) exit
ind_V=k
end do
do k=1,NumAngH(LUZ(j,7)),1 !consulta aos vetores do arquivo IES
if (betaIES<=AngH(LUZ(j,7),k)) exit
ind_H=k
end do
EEE(i)=EE_IES(LUZ(j,7),((ind_H-1)*NumAngV(LUZ(j,7))+ind_V ))
EE(i)=abs((EEE(i)*cos(Gama(i,j)))/(Dist(i,j)**2)+EE(i))
end do
write(1,*)EE(i)
end do
close(1)
end
```

APÊNDICE 2 – INSTALAÇÃO DO VTK

INSTALAÇÃO DO VTK NO WINDOWS XP

Existem duas maneiras de instalar o VTK 4.2: usando o instalador das bibliotecas pré-compiladas e compilando os códigos fonte. Tanto o programa instalador (*vtk42-LatestRelease.exe*), quanto os códigos fonte (*VTK-4.2-LatestRelease.zip*), estão disponíveis para *download* no endereço eletrônico *www.vtk.org/get-software.php* (VTK, 2006).

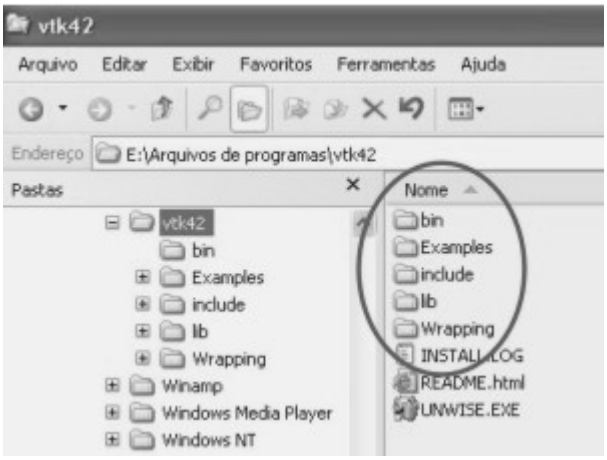
A instalação usando as bibliotecas pré-compiladas é mais rápida e mais fácil, no entanto, estas, são disponibilizadas apenas para a plataforma Win32. Além disso, foram construídas com o VC6, que usa velhos padrões "não-ANSI" de arquivos *header* (como o "*iostream.h*"), dessa forma, para construir soluções VTK com o compilador VS .Net 2003, é necessário primeiramente compilar o código fonte do VTK usando o VS .Net 2003 (LI, 2006).

Maiores detalhes sobre a compilação e instalação do VTK no Windows XP e a construção de projetos e soluções usando o VC 6.0 e o VS .Net 2003, podem ser encontrados em (LI, 2006). A seguir são descritos detalhadamente os passos para instalar o VTK 4.2 no Windows XP e para utilizá-lo com o compilador VC8 *Express Edition*.

1 INSTALAÇÃO DO VTK 4.2 PARA WIN32

A maneira mais simples de instalar o VTK 4.2 no Windows XP é através da execução do programa *vtk42-LatestRelease.exe* disponível para *download* em (VTK, 2006). Este programa instala as bibliotecas pré-compiladas do VTK no disco rígido e também alguns exemplos. Por *defaul*, o diretório de instalação é o *C:\Program Files\vtk42*. Nele são criadas as pastas *bin*, contendo os binários do VTK, que são bibliotecas de vínculo dinâmico (*.dll), *lib*, contendo as bibliotecas (*.lib), *include*, contendo os arquivos *header* (*.h) e outras (figura 4.2.1).

FIGURA 1 – INSTALAÇÃO DO VTK 4.2



FONTE: O autor

2 INSTALAÇÃO DO *VISUAL C++ 2005 EXPRESS EDITION*

O *Visual C++ 2005 Express Edition* (ou VC8) é disponibilizado pela *Microsoft* para ser usado gratuitamente pelo período de um ano. Em (MICROSOFT, 2006) pode ser encontrados os *links* para *download* e também instruções de instalação que são basicamente baixar e executar os programas instaladores do VC8 e da Plataforma Windows SDK (*Software Development Kit*), e depois, configurar o VC8 como descrito a seguir.

O programa de instalação da plataforma SDK (*PSDK-x86.exe*) tem 1.294KB e levou 2 horas e 40 minutos para ser instalado em um Pentiun(R) 4, 2.80GHz, com 256MB de memória RAM. O programa instalador do CV8 (*vcsetup.exe*) tem 2.899KB e levou 1 horas e 15 minutos para ser instalado nesta mesma máquina.

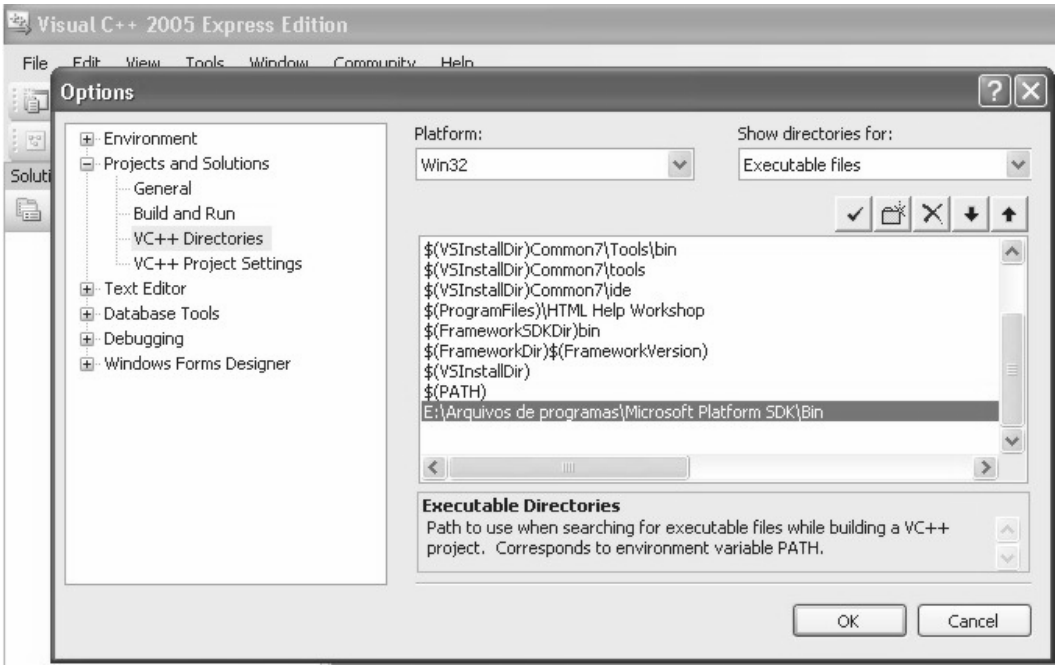
Após ter instalado o VC8 e a plataforma SDK, deve-se configurar os diretórios do VC8 na seção *Projects and Solutions* da caixa de diálogo *Options* (figura 4.2.2), indicando os “caminhos” nas subseções adequadas, como segue abaixo:

Executable files: C:\Program Files\Microsoft Platform SDK\Bin

Include files: C:\Program Files\Microsoft Platform SDK\include

Library files: C:\Program Files\Microsoft Platform SDK\lib

FIGURA 2 – CONFIGURAÇÃO DO VC8 PARA USAR O SDK



FONTE: O autor

O próximo passo é editar o arquivo *corewin_express*.vsprops localizado em *C:\Program Files\Microsoft Visual Studio 8\VC\VCProjectDefaults* e substituir a linha:

AdditionalDependencies="kernel32.lib"

pela linha

AdditionalDependencies="kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib".

E por último, para habilitar a opção *Win32 Application* em *Win32 Aplication Wizard*, é necessário editar o arquivo *AppSettings.htm* localizado em

C:\\ProgramFiles\Microsoft Visual Studio8\VC\VCWizards\AppWiz\Generic\Application\html\1033\".

Com um editor de texto deve-se inserir // no começo das linhas 441-444, como segue:

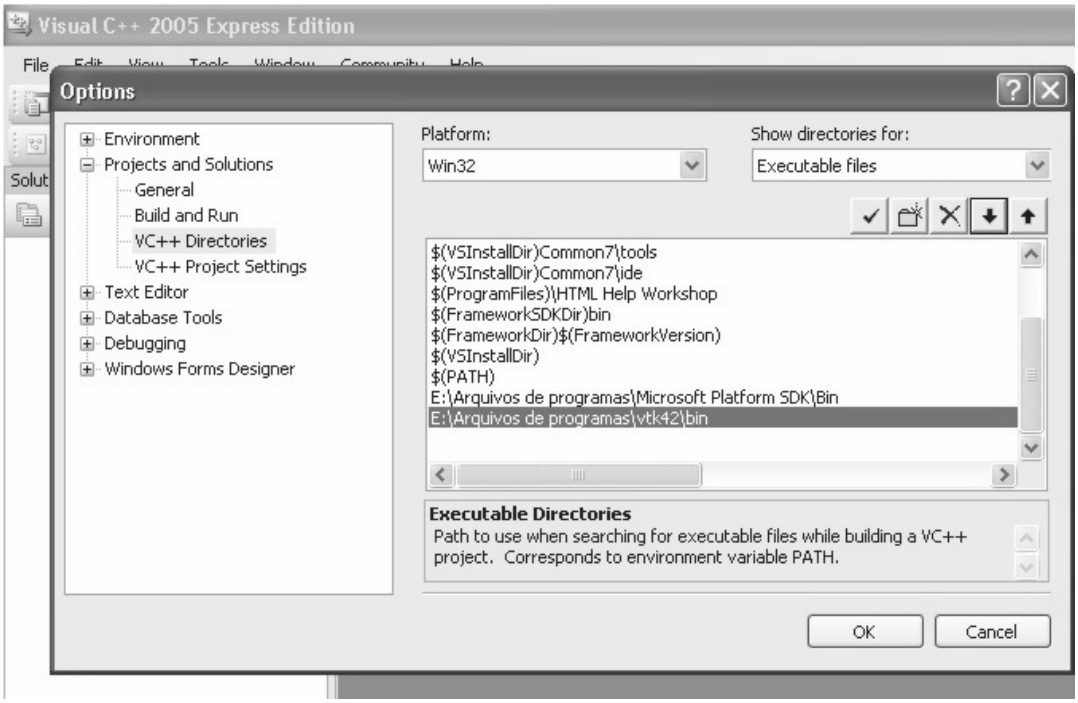
```
// WIN_APP.disabled = true;
// WIN_APP_LABEL.disabled = true;
// DLL_APP.disabled = true;
// DLL_APP_LABEL.disabled = true;
```

1.3 CONFIGURAÇÃO DO VC8 PARA USO COM O VTK 4.2

O procedimento é parecido com o descrito no item anterior bastando configurar os diretórios do VC8 na seção *Projects and Solutions* da caixa de diálogo *Options* (figura 4.2.3), indicando os “caminhos” nas subseções adequadas, como segue abaixo:

Executable files: C:\Program Files\vtk42\Bin
Include files: C:\Program Files\vtk42\include\vtk
Library files: C:\Program Files\vtk42\lib\vtk

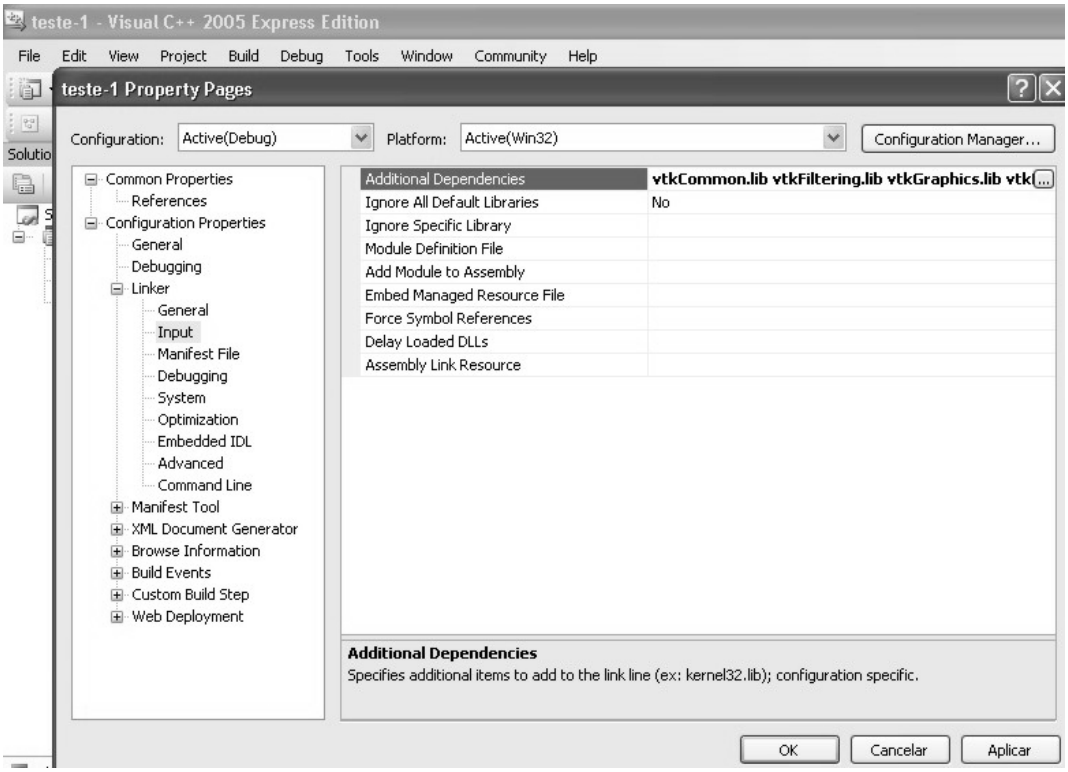
FIGURA 3 – CONFIGURAÇÃO DO VC8 PARA USAR O VTK 4.2



FONTE: O AUTOR

Uma vez criado um novo projeto, é preciso adicionar as bibliotecas vtk*.libs (*vtkCommon.lib vtkFiltering.lib vtkGraphics.lib vtkHybrid.lib vtkImaging.lib vtkIO.lib vtkpng.lib vtkjpeg.lib vtkParallel.lib vtkRendering.lib vtkzlib.lib*) às configurações do projeto, em *Addicional Dependencies* da categoria *Input*, na lista *Linkers* da janela *Property* (Figura 4.2.3.1).

FIGURA 13 – ADIÇÃO DAS BIBLIOTECAS DO VTK 4.2



FONTE: O AUTOR

Alguns problemas podem surgir pelo fato de que as bibliotecas pré-compiladas do VTK 4.2 usam (*defaul*) velhos padrões de arquivos *IO headers/libs* (bibliotecas de entrada e saída de dados). Uma forma de neutralisar este problema é modificar a linha 57 do arquivo *vtkConfigure.h* localizado em *C:\Program Files\vtk42\include\vtk*, subsituindo a frase "*#undef VTK_USE_ANSI_STDLIB*" por "*#define VTK_USE_ANSI_STDLIB*".

Outra forma é copiar os arquivos *iostream.h, useoldio.h, ios.h, streamb.h, istream.h, ostream.h, strstrea.h* e *fstream.h* do VC6 para o diretório *vtk42\include\vtk.* Se uma mensagem de erro dizendo: "*can not open msvcirtd.li*b" aparecer, pode-se ignorar a biblioteca inserindo em *Ignore Specific Library* (ver figura 5) a biblioteca *msvcirtd.lib*. O mesmo pode ocorrer com as bibliotecas *libcid.lib, libcid.pdb, libci.lib* e *libcimt.lib*.

APÊNDICE 3 – CÓDIGO DO ALGORITMO PARA MAPEAMENTO DE CORES E EXPORTAÇÃO PARA VRML

```
//adiciona as classes utilizadas
#include "vtkActor.h"
#include "vtkCubeAxesActor2D.h"
#include "vtkCamera.h"
#include "vtkOutlineFilter.h"
#include "vtkPoints.h"
#include "vtkRenderWindow.h"
#include "vtkRenderWindowInteractor.h"
#include "vtkRenderer.h"
#include "vtkStructuredPoints.h"
#include "vtkStructuredPointsToPolyDataFilter.h"
#include "vtkDataSetMapper.h"
#include "vtkDataSetToPolyDataFilter.h"
#include "vtkLookupTable.h"
#include "vtkPointData.h"
#include "vtkVRMLExporter.h"
#include "vtkScalarBarActor.h"
#include "vtkStructuredPointsWriter.h"
#include "vtkStructuredPointsReader.h"
#include "vtkTextProperty.h"
#include "vtkVRMLExporter.h"
#include "vtkWindowToImageFilter.h"
#include "vtkJPEGWriter.h"

int main(int argc, char *argv[]){
vtkStructuredPoints *dados = vtkStructuredPoints::New();

///////////////Leitor de arquivo VTK
  vtkStructuredPointsReader *reader = vtkStructuredPointsReader::New();
  reader ->SetFileName("arquivo.vtk");
  reader ->OpenVTKFile();
  dados = reader->GetOutput
  reader ->CloseVTKFile();
  dados->Update();

   vtkLookupTable *stc = vtkLookupTable::New();//tabela de cores
   stc->SetRange(dados->GetScalarRange()); // valores máx. e mín.
   stc->SetHueRange (.6667, .0); //cores da tabela(azul->vermelho)
   stc->SetNumberOfTableValues(1024);//define o número de cores da tabela
   stc->Build();//construção da tabela

   vtkDataSetMapper *mapper = vtkDataSetMapper::New();
   mapper->SetInput(dados);
   mapper->SetLookupTable(stc); //define a tabela criada para o mapper
   mapper->SetScalarRange(dados->GetScalarRange());//define a variação
   //dos valores do mapper de acordo com os valores de entrada

   // Cria os atores para o mapper
   vtkActor *ator = vtkActor::New();
   ator->SetMapper(mapper);

   vtkCubeAxesActor2D *atorEixos = vtkCubeAxesActor2D::New ();
   atorEixos->SetInput(dados);
```

```
//Barra com escala de cores
mapper->SetLookupTable(stc);

vtkScalarBarActor *barra = vtkScalarBarActor::New();//barra de cores
barra->vtkScalarBarActor::SetLookupTable(mapper->GetLookupTable());
barra->SetNumberOfLabels(7);
barra->SetOrientationToVertical(); //barra de cores
barra->SetTitle("Escalares");

vtkTextProperty *Texto = vtkTextProperty::New();
Texto->SetFontFamily(VTK_ARIAL);
Texto->SetColor(1,1,1);
Texto->ItalicOn();
Texto->BoldOn();
Texto->ShadowOn();

barra->SetLabelTextProperty(Texto);
barra->SetTitleTextProperty(Texto);

 // Cria uma câmera
 vtkCamera *camera = vtkCamera::New();
 camera->SetPosition(0,0,100);
 camera->SetFocalPoint(0,0,0);
 camera->SetViewUp(0,0,0);
 camera->ParallelProjectionOn();

 atorEixos->SetCamera (camera);

 // Inicializa um renderer
 vtkRenderer *renderer = vtkRenderer::New();

 // Inicializa um RendereWindow
 vtkRenderWindow *renWin = vtkRenderWindow::New();
 renWin->AddRenderer(renderer);
 renWin->SetSize(640,480); //tamanho da janela

 // Inicializa um RenderWindowInteractor, com os eventos padrão
 vtkRenderWindowInteractor *iren = vtkRenderWindowInteractor::New();
 iren->SetRenderWindow(renWin);

 // adiciona o ator ao renderer e define outras propriedades
 renderer->AddActor(ator);
 renderer->AddActor(atorEixos);

 renderer->SetActiveCamera(camera); //define a câmera
 renderer->ResetCamera();
 renderer->SetBackground(.85,.85,.85); //define a cor de fundo
 renderer->AddActor(barra);

 //Renderiza
 renWin->Render();
 camera->Zoom(1.6);
```

```
    iren->Start(); //Inicia o interactor

     //Exporta para VRML
     vtkVRMLExporter *VRML = vtkVRMLExporter::New();
       VRML->SetInput(renWin);
       VRML->SetFileName{"ColorMap2DVRML.wrl");
       VRML->Write();

     //Cria uma imagem JPEG com a imagem renderizada na janela
       vtkWindowToImageFilter *imagem = vtkWindowToImageFilter::New();
       vtkJPEGWriter *JPEG= vtkJPEGWriter::New();
       imagem->SetInput(renWin);
       JPEG->SetInput(imagem->GetOutput());
       JPEG->SetFileName("ColorMap2DJPEG.jpg");
       JPEG->Write();

    //apaga os objetos criados
    atorEixos->Delete();
    stc->Delete();
    ator->Delete();
    barra->Delete();
    camera->Delete();
    renderer->Delete();
    renWin->Delete();
    iren->Delete();
    VRML->Delete();
    JPEG->Delete();
    imagem->Delete();
    return 0;
}
```

APÊNDICE 4 – ALGORITMO PARA EXTRAÇÃO DE ISOSUPERFÍCIES E EXPORTAÇÃO PARA VRML

```
/Programa para criação de isosuperfícies.
#include "vtkActor.h"
#include "vtkCamera.h"
#include "vtkCubeAxesActor2D.h"
#include "vtkPolyData.h"
#include "vtkPolyDataMapper.h"
#include "vtkRenderWindow.h"
#include "vtkRenderWindowInteractor.h"
#include "vtkRenderer.h"
#include "vtkUnstructuredGrid.h"
#include "vtkContourGrid.h"
#include "vtkContourFilter.h"
#include "vtkDataSetMapper.h"
#include "vtkDataSetToPolyDataFilter.h"
#include "vtkLinearSubdivisionFilter.h"
#include "vtkLookupTable.h"
#include "vtkOutlineFilter.h"
#include "vtkProperty.h"
#include "vtkScalarBarActor.h"
#include "vtkStructuredPoints.h"
#include "vtkStructuredPointsToPolyDataFilter.h"
#include "vtkStructuredPointsWriter.h"
#include "vtkStructuredPointsReader.h"
#include "vtkTextProperty.h"
#include "vtkPolyDataNormals.h"
#include "vtkStripper.h"
#include "vtkVRMLExporter.h"

int main( int argc, char *argv[], char key ){
 int i;

  //Define uma estrutura de dados
  vtkStructuredPoints *dados = vtkStructuredPoints::New();

  //Leitor de arquivo VTK do tipo STRUCTURED_POINTS
  vtkStructuredPointsReader *reader = vtkStructuredPointsReader::New();
  reader->SetFileName("arquivo.vtk");
  reader->OpenVTKFile();
  dados = reader->GetOutput();
  reader->CloseVTKFile();
  dados->Update();

  //Define e cria os filtros que são usados
  vtkContourFilter*filtroIsoSup = vtkContourFilter::New();
  vtkOutlineFilter *filtroContornos = vtkOutlineFilter::New();

  //Define e cria os mapeadores que são usados
  vtkDataSetMapper *mapperDados = vtkDataSetMapper::New();
  vtkDataSetMapper *mapperIsoSup = vtkDataSetMapper::New();
  vtkDataSetMapper *mapperTransparente = vtkDataSetMapper::New();
  vtkDataSetMapper *mapperContornos = vtkDataSetMapper::New();

  //Define e cria os atores que são usados
```

```
  vtkActor *atorIsoSup = vtkActor::New();
  vtkActor *atorTransparente = vtkActor::New();
  vtkActor *atorContornos = vtkActor::New();
  vtkScalarBarActor *barra = vtkScalarBarActor::New();
  vtkCubeAxesActor2D *atorCubeAxes = vtkCubeAxesActor2D::New();

   //Aplica filtro de isosuperfícies
  filtroIsoSup->SetInput(dados);
  float *range;
  range = dados->GetScalarRange();
  filtroIsoSup->GenerateValues(7, 0, 1000);
  filtroIsoSup->ComputeNormalsOn();
  filtroIsoSup->ComputeGradientsOn();

  //Aplica filtro de contornos
  filtroContornos->SetInput(dados);

  //Suavização das superfícies
  vtkPolyDataNormals *suav = vtkPolyDataNormals::New();
  suav->SetInput(filtroIsoSup->GetOutput());
  suav->FlipNormalsOn();
  suav->SetFeatureAngle(175.0);

  vtkStripper *striper = vtkStripper::New();
  striper->SetInput(suav->GetOutput());

  //Aplica mapper para os cantos (corners)
  mapperContornos->SetInput (filtroContornos->GetOutput());

  //Aplica mapper para as isosuperfícies
  mapperIsoSup->SetInput(striper->GetOutput());
  mapperIsoSup->SetScalarRange(0,1000);

  //Aplica mapper para a malha transparente
  mapperTransparente->SetInput(dados);
  mapperTransparente->ScalarVisibilityOff();

  //Cria tábua de valores
  vtkLookupTable *stc = vtkLookupTable::New();//inicia tabela de cores
  stc->SetRange(0,1000);
  stc->SetHueRange (.6667, .0); //define variação de cores da tabela
  stc->SetNumberOfTableValues(7);//define o número de cores da tabela
  stc->Build();//construção da tabela

  mapperIsoSup->SetLookupTable(stc);

barra->vtkScalarBarActor::SetLookupTable(mapperIsoSup->GetLookupTable());
  barra->SetNumberOfLabels(7);
  barra->SetOrientationToVertical(); //barra de cores
  barra->SetTitle("Escalares");

  vtkTextProperty *texto = vtkTextProperty::New();
  texto->SetFontFamily(VTK_ARIAL);
```

```
texto->SetColor(1,1,1);
texto->ShadowOn();
texto->BoldOn();
texto->ItalicOn();

vtkTextProperty *textoTitulo = vtkTextProperty::New();
textoTitulo->SetFontFamily(VTK_ARIAL);
textoTitulo->SetColor(1,1,1);
textoTitulo->ItalicOn();
textoTitulo->BoldOn();
textoTitulo->ShadowOn();

barra->SetLabelTextProperty(texto);
barra->SetTitleTextProperty(textoTitulo);

atorIsoSup->SetMapper(mapperIsoSup);
atorIsoSup->GetProperty()->SetInterpolationToPhong();
atorIsoSup->GetProperty()->SetOpacity(1);
atorTransparente->SetMapper(mapperTransparente);
atorTransparente->GetProperty()->SetInterpolationToPhong();
atorTransparente->GetProperty()->SetOpacity(0.28);
atorTransparente->GetProperty()->SetColor(.2,.6,.74);
atorContornos->SetMapper (mapperContornos);
atorContornos->GetProperty()->SetColor (0,0,0);
atorCubeAxes->SetInput(dados);

vtkCamera *camera = vtkCamera::New();
camera->SetPosition(-8,-8,8);
camera->SetViewUp(0, 0, 1);
camera->SetFocalPoint(0,0,0);
camera->ParallelProjectionOn();

atorCubeAxes->SetCamera(camera);   //(camera);
vtkRenderer *renderer = vtkRenderer::New();
vtkRenderWindow *renWin = vtkRenderWindow::New();
renWin->AddRenderer(renderer);
vtkRenderWindowInteractor *iren = vtkRenderWindowInteractor::New();
iren->SetRenderWindow(renWin);

renderer->AddActor(atorIsoSup);
renderer->AddActor(atorContornos);
renderer->AddActor(atorCubeAxes);
renderer->AddActor(atorTransparente);
renderer->AddActor(barra);
renderer->SetBackground(.8,.8,.8);
renderer->SetActiveCamera(camera);
renderer->ResetCamera();
camera->Zoom(1.8);
renWin->SetSize(640,480);
renWin->Render();
iren->Initialize();
iren->Start();
renderer->AddActor(atorIsoSup);
```

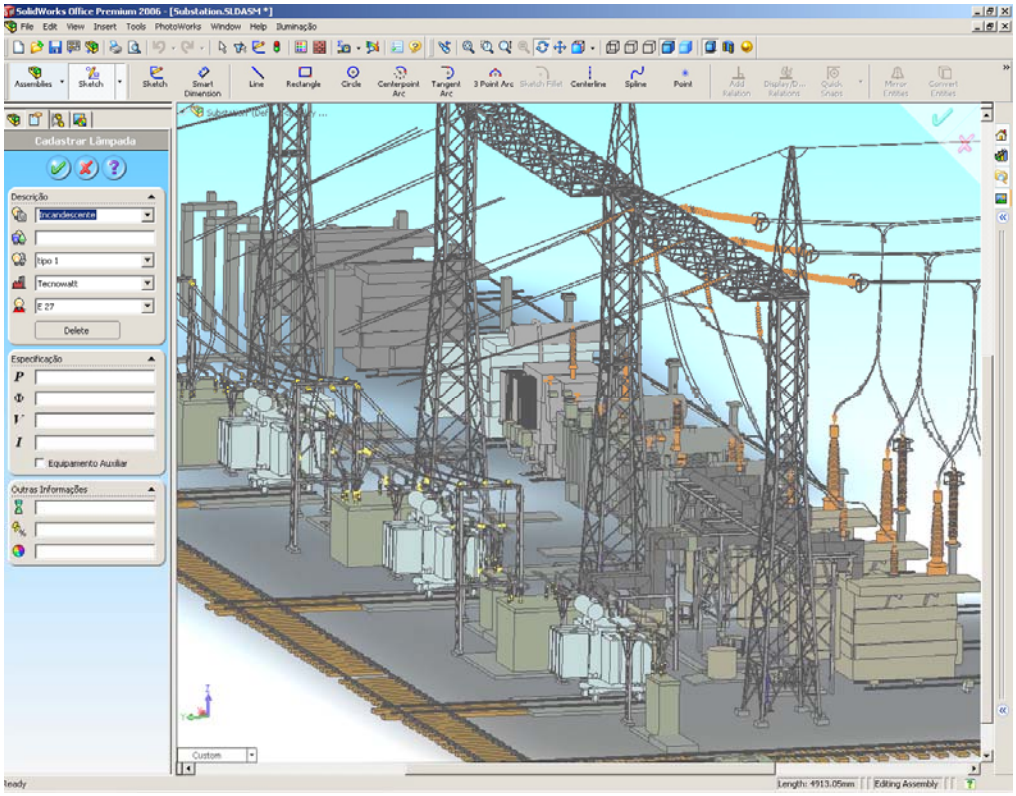
```
renderer->AddActor(atorTransparente);
renderer->AddActor(barra);
renderer->SetBackground(0,0,0);
renderer->SetActiveCamera(camera);

//Exporta para VRML
  vtkVRMLExporter *VRML = vtkVRMLExporter::New();
  VRML->SetInput(renWin);
  VRML->SetFileName("Isosup3dDVRML.wrl");
  VRML->Write();

dados->Delete();
filtroIsoSup->Delete();
filtroContornos->Delete();
mapperIsoSup->Delete();
mapperContornos->Delete();
mapperTransparente->Delete();
atorIsoSup->Delete();
atorContornos->Delete();
atorCubeAxes->Delete();
atorTransparente->Delete();
stc->Delete();
barra->Delete();
texto->Delete();
textoTitulo->Delete();
reader->Delete();
writer->Delete();
camera->Delete();
iren->Delete();
VRML->Delete();
return 0;
}
```

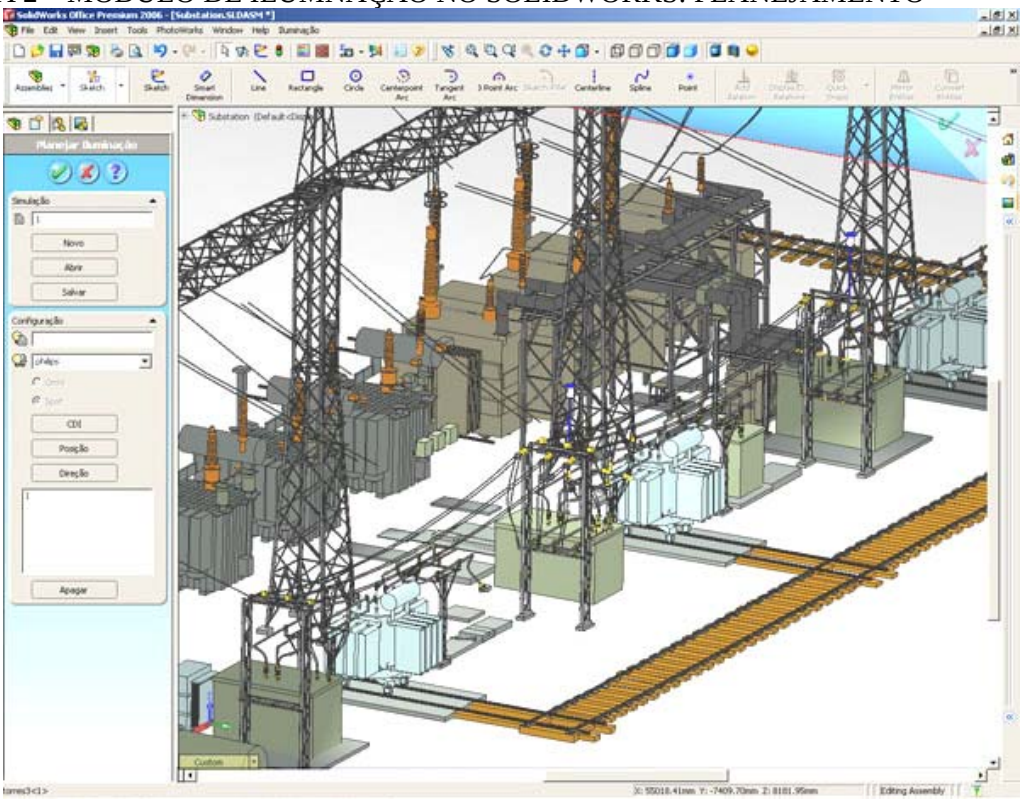
APÊNDICE 5 – IMAGENS ADICIONAIS

FIGURA 1 – MÓDULO DE ILUMINAÇÃO NO SOLIDWORKS: CADASTRO



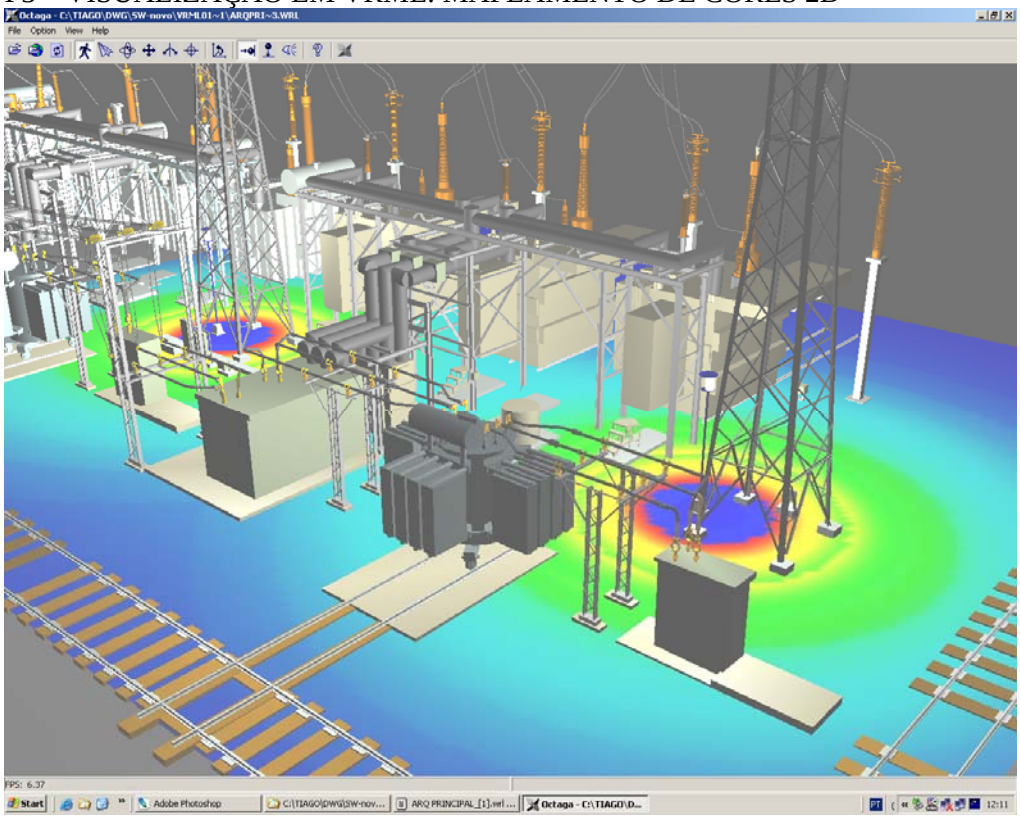
FONTE: O autor.

FIGURA 2 – MODÚLO DE ILUMNAÇÃO NO SOLIDWORKS: PLANEJAMENTO



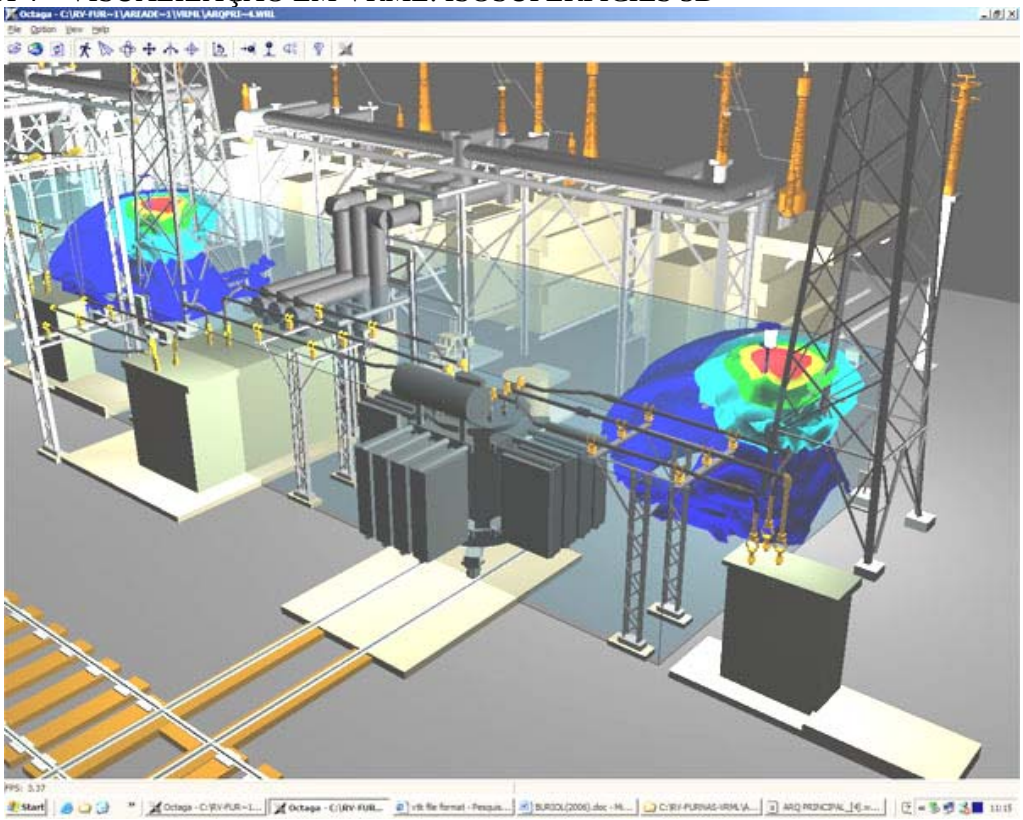
FONTE: O autor.

FIGURA 3 – VISUALIZAÇÃO EM VRML: MAPEAMENTO DE CORES 2D



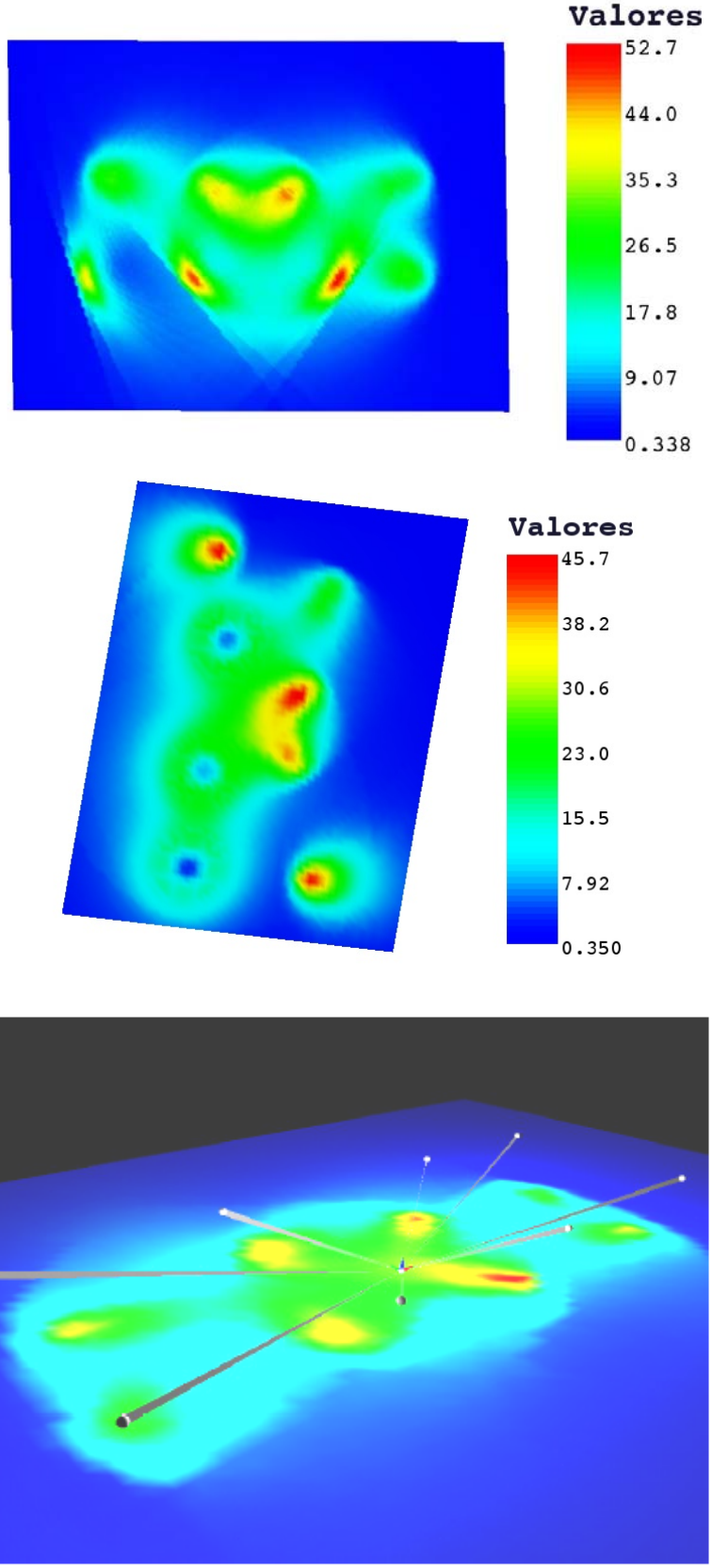
FONTE: O autor.

FIGURA 4 – VISUALIZAÇÃO EM VRML: ISOSUPERFÍCIES 3D



FONTE: O autor.

FIGURA 5 – RESULTADOS DE OUTRAS SIMULAÇÕES



FONTE: O autor.