Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Computação e Estatística

Interface Gráfica Interativa para Definição de Tabelas de Cores em Visualização

Carlos Eduardo de Paula

Orientação: Profa. Dra. Maria Cristina Ferreira de Oliveira

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação -USP, como parte dos requisitos para obtenção do título de Mestre em Ciências -Área de Ciências de Computação e Matemática Computacional.

USP – São Carlos
Abril de 2000

A Deus, por me acompanhar e iluminar em todos os momentos.

"A vós Senhor, toda honra e toda glória. Agora e sempre."

Aos meus pais

José Geraldo e Margarida

Agradecimentos

A Professora Doutora Maria Cristina Ferreira de Oliveira, pela orientação e paciência durante esses anos de trabalho em conjunto.

A Professora Doutora Rosane Minghim pela atenção e contribuições.

A minha família, principalmente minha irmã Néia, minha tia Margarida e minha madrinha Nina, pelo carinho e incentivo em todos momentos.

Aos meus queridos amigos, Fábio Hernandes, Maria José, Selma, Sílvio, Maristela, Nilda, Ricardo, Juliana Oliveira, Sandra, Lilian, Luciana Paiva, Luciane Grossi, Luciana (LABES), Will e Patrícia, pela ajuda e pelos bom momentos que me proporcionaram. E aos parceiros de CG, Alexandre (Irmão), Igor e José Geraldo (Zé), pelo força e socorro.

Aos meus amigos e companheiros de república, Emerson, Douglas, André e Werley, pela paciência, bom convívio e pela força nas horas difíceis.

A Vera e Wanderley, por tudo que têm feito e por tudo que representam pra mim.

Ao senhor Titão e Dona Regina, pais de minha amiga Erika, pelo carinho com que me acolheram ao chegar em São Carlos.

Ao meu amigo Fábio Chinália, pelas contribuições e incentivo e ao meu amigo Sérgio, pelo acolhimento, hospedagem e ombro amigo.

A Flávia, minha querida amiga e parceira de dança, pelo amor, amizade e incentivo, pelas muitas risadas que demos juntos, e pelas noites de farra, dança e alegria.

A todo pessoal do vôlei da Atlética, Batman (Francisco) e companhia, a Lilian , Raquel, Iara e Juliana, pelas excelentes horas de descontração e cansaço.

A todos os funcionários do ICMC, principalmente a Beth, Laura e Marília, pela atenção, zelo e sua eterna disposição em ajudar, não só a mim, como a todos os alunos deste intituto.

A todos com quem convivi nesses anos e que, de algum modo, contribuíram para a realização deste trabalho.

Índice

INTRODUÇÃO **1**

1.1. CONSIDERAÇÕES INICIAIS 1

1.2 MOTIVAÇÃO 3

1.3 ORGANIZAÇÃO DO TRABALHO 4

MODELOS DE DESCRIÇÃO DE CORES **7**

2.1. CONSIDERAÇÕES INICIAIS 7

2.2. DEFINIÇÕES 8

2.3. O OLHO HUMANO 9

2.3.1. Cones e Bastonetes *11*

2.4. TRICROMANCIA 12

2.5. COLORIMETRIA 14

2.6. DIAGRAMA DE CROMATICIDADE DA CIE 15

2.7. MODELOS DE CORES 19

2.7.1. Modelo RGB *19*

2.7.2. Modelo CMY *20*

2.7.3. Modelo HSV *21*

2.7.4. Modelo HLS *23*

2.7.5. Modelo CIE Luv *24*

2.8. CONSIDERAÇÕES FINAIS 26

UMA ABORDAGEM BASEADA EM REGRAS PARA A INCORPORAÇÃO DE CONHECIMENTO SOBRE PERCEPÇÃO AO PROCESSO DE VISUALIZAÇÃO **27**

3.1. CONSIDERAÇÕES INICIAIS 27

3.2. O PROCESSO DE VISUALIZAÇÃO 28
3.2.1. Mapeamento por Cores *31*
3.3. PERCEPÇÃO E VISUALIZAÇÃO 32
3.4. VISUALIZAÇÃO BASEADA EM REGRAS 37
3.5. PRAVDACOLOR 42
3.6. CONSIDERAÇÕES FINAIS 45

INTERFACES EM SISTEMAS DE VISUALIZAÇÃO 47

2.1. CONSIDERAÇÕES INICIAIS 47
4.2. ARQUITETURA DOS SISTEMAS DE VISUALIZAÇÃO 48
4.2.1. Bibliotecas de Visualização *50*
4.2.2. Sistemas Turnkey *51*
4.2.3. Geradores de Aplicação *52*
4.3. ASPECTOS DE INTERAÇÃO 54
4.3.1. Questões Cognitivas e Perceptuais *55*
4.4. INTERFACES DE ALGUNS SISTEMAS 56
4.4.1. Visualization Toolkit *56*
4.4.1.1. A Classe vtkLookupTable *58*
4.4.2. IRIS EXPLORER 59
4.4.3. IBM Open Visualization Data Explorer *62*
4.6. CONSIDERAÇÕES FINAIS 65

IMPLEMENTAÇÃO DA INTERFACE PARA O MÓDULO DE MAPEAMENTO POR CORES BASEADO EM REGRAS 67

5.1. CONSIDERAÇÕES INICIAIS 67
5.2. A CLASSE VTKRULELOOKUPTABLE 68
5.2.1. Tarefa de Isomorfismo *70*
5.2.2. Tarefa de Segmentação *71*
5.2.3. Tarefa de Enfatização *73*
5.2.4. Determinação da Freqüência Espacial *75*
5.2. A NOVA CLASSE VTKRULELOOKUPTABLE 76
5.2.1. Extensão da Nova Classe *78*

5.3. IMPLEMENTAÇÃO DA INTERFACE 80
5.3.1. Funcionamento da Interface *83*
5.3.1. Renderização Volumétrica Direta *85*
5.3.2. Renderização Volumétrica por Isosuperfícies *90*
5.3.3. Características da Implementação *91*
5.4. CONSIDERAÇÕES FINAIS 92

TESTES 95

6.1. CONSIDERAÇÕES INICIAIS 95
6.2. CASOS DE TESTE 96
6.2.1. Dados Marítimos *96*
6.2.2. Dados de Neurônios *99*
6.2.3. Dados de Topografia *102*

CONCLUSÕES 105

REFERÊNCIAS BIBLIOGRÁFICAS 109

APÊNDICE A 115

Índice de Figuras

Figura 2.1- Esquema da estrutura do olho humano *10*
Figura 2.2 - Codificação dos sinais dos cones em sinais oponentes no sistema visual humano *13*
Figura 2.3 –Funções de eficiência luminosa da CIE para a visão escotópica e fotópica *15*
Figura 2.4 – Diagrama de cormaticidade da CIE *17*
Figura 2.5 – Gamas de cores no diagrama de cromaticidade *18*
Figura 2.6 – Modelo de Cores RGB *19*
Figura 2.7 – Mistura subtrativa de cores *21*
Figura 2.8 – Modelo de cores HSV *22*
Figura 2.9 – Cubo de cores do RGB visto ao longo da diagonal principal *22*
Figura 2.10 – Modelo de cores HLS *24*
Figura 3.1 – Processo interativo de visualização *29*
Figura 3.2 – Arquitetura dos sistemas de visualização modernos *30*
Figura 3.3 – Mapeamento linear de valores escalares através de tabelas de cores *31*
Figura 3.4 – Ilusão de Muller-Lyer *33*
Figura 3.5 – Exemplo de oclusão *36*
Figura 3.6 – Efetio do contraste de brilho na percepção do matiz *37*
Figura 3.7 – Arquitetura dos sistemas de visualização modernos com a inclusão de regras *40*
Figura 3.8 – Arquitetura de uma operação baseada em regras *41*
Figura 3.9 – Classificação de mapas de cores baseada no tipo de dados, tarefa de representação e princípios de percepção *43*
Figura 4.1 – Arquitetura de um sistema de visualização genérico *48*
Figura 4.2 – Exemplo de módulo de visualização *49*
Figura 4.3 – Exemplo de aplicação modular de visualização. *52*
Figura 4.4 – Pipelines de um programa VTK *57*
Figura 4.5 – Tabelas de cores geradas pelo VTK *59*
Figura 4.6 – Editor de mapas do IRIS Explorer *60*
Figura 4.7 – Módulo GenerateColormap do IRIS Explorer *61*
Figura 4.8 – Ferramentas do DX no processo de visualização *63*

Figura 4.9 – Editor visual de programas do OpenDX 63
Figura 4.10 – Editor de mapa de cores do OpenDX 64
Figura 5.1 – Esquema da classe vtkRuleLookupTable com principais métodos e atributos 68
Figura 5.2 – Classificação de tabelas para a tarefa de isomorfismo 71
Figura 5.3 – Número de segmentos para tarefas de segmentação, segundo o tipo e a freqüência espacial dos dados 73
Figura 5.4 – Classificação de tabelas para a tarefa de segmentação 73
Figura 5.5 – Classificação de tabelas para a tarefa de enfatização 75
Figura 5.6 – Esquema da classe SpatialFrequency com os principais métodos e atributos 75
Figura 5.7 – Esquematização simplificada do funcionamento do aplicativo 81
Figura 5.8 – Janela principal do VtkRLTInterface 83
Figura 5.9 – Janela de definição do arquivo e do tipo dos dados 83
Figura 5.10 – Resultado da análise do conjunto de dados 84
Figura 5.11 – Janela de definição do tipo e da tarefa de visualização desejada 84
Figura 5.12 – Definição do intervalo de opacidade da tabela 85
Figura 5.13 – Opções de tabelas para tarefas de isomorfismo para dados do tipo intervalo 85
Figura 5.14 – Opções de tabelas para tarefas de isomorfismo para dados do tipo razão 86
Figura 5.15 – Diálogo de definição dos segmentos da tabela de segmentação 87
Figura 5.16 – Diálogo de definição da tabela de enfatização 88
Figura 5.17 – Diálogo que contém a tabela gerada 88
Figura 5.18 – Diálogo de alteração da tabela 89
Figura 5.19 – Configuração da janela de visualização 90
Figura 5.20 – Diálogo de seleção das isosuperfícies 91
Figura 6.1 – Visualizações do volume de dados maritimos utilizando diferentes tabelas de cores 96
Figura 6.2 - Visualizações de superfícies de dados marítimos utilizando diferentes tabelas de cores 98
Figura 6.3 - Visualização de neurônio utilizando escala de cinzas 99
Figura 6.4 - Visualizações do volume do conjunto de dados de neurônios utilizando diferentes tabelas 100
Figura 6.5 - Visualizações de superfícies de neurônios utilizando diferentes tabelas 101
Figura 6.6 - Visualização de dados topográficos utilizando diferentes tabelas de cores 103

Resumo

Este trabalho descreve uma interface gráfica para a utilização da técnica de mapeamento por cores em tarefas de Visualização Científica. Apresentamos, como parte da revisão bibliográfica, um estudo geral sobre cores e sua percepção pelo olho humano, bem como sobre o processo de geração de visualizações e uma classificação dos sistemas de visualização existentes. No escopo do estudo do processo de geração de visualizações, discutimos sua complexidade e algumas abordagens que tentam auxiliar um usuário nesse processo. Enfatizamos a abordagem de visualização baseada em regras, que utiliza regras heurísticas que para incorporar conhecimentos sobre percepção visual de cores e informações sobre os dados ao processo de visualização. Essa proposta serviu de base para a implementação de um módulo de apoio à geração de tabelas de cores para tarefas de visualização. Como parte deste trabalho foi desenvolvida uma interface gráfica interativa para o módulo de mapeamento por cores baseado em regras. A interface oferece ao usuário acesso à biblioteca de tabelas sugerida pelo módulo, e apóia a sua aplicação no contexto de visualização volumétrica por extração de superfícies e por rendering volumétrico direto. Permite, também, a alteração das tabelas, de forma interativa. No final deste trabalho apresentamos algumas visualizações obtidas por meio da interface.

Abstract

This work describes the development and implementation of a graphical user interface to support color mapping in Scientific Visualization tasks. A general survey on color representation in Computer Graphics and on color perception by the human visual system is presented. The process of generating visualizations is discussed, and a classification of current visualization systems is presented. In the discussion of the visualization process, emphasys is placed on some approaches that aim at supporting visualization users in the process of producing effective visualizations. Particularly, rule-based visualization, that relies on heuristic rules to guide users, is discussed in the context of the color mapping technique applied to data visualization. A module that uses a set of rules on color perceptio andnd knowledge about the data and the visualization task to suggest suitable color scales is presented. As part of this work we developed a graphical interface for this rule-based color mapping module. The interface provides direct access to the set of color scales suggested by the rule-based module, and supports their application in volume visualizations using surface extraction or direct volume rendering. Users also have the ability of editing a suggested color mapping in order to suit his/her needs.

Capítulo I

Introdução

1.1. Considerações Iniciais

O aumento do poder computacional e o desenvolvimento de técnicas mais avançadas de simulação criaram uma demanda por ferramentas gráficas em várias áreas de pesquisa que apoiassem computacionalmente o processo de interpretação das informações geradas. A tradução dos dados e informações em um formato gráfico, denominada *visualização*, é essencial para que o ser humano consiga processar dados tão rapidamente quanto necessário.

Pesquisando a literatura da área podemos encontrar várias definições para o termo *visualização*. Brodlie e outros [Bro92] definem a visualização como "a formação de imagens visuais; a ação ou processo de interpretação m termos visuais ou de exibição em forma visual". McCormick e outros [Mcc87] definem a visualização computacional como sendo "um instrumento ou método para a interpretação de dados gerados por um computador e para geração de imagens a partir de conjuntos de dados multidimensionais complexos". Eles afirmam, ainda, que a visualização enriquece o processo de descoberta científica e promove esclarecimentos profundos e inesperados. Haber e McNab [Hab90] dizem que a visualização é

o uso da tecnologia de computação de imagens como uma ferramenta para entender dados obtidos por simulação. Muitas áreas de atuação humana que envolvem a computação têm sido beneficiadas com a produção de visualizações, particularmente a área científica. A aplicação de técnicas gráficas para ampliar a capacidade de interpretação de dados científicos é denominada *visualização científica* (ViSC – *Visualization in Scientific Computing*).

No escopo de atuação da Visualização Científica está a *Visualização Volumétrica*, que compreende a representação, manipulação e o *rendering* de conjuntos de dados volumétricos [Kau93; Kau90]. Esses conjuntos de dados são, geralmente, representados como uma malha tridimensional de células, sendo que a cada ponto da malha pode-se associar um ou mais valores escalares ou vetoriais.

Existem várias maneiras de se representar e gerar imagens de dados volumétricos. No caso de dados escalares, podemos gerar uma representação intermediária (como superfícies) ou gerar a imagem diretamente a partir dos dados (renderização volumétrica direta). Além disso, existe um grande número de técnicas auxiliares para ajudar na visualização de dados vetoriais multidimensionais, tais como *ribbons*, *streamlines*, *streamsurfaces*, *glyphs* dentre outros [Sch99]. Aliada a essas técnicas, é comum o uso da técnica de mapeamento por cores, que consiste no mapeamento de características dos dados em atributos de cor. O resultado desse mapeamento, ou seja, a relação entre os valores dos dados e a cor apresentada, pode ser expresso na forma de uma *tabela de cores*.

A interpretação de dados complexos utilizando técnicas de visualização é uma área em plena expansão [Hea96a]. Muitas técnicas já foram propostas e há muito ainda a ser descoberto. As técnicas atuais, apesar de úteis e eficientes, podem se tornar mais eficazes. Isso se deve ao enorme avanço das tecnologias e o conseqüente aumento da complexidade dos dados a serem visualizados. Aliada a essa complexidade, existe a dificuldade de mapear graficamente dados multidimensionais para dispositivos bidimensionais sem que haja perda de informações. O tratamento desses problemas exige pesquisas sobre aspectos relacionados à percepção visual e sonora pelos seres humanos e sobre mecanismos eficientes de comunicação de informações, incluindo o uso de meios alternativos de representação, como o som [Min95; Sal98].

Outro problema associado ao uso dos atuais sistemas e técnicas de visualização é a falta de orientação aos usuários quanto ao melhor mapeamento de atributos dos dados em representações visuais. A grande variedade de representações visuais existentes dificulta a

criação de uma imagem que realmente ajude a analisar o fenômeno subjacente aos dados. Por isso, existe um grande interesse no desenvolvimento de ferramentas que ajudem a escolher uma representação visual de dados que seja realmente efetiva.

1.2 Motivação

Um dos grande desafios da Visualização é apresentar informações de maneiras mais adequadas ao entendimento do usuário. As técnicas e estratégias de visualização disponíveis atualmente são muitas, mas o que se verifica na prática é que usuários não especialistas têm dificuldades em gerar uma visualização efetiva. Isso porque a maioria das técnicas ainda exige do usuário um conhecimento sobre o processo de visualização e suas técnicas, o que um usuário comum ainda não possui. As características dos dados e do domínio de aplicação, a natureza das tarefas a serem suportadas pela visualização e conhecimento sobre o processo de percepção visual são aspectos que devem ser melhor considerados no mapeamento de informações relevantes em elementos gráficos da visualização [Ear97].

Profissionais da área de visualização têm se voltado para aspectos psicológicos na busca de soluções para estes problemas [Rog93a; Rog93b] e princípios de percepção têm sido estudados já há algum tempo, com o objetivo de enriquecer e fortalecer técnicas de computação gráfica [Mey80]. Vários aspectos relacionados à percepção, como a identificação de cor, textura e forma, estão sendo estudados e incorporados a sistemas de visualização [Gri95]. Entretanto, a ausência de uma abordagem mais direta em relação ao problema tem limitado o alcance de importantes técnicas de visualização. Acredita-se que algoritmos e interfaces podem ser melhorados se forem adaptados para tratar aspectos cognitivos e de percepção visual, alguns dos quais dependem do tipo de aplicação.

Para os objetivos da visualização, a escolha de mapeamentos de cores adequados é particularmente importante. Uma tabela de cores adequada deve evidenciar as características relevantes do conjunto de dados a ser estudado, de modo a efetivamente ajudar o usuário a entender a estrutura dos dados [Ber95]. Em geral, os sistemas de visualização oferecem uma tabela de cores *default* (normalmente a escala *rainbow*) e permitem ao usuário definir suas próprias tabelas, porém, não o auxiliam na escolha ou na geração de uma tabela adequada. Assim, o uso de mapeamentos para cores que não levam em consideração aspectos de percepção visual e as características dos dados é bastante comum..

Este projeto abrange alguns tópicos no âmbito do desenvolvimento e melhoria de algoritmos para Visualização Volumétrica, em continuação a trabalhos anteriores desenvolvidos no ICMC [Tut98b, Fod98]. Mais especificamente, explora o uso de aspectos de percepção visual de cores na implementação de um assistente para criação de tabelas de cores em um sistema de visualização. Esse programa assistente, desenvolvido por Tutida [Tut98] e Fodra [Fod98], atua juntamente com um procedimento que implementa um conjunto de regras para a definição de mapeamentos de dados escalares para cores. Essas regras são baseadas em critérios como a natureza dos dados, sua freqüência espacial, a tarefa de visualização a ser realizada, e a área de aplicação, restringindo o número e o conjunto de cores que podem estar contidos em uma tabela. Como parte deste projeto de mestrado foi criada uma interface gráfica para acesso do usuário às opções oferecidas pelo assistente.

1.3 Organização do Trabalho

Neste capítulo apresentamos algumas considerações gerais e as bases para o desenvolvimento deste trabalho.

No capítulo 2 apresentamos um estudo geral sobre cores, sua percepção pelo ser humano e modelos de cores. Dentre os tópicos abordados podemos citar: alguns aspectos relacionados à representação e descrição de cores, a terminologia básica utilizada, a fisiologia do olho humano e como este capta a sensação de cor, algumas teorias existentes que tentam explicar o funcionamento da visão humana, o estudo da colorimetria e alguns modelos de cores utilizados em Computação Gráfica.

No capítulo 3 abordamos o processo de geração de visualizações e sua contextualização. Discutimos a importância do conhecimento sobre o processo de percepção visual de cores na geração de visualizações efetivas. Uma proposta de incorporação desses aspectos ao processo, na qual este trabalho está baseado, é apresentada.

No capítulo 4 apresentamos uma arquitetura genérica de sistemas de visualização e uma classificação desses sistemas baseada na arquitetura e no tipo de interface oferecida. Discutimos alguns aspectos de interação relevantes em uma interface, dentre os quais aspectos cognitivos, perceptuais e relacionados ao uso de cores. Apresentamos três sistemas de visualização atuais: o *IRIS Explorer* e o *IBM Open Visualization Data Explorer*

(OpenDX), enfatizando suas interfaces com o usuário; e o *Visualization Toolkit* (VTK), utilizado neste trabalho.

No capítulo 5 descrevemos a implementação da classe *vtkRuleLookupTable* desenvolvida por Tutida [Tut98b], uma nova versão dessa classe desenvolvida por Fodra [Fod98]. A seguir, relatamos as alterações efetuadas sobre a nova versão da classe como parte deste trabalho e descrevemos a implementação da interface gráfica.

No capítulo 6 são apresentadas algumas imagens ilustrativas dos resultados obtidos com a aplicação dos diferentes tipos de tabelas de cores geradas pelo aplicativo. As conclusões deste trabalho são apresentadas no capítulo 7.

Capítulo II

Modelos de Descrição de Cores

2.1. Considerações Iniciais

O uso eficaz de cores em Computação Gráfica e, particularmente, visualização requer informações sobre a relação entre a percepção de cores pelos usuários e os sistemas ou modelos de cores existentes. A sensação de cor é causada pela excitação dos sistema visual humano por estímulos luminosos que se propagam como ondas eletromagnéticas. As diferentes cores que podem ser percebidas pelo sistema visual humano correspondem a um pequeno intervalo do espectro eletromagnético, que inclui as ondas de rádio, microondas, os raios infravermelhos e os raios X [Gro94]. Nesse intervalo, conhecido como espectro visível, o menor comprimento de onda corresponde à cor violeta (440nm[1]) e o maior, à cor vermelha (700nm), passando pelo laranja, amarelo, verde e azul. O estudo de aspectos físicos, fisiológicos, psicofísicos, perceptuais e cognitivos desse sistema se faz necessário para que as imagens sejam produzidas de forma a refletir um conteúdo informativo fiel. O processo de

[1] Nanômetros (nm) é uma medida física que eqüivale a 10^{-9}m. Na literatura também pode ser encontrada a unidade μ, denominada milimicron e que também eqüivale a 10^{-9}m.

percepção de cor é individual, de forma que indivíduos reagem de maneira diferente aos mesmos estímulos. Com o objetivo de tentar expressar e padronizar as descrições das cores existentes e visíveis ao olho humano é que surgiram os sistemas (ou modelos) de cores. Um modelo de cores é um mecanismo que permite descrever sistematicamente as propriedades ou o comportamento de cores em um contexto particular. Nenhum modelo de cores pode explicar todos os aspectos e fenômenos relacionados a cores. Assim, diferentes modelos são usados para descrever diferentes características perceptuais da cor [Hea94].

Em geral, os modelos de cores descrevem cores como combinações de duas ou mais cores primárias que geram um conjunto de cores denominado espaço ou gama de cores (color gamut) do modelo. Um único modelo de cores não é capaz de reproduzir todas as cores visíveis ao olho humano e existem modelos de difícil utilização por não fornecerem uma forma intuitiva de especificação de cor.

Nesse capítulo discute-se brevemente os aspectos relacionados a representação e descrição de cores. Na seção 2.2 é descrita a terminologia básica utilizada neste trabalho. Na seção 2.3 são abordados a fisiologia do olho humano e como este capta a sensação de cor. A seção 2.4 apresenta algumas teorias existentes que tentam explicar o funcionamento da visão humana. Na seção 2.5 dá-se início ao estudo da colorimetria, que nada mais é que o estudo da cor. A seção 2.6 apresenta o diagrama de cromaticidade da CIE (*Comission Internacionale L'Eclairage*) e alguns modelos de cores utilizados em Computação Gráfica que são relevantes para o desenvolvimento deste trabalho.

2.2. Definições

Os termos utilizados por cientistas e estudiosos com relação a cor são muito variados. É importante ter em mente uma idéia clara do que cada um deles significa para que haja uma comunicação efetiva. Para isso, apresenta-se algumas definições de termos que serão mencionados nesse trabalho. Existem aproximadamente 950 termos utilizados no estudo da cor e luz [Fai98]. Apenas os mais comuns e citados são descritos a seguir, como definidos por Fairchild e Hunt [Fai98; Hunt78]:

Cor (*color*): atributo de percepção visual que pode ser descrito pelo nomes das cores: branco, preto, cinza, amarelo, vermelho, etc.

Cor acromática (*achromatic color*): cor percebida desprovida de um matiz.

Cor cromática (*chromatic color*): cor percebida contendo um ou mais matizes.

Matiz (*hue*): atributo da sensação visual segundo o qual uma área parece similar a uma das cores percebidas vermelho, amarelo, laranja, verde, azul e púrpura; ou a uma combinação de duas delas.

Saturação (*saturation*): atributo de uma sensação visual segundo o qual uma área parece exibir mais ou menos cor cromática, julgada em proporção ao seu brilho.

Brilho (*brightness*): atributo de uma sensação visual de acordo com a qual uma área parece emitir, transmitir ou refletir mais ou menos luz.

Luminosidade (*lightness*): atributo de uma sensação visual segundo o qual uma área parece refletir difusamente, ou transmitir, uma maior ou menor fração da luz incidente.

Cromaticidade (*chromaticness*): atributo de uma sensação visual usado para se referir coletivamente a duas propriedades que descrevem características das cores: saturação e matiz [Hea94].

Fator de luminância (*luminance factor*): razão entre a luminância de um corpo e de um difusor de reflexão ou transmissão perfeita identicamente iluminados.

Gama ou **espaço de cores** (*color gamut*): o conjunto de cores que podem ser reproduzidas por um determinado sistema de cores.

2.3. O Olho Humano

Esta seção fornece uma visão geral do sistema visual humano. É interessante estudar a estrutura do olho humano, visto que é a partir dele que a percepção visual se inicia. A figura 2.1 mostra um representação esquemática simplificada da estrutura do olho humano. O olho age como uma câmera. A córnea e a lente (cristalino) funcionam como as lentes de um câmera para projetar uma imagem do mundo real na retina, localizada na parte de trás do olho, que age como um filme.

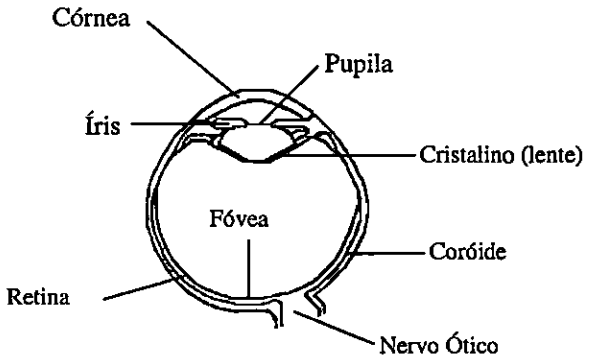


Figura 2.1 – Esquema da estrutura do olho humano [Fai98].

A córnea é a superfície transparente que reveste a parte frontal do globo ocular. Por causa da interface entre a superfície curva da córnea e o ar, e ainda pela grande mudança de índice de refração no sistema óptico do olho, ela representa um elemento importante na formação da imagem. Problemas na visão, como miopia, hipermetropia e astigmatismo, podem ser atribuídos a variações no formato da córnea. A íris é o músculo que controla o diâmetro da pupila. É ela que dá a cor ao olho, cor essa determinada pela concentração e distribuição de melanina na íris. A pupila, que é o orifício no meio da íris através do qual a luz passa, define o nível de iluminação que atinge a retina.

A lente (cristalino) é uma estrutura flexível e transparente que varia o índice de refração, sendo este maior no centro da estrutura. Com o avanço da idade, a estrutura interna das lentes se modifica, ocasionando a perda de sua flexibilidade. Paralelo a isso, ocorre o aumento de sua densidade ótica, tornando-se mais amareladas com o passar do tempo [Fai98].

A fóvea é talvez a área estrutural mais importante da retina, onde temos a melhor visão espacial e de cores. Ela engloba uma área de cerca de 2 graus do ângulo visual na parte central do campo de visão [Fai98].

A retina é uma fina camada de células localizada na porção posterior do olho. Essas células são neurônios normalmente conhecidas como fotoreceptores responsáveis por traduzir um estímulo luminoso incidente em sinais elétricos e químicos. Os dois principais tipos de fotoreceptores são os cones (aproximadamente 8 milhões) e os bastonetes (aproximadamente 120 milhões) [Lev97]. Por serem esses os responsáveis pela absorção dos estímulos luminosos, são tratados em mais detalhe.

2.3.1. Cones e Bastonetes

A principal distinção entre os bastonetes e os cones está na sua função visual. Os bastonetes são responsáveis pela visão em baixos níveis de luminosidade (visão escotópica) e não captam informação alguma de cor, enquanto que os cones se responsabilizam pela visão em altos níveis de luminosidade (visão fotópica) e pela visão das cores [Mon98]. Em níveis intermediários ambos, cones e bastonetes, funcionam (visão mesópica). Esse mecanismo nos permite enxergar num extenso intervalo com vários níveis de luminosidade. Porém, os bastonetes são incapazes de captar a sensação de cor e isso pode ser facilmente comprovado quando olhamos uma cena colorida em níveis de luminosidade muito baixos [Fai98].

Existe somente um tipo de bastonete, enquanto há três tipos de cones. Os três tipos de cones são, normalmente, conhecidos por L, M e S, referindo-se aqueles cuja sensibilidade é maior nos comprimentos de onda longo (*long*), médio (*middle*) e curto (*short*), com os picos de sensibilidade em 570nm, 550nm e 440nm, respectivamente [Lev97]. Uma importante característica dos três tipos de cones é a sua distribuição na retina, que é de aproximadamente 40:20:1 para os cones L:M:S. Os cones do tipo S são completamente ausente na região central da fóvea.

Aproximadamente 7% da população masculina e cerca de 0,4% da população feminina possuem algum tipo de deficiência na visão de cores, em geral causada por anomalias em algum dos três tipos de cones [Lev97]. Essa diferença grande no percentual de homens atingidos por algum anomalia na visão se deve ao fato dessas anomalias estarem ligadas ao cromossomo X, do qual os homens possuem apenas uma cópia. As mulheres estão mais protegidas contra essa anomalia porque possuem dois cromossomos X e um gene normal muitas vezes compensa um gene defeituoso [Mon98].

Uma característica relevante dos bastonetes é que eles são ausentes na fóvea. Isso pode ser constatado quando tentamos olhar um objeto iluminado pequeno e distante, como por exemplo uma estrela. O objeto parece desaparecer porque sua imagem não é captada na região da fóvea onde não há bastonetes para detectar seu brilho. Essa falta de bastonetes na fóvea permite que a concentração de cones seja maior e assim o espaço seja usado para produzir uma maior precisão com o sistema de cones.

Esses são os principais componentes do olho humano responsáveis pela percepção dos estímulos luminosos que atingem o sistema visual humano. Porém, para entender melhor a percepção de cores, é preciso compreender o papel dos cones e bastonetes

no processo visual, discutido na seção 2.3, e a natureza da cor, discutida na seção 2.4.

Nessa e nas próximas seções não são abordados todos os aspectos do processo de percepção visual, que está somente se iniciando com a detecção da energia luminosa pelo fotoreceptores na retina. Existem vários outros níveis superiores de processamento que não são abordados ou por sua extensão ou por não estarem diretamente relacionados a este trabalho. Mais informação pode ser obtida em Fairchild [Fai98] e Levkowitz [Lev97].

2.4. Tricromancia

Há muitas teorias que tentam explicar o funcionamento da visão humana e o processo de percepção de cores. Uma delas é a Teoria Tricromática atribuída a Thomas Young, na qual ele assume que três imagens do mundo, uma para cada primária, eram percebidas pelo olho e então somadas para formar a imagem final. Young não acreditava que a retina fosse sensível a todas as partes do espectro visível e propôs a existência de três tipos de partículas sensíveis à luz, uma mais sensível à cor vermelha, uma à cor verde e outra à cor azul [Bok95; Fri91]. HermanVon Helmholtz refinou essa idéia, descobrindo os três tipos de receptores contidos no olho (Teoria Young-Helmholtz). Isso explica porque não é possível distinguir entre um comprimento de onda puro e uma cor resultante, por exemplo, da combinação de duas primárias. Segundo essa teoria, por exemplo, a mistura das luzes verde e vermelha é percebida como sendo uma luz amarela por que elas estimulam respostas dos cones verde e vermelho, que são iguais às respostas estimuladas por um amarelo espectral [Fri91].

Baseado na observação de que certas combinações de matizes nunca são percebidas, Hering propôs a "Teoria das Cores Oponentes". Por exemplo, uma cor nunca é descrita como verde avermelhado ou azul amarelado, enquanto combinações de vermelho e amarelo são percebidas com facilidade. Tal fato sugeriu a Hering que algo fazia com que os pares vermelho-verde e azul-amarelo fossem opostos um ao outro. Ele também observou que pessoas com deficiência visual não são capazes de distinguir esse pares. Hering então propôs a existência de três tipos de receptores que tinham respostas bipolares aos pares claro-escuro, vermelho-verde e azul-amarelo. Em meados do século 20, surge a Teoria Moderna das Cores Oponentes. Essa teoria, ilustrada na figura 2.2, afirma que os sinais captados pelos três tipos de cones não são transmitidos diretamente ao cérebro, havendo uma codificação dos sinais dos cones pelos neurônios da retina em sinais opostos. Um sinal é formado pela soma das

saídas dos três tipos de cones (L + M + S) para produzir uma resposta acromática, e os outros dois sinais oponentes vermelho-verde e azul-amarelo, por L-M+S e L+M-S, respectivamente. [Fai98].

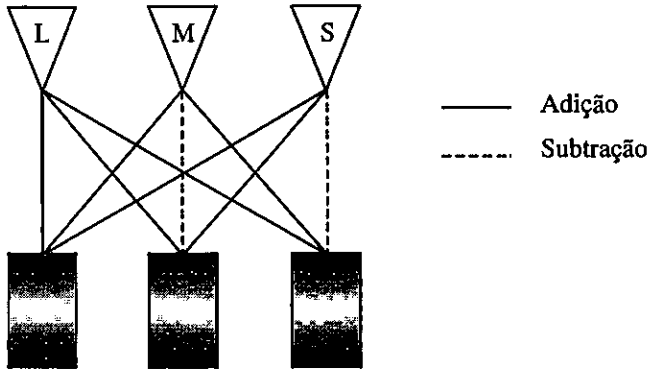


Figura 2.2 – Codificação dos sinais dos cones em sinais de cores oponentes no sistema visual humano [Fai98].

O fenômeno da visão começa na retina, onde os cones fornecem três filtros, ajustados para três comprimentos de onda (como visto anteriormente). A essência da tricromancia é que o matiz percebido depende do vetor tridimensional de sinais detectado pelos três tipos de cones em combinação. Isso sugere que qualquer matiz pode ser formado pela combinação de três primárias e que qualquer matiz pode ser produzido por meio de um número finito de combinações de comprimentos de onda.

Quaisquer três cores que são linearmente independentes ou que não resultam da mistura de uma com as outras podem ser definidas como primárias. Há dois tipos de misturas na produção de cores: a mistura aditiva e a subtrativa [Lev97].

A mistura aditiva é o processo de combinar a luz emitida por diversas fontes que emitem luz em diferentes partes do espectro visível. O preto é o resultado da mistura de nenhuma cor (energia zero) e o branco é o resultado da mistura das quantidades máximas das primárias (energia máxima). Na TV colorida e nos monitores de vídeo convencionais as cores são geradas por esse tipo de processo.

A mistura subtrativa é o processo de filtrar a reflexão de partes do espectro visível. Neste caso, branco é o resultado da mistura de nenhuma primária (o espectro inteiro é refletido), enquanto que a mistura das quantidades máximas das primárias não produz reflexão da luz, ou seja, será o preto.

Esse processo de mistura de cores esclarece como se obtém um certo números

de cores a partir de um conjunto de cores primárias.

2.5. Colorimetria

Colorimetria é o estudo da cor por meio da mistura aditiva [Gom94]. Segundo Wyszecki [Wys73 apud Fai98][2] ela é uma ferramenta usada para dizer se duas fontes de luz (estímulo visual) de diferentes distribuição espectral resultarão numa mesma cor sob dadas condições de observação.

Para entender a colorimetria é preciso considerar a natureza da cor. A aparência colorida dos objetos depende de 3 (três) componentes [Fai98]: de uma fonte de luz (energia eletromagnética visível), do objeto e do sistema visual humano. A fonte de luz é necessária para iniciar o processo da visão e é quantificada por meio da sua distribuição espectral (medida como uma função de comprimento de onda cujo intervalo de interesse varia aproximadamente entre 400nm (violeta) e 700nm (vermelho), ou seja, no espectro visível). O objeto modula a energia eletromagnética incidente, sendo especificado por sua geometria e pela distribuição espectral que reflete ou transmite. Essa energia é percebida pelo olho humano, ou seja, detectada pelos fotoreceptores e processada pelos mecanismos do sistema visual para produzir a percepção da cor. O sistema visual, o terceiro componente, quantificado por suas propriedades de "casamento de cor", representa o primeiro estágio da visão (absorção da energia luminosa pelos cones). Dessa maneira, a colorimetria engloba técnicas concebidas em estudos de física, química, psicofísica, fisiologia e psicologia [Fai98].

A medição e a padronização das fontes de luz e dos materiais fornecem a informação física necessária para a colorimetria. Porém, é necessária uma técnica quantitativa para prever a resposta do sistema visual humano aos estímulos luminosos. De acordo com a definição de colorimetria de Wyszecki, essa quantificação ocorre no estágio inicial da visão e é definida pelas respostas dos três tipos de cones, $L(\lambda)$, $M(\lambda)$ e $S(\lambda)$. Se os sinais dos três tipos de cones forem iguais para dois estímulos distintos vistos sob as mesmas condições, ocorrerá o casamento da cor já que, uma vez que a energia luminosa foi absorvida pelos cones, nenhuma informação adicional é introduzida no sistema [Fai98].

[2] Wyszecki, G. – "Current Developments in Colorimetry", AIC Color 73, 1973, pp. 21-51. Apud Fairchild, M. D. – "Color Appearence Models", Addison-Wesley, 1998.

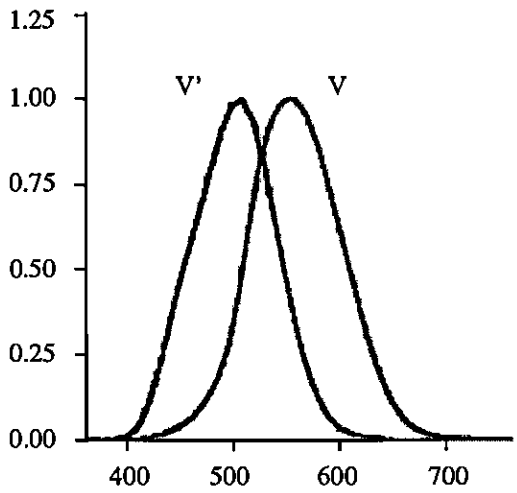


Figura 2.3 – Funções de eficiência luminosa da CIE para a visão escotópica, V'(λ), e fotópica, V'(λ) [Fai98].

Para entender melhor a colorimetria, é interessante estudar o sistema de fotometria estabelecido em 1924 porque ele é a base para o desenvolvimento do diagrama de cromaticidades da CIE. Esse sistema foi desenvolvido com o intuito de encontrar uma função que pudesse ser usada para descrever a percepção do brilho. Naquela data, a função de eficiência luminosa da CIE (*Comission Internacionale L'Eclairage*), V(λ), foi estabelecida para a visão fotópica. Essa função é ilustrada na figura 2.3 onde se vê que o sistema visual é mais sensível (no que se refere à percepção de brilho) a comprimentos de onda no meio do espectro visível, e menos sensível nas extremidades. Uma função de eficiência luminosa para a visão escotópica (bastonetes), V'(λ), ou seja, visão em baixos níveis de luminosidade, também foi estabelecida. O pico de sensibilidade é deslocado na direção dos menores comprimentos de onda, o que explica o porquê de objetos azuis parecerem mais claros que os vermelhos em baixos níveis de luminosidade.

2.6. Diagrama de Cromaticidade da CIE

Após estabelecida a função de eficiência luminosa da CIE 1924, V((), o objetivo foi desenvolver um sistema de colorimetria que pudesse ser usado para especificar uma cor para um observador normal [Fai98]. Visto que as receptividades dos cones não eram conhecidas naquele tempo, um sistema de colorimetria foi desenvolvido baseado nos princípios da tricromancia da mistura aditiva de cores. O conceito desse sistema é que cores podem ser especificadas em termos das quantidades de três primárias, como ilustrado na

equação 2.1.

$$C \equiv R(\boldsymbol{R}) + G(\boldsymbol{G}) + B(\boldsymbol{B}) \tag{2.1}$$

A cor C é obtida por meio de R unidades da primária ***R***, por G unidades da primária ***G*** e por B unidades da primária ***B***. Essas quantidades são chamadas "valores triestímulos". Alguns valores triestímulos espectrais, porém, são negativos para primárias monocromáticas em 435,6nm (***B***), 546,1nm (***G***) e 700,0nm (***R***) [Fai98].

Na intenção de eliminar esses valores negativos decidiu-se transformar esse conjunto de primárias em um outro com as primárias ***XYZ***. Isso foi feito escolhendo primárias imaginárias (virtuais) que são mais saturadas do que as luzes monocromáticas e forçando uma das funções de casamento de cor a ser igual à função de eficiência luminosa. Sendo assim, em 1931 a CIE definiu três cores primárias (***X***, ***Y***, ***Z***) que podem ser combinadas com pesos positivos de forma a definir todas as sensações de luz que experimentamos. Esse sistema é denominado CIE XYZ ou xyY [Bou95]. As primárias da CIE, que não são visíveis, definem um padrão internacional para espècificação de cor. Nesse sistema uma cor C(pode ser expressa pela equação 2.2.

$$C_\lambda = x\boldsymbol{X} + y\boldsymbol{Y} + z\boldsymbol{Z} \tag{2.2}$$

onde x, y e z definem as quantidades das primárias padrão necessárias para descrever uma cor espectral. As cores desse sistema podem ser expressas pelas quantidades, ou valores de cromaticidade, normalizadas conforme as equações 2.3, 2.4 e 2.5 abaixo, onde x + y + z = 1:

$$x = \frac{\boldsymbol{X}}{\boldsymbol{X}+\boldsymbol{Y}+\boldsymbol{Z}} \tag{2.3}$$

$$y = \frac{\boldsymbol{Y}}{\boldsymbol{X}+\boldsymbol{Y}+\boldsymbol{Z}} \tag{2.4}$$

$$z = \frac{\boldsymbol{Z}}{\boldsymbol{X}+\boldsymbol{Y}+\boldsymbol{Z}} \tag{2.5}$$

Dessa maneira, qualquer cor pode ser definida pelos parâmetros x e y que, por estarem normalizados e dependerem apenas do matiz e saturação, são denominados "coordenadas de cromaticidade". Assim, a descrição completa de uma cor é dada pelas coordenadas de cromaticidade e pelo valor de um dos estímulos originais, normalmente o Y, que contém a informação de luminância. A partir dessa descrição obtemos as quantidades de X e Z expressas nas equações 2.6 e 2.7, onde z = 1 – x – y.

$$X = \frac{x}{y}Y \tag{2.6}$$

$$Z = \frac{z}{y}Y \tag{2.7}$$

Traçando o gráfico de x e y para todas as cores visíveis, obtemos uma representação bidimensional denominada "diagrama de cromaticidade" da CIE, mostrado na figura 2.4. Este contém todas as cores visíveis. As cores puras do espectro estão sobre a curva do gráfico (linha cheia) e seus comprimentos de onda estão indicados nesta figura. A luz branca está localizada no centro do gráfico (ponto C), próxima ao ponto x = y = z = 1/3. O segmento de reta que une os pontos vermelho e violeta é chamado de "linha púrpura" (linha pontilhada) e não faz parte do espectro visível [Hea94; Gom94; Mey86].

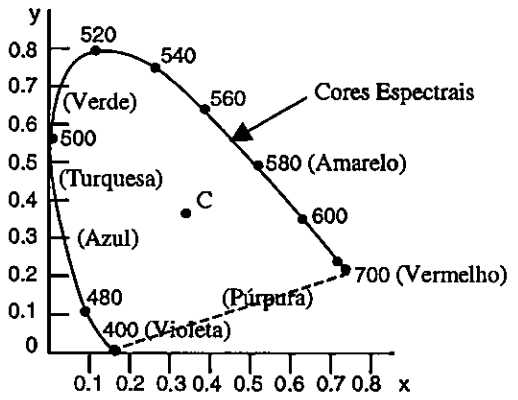


Figura 2.4 – Diagrama de cromaticidade da CIE [Hea94].

Os valores de luminância não estão representados no diagrama de cromaticidade por causa da normalização. Cores com luminâncias diferentes, mas com a mesma cromaticidade, são mapeadas no mesmo ponto. O diagrama de cromaticidade é muito útil, por exemplo, para comparar as gamas de cores geradas por diferentes conjuntos de primárias, para identificar cores complementares e para determinar o comprimento de onda dominante e a saturação de uma cor [Hea94].

Gamas de cores são representados no diagrama de cromaticidade como segmentos de retas ou como polígonos. Todas as cores ao longo da reta entre 2 pontos, por exemplo os pontos C_1 e C_2 na figura 2.5(a), podem ser obtidas pela mistura de quantidades diferentes das cores C_1 e C_2. Se uma quantidade maior da cor C_1 é usada, a cor resultante estará mais próxima de C_1. O gama de cores para 3 pontos, como mostrado na figura 2.5(a)

para os pontos C_3, C_4 e C_5, é um triângulo com vértices definidos pelas posições das 3 cores. As três primárias podem gerar apenas as cores dentro ou na borda do triângulo. Dessa maneira, o diagrama de cromaticidade ajuda a entender porque nenhum conjunto de três primárias pode ser usado para gerar todas as cores do espectro visível, já que nenhum triângulo inscrito ao diagrama pode abranger todas as cores.

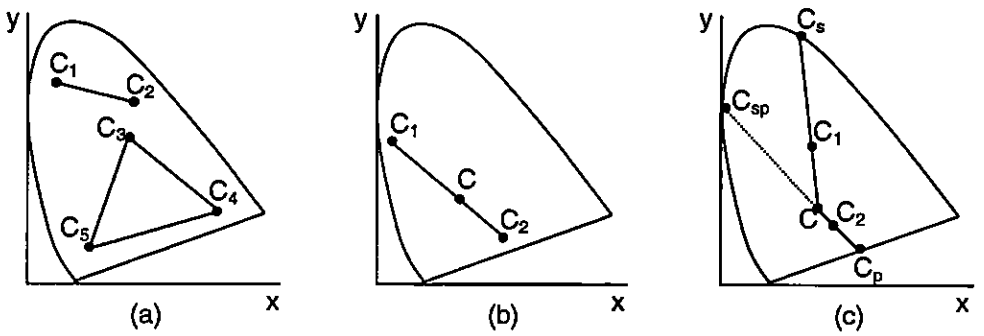


Figura 2.5 – Gamas de cores no diagrama de cromaticidade [Hea94].

Cores complementares são representadas no diagrama como os dois pontos de um segmento de reta que passa pelo ponto C (luz branca). A figura 2.5(b) ilustra essa situação. Quando se mistura quantidades adequadas das cores C_1 e C_2, pode-se obter a luz branca. Para determinar o comprimento de onda dominante de uma cor C_1, figura 2.5(c), traça-se uma reta partindo do ponto C, passando por C_1, até atingir a curva espectral no ponto C_s. A cor C_1 pode então ser representada como uma combinação da luz branca (C) e da cor espectral C_s, pois C_s é o comprimento de onda dominante de C_1. Esse método não funciona na determinação do comprimento de onda dominante para cores que estão na região entre o ponto C e a linha púrpura do diagrama. Nesse caso, ilustrado na figura 2.5(c), traçamos uma reta a partir do ponto C, passando pelo ponto C_2, até o ponto C_p na linha púrpura. Como este ponto não está no espectro visível, a cor correspondente a ele é referida como sendo uma "cor não espectral" e seu comprimento de onda dominante é obtido através do prolongamento da reta que une o ponto C ao C_2, até que ela atinja a curva espectral no ponto C_{sp}, que corresponde então ao comprimento de onda dominante de C_2.

A principal vantagem do CIE XYZ, e de qualquer espaço de cor dele derivado, é a sua completa independência do dispositivo de apresentação utilizado. A principal desvantagem dos espaços baseados no CIE é sua complexidade de implementação, além de não serem muito intuitivos para os usuários [Bou95].

2.7. Modelos de Cores

Modelos de cores representam métodos de descrição de cores. A seguir alguns modelos são apresentados, dente eles um sistema derivado do diagrama de cromaticidade, o CIE Luv, e mais quatro modelos de cores muito mencionados na literatura da área de Computação Gráfica. O RGB (*Red, Green, Blue*) e o CMY (*Cyan, Magenta, Yellow*)são modelos de cores orientados a hardware, isto é, são utilizados em dispositivos de exibição; o primeiro normalmente em monitores e o segundo em impressoras. O HSV (*Hue, Saturation, Value*) e o HLS (*Hue, Luminance, Saturation*) são mais orientados ao usuário, pois fornecem parâmetros mais intuitivos para a definição de cores por um usuário.

2.7.1. Modelo RGB

Como discutido na seção 2.3, o olho humano percebe as cores através da estimulação de três pigmentos visuais na retina. Essa teoria da visão é a base para a apresentação de cores em um monitor de vídeo por meio de três cores primárias (vermelho, verde e azul), utilizando assim, o modelo conhecido como modelo RGB [Hea94]. Esse modelo é usado em monitores CRT e em dispositivos de varredura gráfica [Fol94b].

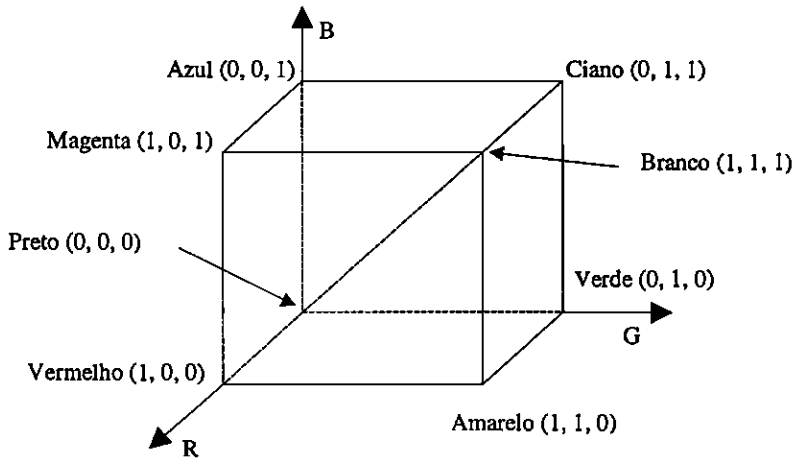


Figura 2.6 - Modelo de cores RGB [Hea94].

As cores primárias do RGB são "primárias aditivas", isto é, as contribuições individuais de cada primária são somadas para produzir o resultado. Podemos representar graficamente o universo de cores desse modelo como um cubo unitário, em que R, G e B são os eixos (figura 2.6). O ponto de origem dos eixos representa a cor preta e o vértice do cubo

de coordenadas (1, 1, 1) representa a cor branca. A diagonal principal do cubo, com quantidades iguais de cada primária, corresponde aos níveis de cinza, intermediários entre o preto e o branco [Fol94b].

2.7.2. Modelo CMY

O modelo CMY é um exemplo de sistema de descrição de cores que utiliza a mistura subtrativa de cores. Os matizes ciano, magenta e amarelo são complementares, respectivamente, aos matizes vermelho, verde e azul. Quando usados em um modelo de cores como filtros para subtrair cores da luz branca, esses matizes são usados chamado "primárias subtrativas". A representação gráfica do universo de cores do modelo CMY é quase idêntica à do modelo RGB, se diferenciando pelo fato de que o branco está na origem e não o preto. As cores são obtidas por meio da remoção ou subtração de cores da luz branca.

O modelo CMY é interessante e muito usado em dispositivos de impressão que depositam pigmentos coloridos sobre uma superfície, tais como as impressoras. Uma superfície revestida com pigmentos ciano não reflete a luz vermelha. Esta é subtraída da luz branca. Como a luz branca é formada pela soma das luzes vermelha, verde e azul, o que restou é a soma da reflexão das luzes verde e azul. Fato semelhante acontece com o magenta, que reflete a luz verde restando a vermelha e a azul; e com o amarelo, que reflete a luz azul restando a vermelha e a verde. Um superfície revestida com o ciano e o amarelo absorve a luz verde e a azul, refletindo apenas a luz verde da luz branca incidente. Uma superfície revestida com as três primárias desse modelo absorve as luzes vermelha, verde e azul e portanto resulta em preto. Essas relações são representadas pela equação 2.8 e estão esquematizadas na figura 2.7.

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.8}$$

O vetor coluna unitário é a representação RGB para o branco e a representação CMY para o preto. Pela equação vê-se claramente que os modelos são complementares e conversão de um modelo para o outro é simples e direta. Essa conversão é muito relevante para o uso de impressoras coloridas e fotocopiadoras.

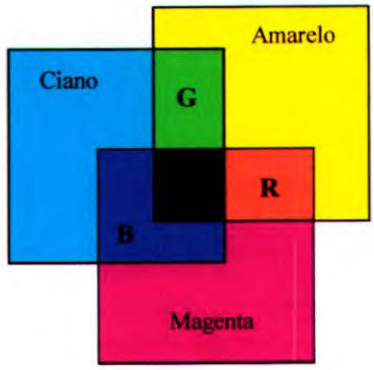


Figura 2.7 – Mistura subtrativa de cores [Fol90].

Um outro modelo de cores, o CMYK, usa o preto (K) como uma quarta primária. Esse modelo é usado em processos de impressão por prensas e algumas copiadores que utilizam quatro cores. Dada uma especificação CMY, as primárias do CMYK são obtidas de acordo com as relações em 2.13.

$$\begin{aligned} K &= \min(C, M, Y) \\ C &= C - K \\ M &= M - K \\ Y &= Y - K \end{aligned} \tag{2.13}$$

2.7.3. Modelo HSV

Os modelos de cores RGB, CMY e outros (como o YIQ, não discutido aqui) utilizados em dispositivos de exibição, apesar de práticos do ponto de vista computacional, são inadequados do ponto de vista do usuário que deseja um sistema capaz de oferecer uma especificação simples e intuitiva de uma determinada cor. Os modelos HSV (*Hue, Saturation* e *Value* - matiz, saturação e valor ou brilho) e HLS (*Hue, Saturation* e *Lightness* – matiz, saturação e luminosidade) por outro lado, são orientados ao usuário, por usarem uma noção mais intuitiva na descrição e definição de cores [Fol94b; Hea94].

A representação gráfica tridimensional do modelo HSV é um cone hexagonal, como mostra a figura 2.8. A base do cone hexagonal corresponde a V=1, que contém as cores com brilho máximo. É importante frisar que o brilho percebido pode não ser o mesmo para todas as cores para as quais V=1.

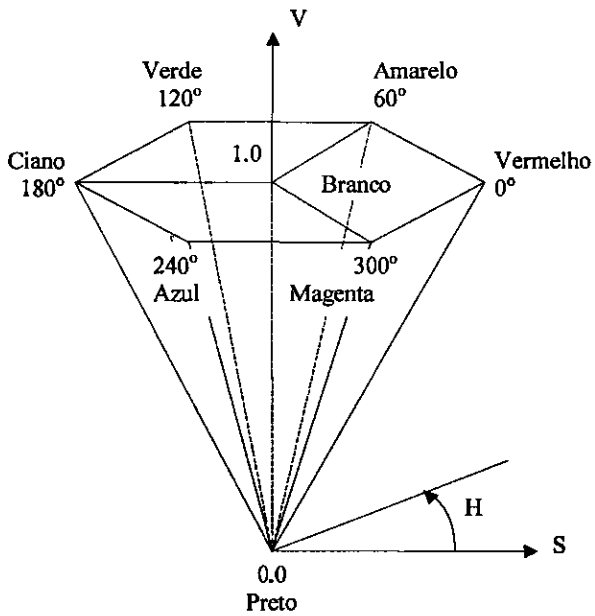


Figura 2.8 - Modelo de cores HSV [Fol94b].

O matiz, ou H, é medido pelo ângulo em torno do eixo central, com o vermelho em 0°, o amarelo em 60°, o verde em 120°, assim por diante (figura 2.8). Cores complementares no cone hexagonal do HSV estão a 180° uma da outra. O valor da saturação (S) varia de 0 (ao longo do eixo central - eixo V) a 1 (nos lados do hexágono). A saturação é medida em relação ao espaço de cores representado pelo modelo, que é um subconjunto de um diagrama de cromaticidade da CIE.

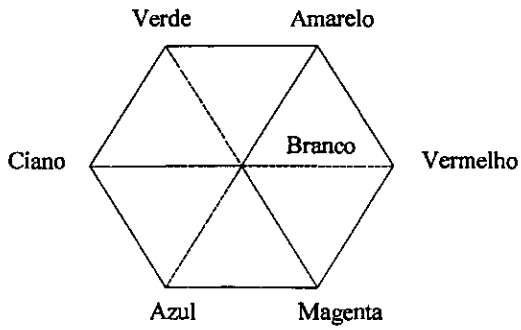


Figura 2.9 - Cubo de cores do RGB visto ao longo da diagonal principal [Fol94b].

O cone possui altura unitária (V = 1), com o ápice na origem. O ponto no cume corresponde ao preto e possui coordenada V = 0. Nesse ponto, os valores H e S são irrelevantes. O ponto (S = 0, V = 1) corresponde ao branco e o matiz (H) não é relevante. Os

valores intermediários de V para S = 0 (na linha central) correspondem aos tons de cinza. Quando S = 0, o valor de H não é relevante, mas, caso contrário, sim.

A base do cone hexagonal do HSV corresponde à projeção do cubo de cores do RGB ao longo da diagonal principal, do ponto correspondente ao branco em direção ao preto (figura 2.9). Cada plano de V constante no espaço HSV corresponde à visão de um subcubo no espaço RGB. A diagonal principal do espaço RGB toma-se o eixo V do espaço HSV. Assim, podemos observar intuitivamente a correspondência entre os modelos RGB e HSV [Fol94b]. Isso facilita o processo de conversão entre os dois modelos por meio de algoritmos simples e facilmente encontrados na literatura como, por exemplo, no livro *Introdution to Computer Graphics*, escrito por Foley e outro, no capítulo 13, seção 13.3.4 [fol94b].

2.7.4. Modelo HLS

O modelo de cores HLS (*Hue, Lightness, Saturation* - matiz, luminosidade ou luminância, saturação) também é baseado em parâmetros intuitivos de cores. Esse modelo tem uma representação em forma de um cone hexagonal duplo, como ilustra a figura 2.10.

O matiz é definido pelo ângulo em tomo do eixo vertical do cone hexagonal, com o vermelho em 0°. As cores ocorrem ao longo do perímetro na mesma ordem do modelo HSV, ou seja, vermelho, amarelo, verde, ciano, azul e magenta. Podemos imaginar o HLS como uma deformação do HSV em que o ponto correspondente a cor branca é puxado para cima formando o cone hexagonal superior a partir do plano V = 1. Assim como no modelo HSV, os matizes complementares estão localizados a 180° de distância e a saturação é medida radialmente a partir do eixo vertical, do ponto 0 ao ponto 1 na borda. Luminosidade (ou luminância) tem valor 0 para preto (na ponta do cone hexagonal inferior) e o valor 1 para branco (na ponta do cone hexagonal superior) [Fol94b].

Os tons de cinza têm parâmetros S = 0, mas os matizes saturados ao máximo estão em S = 1, L = 0,5. O fato de L ter que ser 0,5 para obter as cores mais fortes é uma desvantagem em relação ao modelo HSV, no qual com os valores S = 1 e V = 1 obtemos o mesmo efeito. Contudo, analogamente ao HSV, as cores do plano L = 0,5 não têm o mesmo brilho percebido. Assim, duas cores diferentes com igual brilho percebido geralmente terão valores de L diferentes.

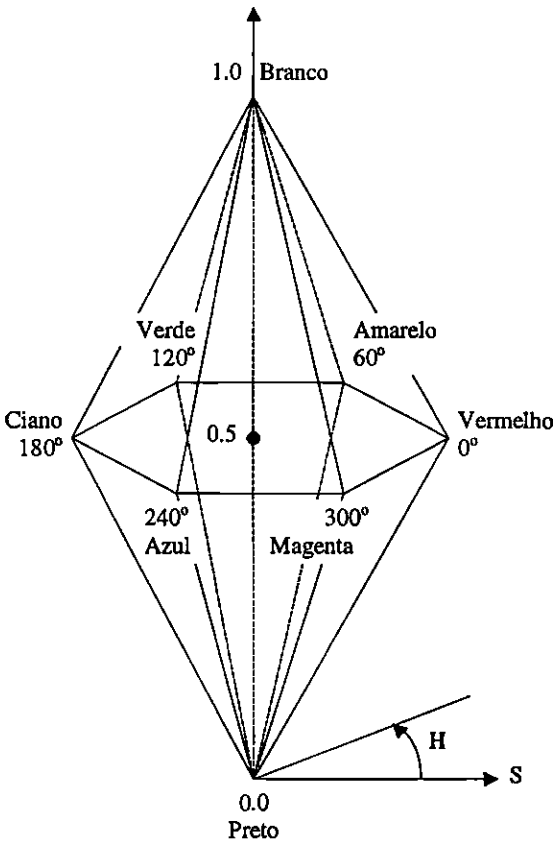


Figura 2.10 - Modelo de cores HLS [Fol90].

2.7.5. Modelo CIE Luv

Existem vários sistemas de descrição de cores da CIE, por exemplo, o CIE LAB, o CIECAM97 e o ZLAB [Fai98], não mencionados aqui. A apresentação do sistema CIE Luv se dá pelo fato de que esse sistema possui uma propriedade muito interessante: ele é perceptualmente uniforme. Isso significa que ele foi projetado para que a variação de um parâmetro de especificação de cor seja percebida como uma variação na mesma proporção da cor resultante. O CIE XYZ não possui essa propriedade, sendo que em 1976 a CIE propôs o modelo CIE Luv, cujas componentes são L*, u* e v*. A componente L* define a luminância, e as componentes u* e v* definem a cromaticidade [Hea96; Taj83; Bou95]. O espaço de cores desse sistema deriva do Cie XYZ como segue:

$$
\begin{aligned}
L^* &= 116\left(\frac{Y}{Y_0}\right)^{\frac{1}{3}} - 16, \text{para} \frac{Y}{Y_0} > 0{,}008856 \\
L^* &= 903{,}3\left(\frac{Y}{Y_0}\right), \text{para} \frac{Y}{Y_0} \leq 0.008856 \\
u^* &= 13L^*(u'-u_0') \\
v^* &= 13L^*(v'-v_0')
\end{aligned}
\tag{2.14}
$$

onde

$$u' = \frac{4X}{X+15Y+3Z} \tag{2.15}$$

$$v' = \frac{9Y}{X+15Y+3Z} \tag{2.16}$$

com u_0' e v_0'tendo a mesma definição de u' e v', porém, aplicados ao ponto de cor branca no espaço.

Cores que estão a uma mesma distância uma das outras no espaço de representação do modelo podem não ser perceptualmente eqüidistantes. Isto normalmente ocorre quando há uma pequena diferença de cor. A diferença de cor que o computador detecta não é proporcional à diferença de cor que o olho humano percebe. Experimentos realizados por MacAdam na década de 40 mostraram que o conjunto formado por todas as cores no diagrama de cromaticidade com a mesma diferença perceptual em relação a uma cor c0 é uma elipse com centro em c_0 (c_0 é o ponto médio da distância entre os dois centros de uma elipse). O tamanho dessa elipse varia para cores em diferentes regiões do diagrama. Por essa razão, a distância perceptual entre duas cores não pode ser medida usando a métrica euclidiana (equação 2.8), onde (ΔE é a diferença perceptual [Gom94]. A métrica perceptual é conhecida na literatura como "métrica jnd" (*just noticeable color difference metric*) [Gom94; Rob85; Wat93]. Quando um sistema de cores é perceptualmente uniforme, a métrica *jnd* é a métrica euclidiana. Em um espaço de cores perceptualmente uniforme, dada uma variação de certa magnitude no valor de um parâmetro, a variação da cor resultante é percebida com igual magnitude [Rob88; Mey80]. Essa falta de equilíbrio perceptual ocorre no CIE XYZ [Hea96; Taj83].

$$\Delta E^* = \sqrt{\Delta L^{*2} + \Delta u^{*2} + \Delta v^{*2}} \tag{2.17}$$

Apesar de ser atrativo pela uniformidade perceptual, o CIE Luv, bem como os demais sistemas derivados do CYE XYZ, não são normalmente usados em aplicações práticas

de computação gráfica. Um dos fatores que contribui para isso é o alto custo computacional dos cálculos de mudança entre esses sistemas e o sistema padrão CIE XYZ [Gom94]. Existem vários outros sistemas de definição de cores que não têm as mesmas propriedades interessantes do CIE XYZ e CIE Luv, como por exemplo os citados aqui, contendo um número bem menor de cores.

2.8. Considerações Finais

O entendimento de como o sistema visual humano processa a informação de cores é de muita importância no estudo do desenvolvimento dos modelos de cores. Essa informação chega ao olhos humano na forma de ondas eletromagnéticas, captadas pelos cones e bastonetes, pré-processada e transmitida ao cérebro. Várias teorias tentam explicar como os sinais luminosos são percebidos pelo sistema visual humano. A colorimetria dá início à padronização das descrições das cores existentes, tendo como conseqüência o desenvolvimento dos modelos de cores. Os modelos de cores são essenciais no uso de uma técnica de vital importância na visualização, o mapeamento por cores.

No capítulo seguinte veremos como se dá o processo de visualização, os aspectos importantes que devem ser levados em consideração e como a utilização do conhecimento sobre o processo de percepção na forma de regras pode ajudar o usuário nesse processo.

Capítulo III

Uma Abordagem Baseada em Regras para a Incorporação de Conhecimento sobre Percepção ao Processo de Visualização

3.1. Considerações Iniciais

A visualização computacional é uma área de pesquisa voltada para o estudo de técnicas para mapear informações em representações gráficas por meio do mapeamento de atributos dos dados em dimensões sensoriais, em geral, visuais. Esse mapeamento é feito usando técnicas da Computação Gráfica, e tem por objetivo ajudar cientistas e outros profissionais a capturar o significado dos seus dados e entender os fenômenos que geram esses dados. Uma visualização de sucesso fornece uma representação que permite ao seu usuário conhecer melhor a estrutura dos dados, ou transmitir aspectos dessa estrutura de forma eficiente. Mesmo em sistemas modernos de visualização, que fornecem ao usuário um controle considerável sobre o processo de mapeamento, pode ser difícil produzir uma visualização efetiva. Isso ocorre devido à grande quantidade de parâmetros a serem estabelecidos, à dificuldade de escolha de

parâmetros adequados e à perda de informação quando da apresentação de imagens 3D em dispositivos 2D. Existem várias propostas que visam ajudar o usuário no processo de criação de uma visualização. Uma delas restringe o conjunto de operações disponíveis ao usuário para um subconjunto de operações que são mais apropriadas aos dados e à tarefa de visualização em questão.

Este capítulo está organizado da seguinte maneira: na seção 3.2 é apresentado o processo de geração de visualizações e o contexto no qual esse processo está inserido, bem como a arquitetura básica dos sistemas de visualização modernos e a técnica de mapeamento por cores. Na seção 3.3 discute-se porque o estudo de aspectos de percepção é importante para gerar visualizações efetivas. A seção 3.4 contém a descrição de algumas abordagens discutidas na literatura para incorporar aspectos de percepção ao processo de visualização. A proposta de Rogowitz e outros é a única descrita com detalhes, visto que o projeto desenvolvido baseia-se nessa proposta. Na seção 3.5 é discutida a utilização dessa abordagem na criação de uma ferramenta para a geração de mapas de cores que foi incorporada ao sistema *IBM Data Explorer*.

3.2. O Processo de Visualização

O usuário é parte ativa no processo de visualização, cujo objetivo é apresentar os dados de uma maneira que ajude o usuário a identificar tendências, características e padrões, gerar hipóteses e associar significados à informação visual [Rog93a].

A figura 3.1 ilustra o processo interativo de geração de uma visualização. O Servidor de Dados fornece os dados, obtidos a partir do resultado de simulações ou medições para o sistema de visualização, no qual esses podem ser processados e estruturas geométricas podem ser computadas, resultando em uma imagem renderizada na tela. O usuário interage com o modelo representado na imagem e, baseado na sua interpretação da representação visual criada, modifica a imagem e/ou modelo para redirecionar os cálculos, redefinir parâmetros para conduzir a simulação, mapear novamente os atributos dos dados para entender melhor sua estrutura, ou criar uma visualização que realce uma característica em particular. Dessa forma, o usuário é o foco do sistema, pois ele controla o processo, e os demais componentes servem para apoiar o processo de interpretação e análise dos dados pelo

usuário [Rog93a]. A interação do usuário com o sistema é de grande importância, pois, por meio dessa interação, as representações criadas são interpretadas, avaliadas e novas representações são geradas. A cada iteração as representações são melhoradas, até que se obtenha uma representação adequada. Nesse contexto, o poder de reconhecimento e a capacidade de tomada de decisão da pessoa dirigem o processo de visualização.

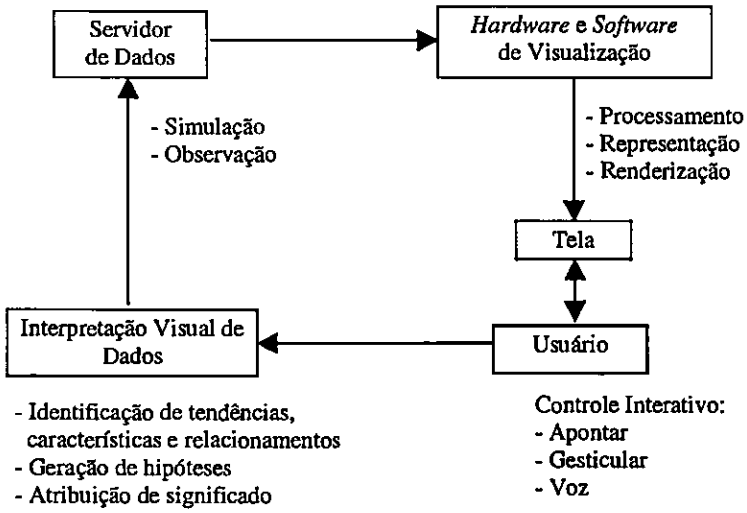


Figura 3.1 - Processo interativo de visualização [Rog93a].

A figura 3.2 ilustra como esse processo é suportado por muitos dos atuais sistemas de visualização. Esses sistemas fornecem muitas ferramentas de processamento (como a aplicação de filtros), de mapeamento (aplicação de mapas de cores e geração de estruturas geométricas, como isolinhas ou isosuperfícies) e de renderização (como a geração da imagem por técnicas de tonalização, ou por *ray-tracing*). Em geral, eles dão ao usuário a liberdade de definir praticamente qualquer seqüência de utilização dessas ferramentas, oferecendo uma enorme variedade de operações que, na verdade, oferecem mais do que o usuário pode aplicar efetivamente [Rog93a; Rog93b].

Os sistemas de visualização atuais são altamente interativos, e dão ao usuário livre controle sobre o mapeamento de atributos dos dados em dimensões visuais. Adicionalmente, o número de dimensões visuais possíveis para uma representação é grande e continua se expandindo. Contudo, toda essa liberdade cria uma infinidade de problemas para o usuário, e pode levá-lo a gerar representações que expressam de forma inadequada a estrutura inerente aos dados, devido a mapeamentos incorretos ou introdução de artefatos

visuais enganosos. Isso pode ocorrer, por exemplo, quando dados contínuos são mapeados em uma representação discreta.

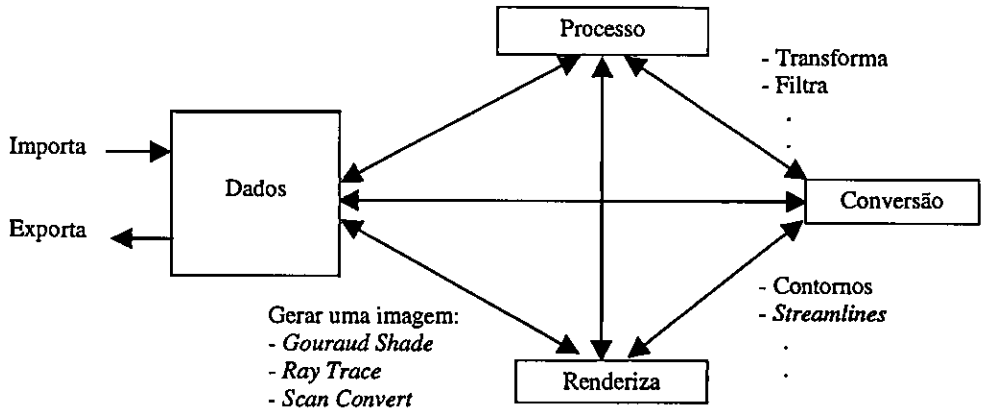


Figura 3.2 - Arquitetura dos sistemas de visualização modernos [Tut98b].

A visualização é também um processo inerentemente iterativo, especialmente para tarefas de exploração, que são tarefas de análise mais detalhada dos dados através da "navegação" e exploração de dados e gráficos. Uma imagem visual é criada, e em seguida modificada e recriada, na esperança de que cada iteração forneça um entendimento melhor da estrutura dos dados, ou uma maneira melhor de transmitir seu significado. Um determinado mapa de cores pode, por exemplo, revelar um relacionamento inesperado entre duas variáveis, que podem ser novamente exploradas examinando esse relacionamento sob outros pontos de vista. Independentemente da estrutura inerente aos dados, o processo de visualização é mais eficiente se for iterativo [Rog93a].

Pelo do estudo da evolução dos sistemas de visualização, observamos que, nos primeiros sistemas, as imagens eram geradas com a utilização de ferramentas de baixo nível denominadas bibliotecas, como será visto no capítulo 4. Essas ferramentas, embora flexíveis e poderosas, são arcaicas, tendo sido desenvolvidas para e por especialistas, e não para usuários finais. Interações são difíceis nesses sistemas, visto que o usuário precisa escrever programas para gerar visualizações.

Os sistemas de visualização mais modernos são abertos e extensíveis (em geral, baseados no paradigma do fluxo de dados ou em uma linguagem de quarta geração), ou fechados como ferramentas integradas (ou seja, dirigidos a dados ou domínios específicos). Esses sistemas fornecem uma interface de alto nível para definir o processo de visualização e

um grande conjunto de ferramentas que tornam as iterações mais rápidas. Nessa classe estão incluídos o AVS (*Application Visualization System*) [AVS95; AVS99] e o *IBM Data Explorer* [IBM96, IBM99].

Embora esses sistemas ofereçam um rico conjunto de transformações, processos, mapas de cores e estratégias de realização, eles muitas vezes não apoiam o usuário no processo de escolher mapeamentos adequados que reproduzam fielmente a estrutura dos dados ou comuniquem adequadamente o objetivo da visualização. Segundo a proposta de Rogowitz e Treinish [Rog93a] a próxima geração de sistemas de visualização fornecerá ajuda ao usuário na forma de regras de visualização. Tais regras incluiriam princípios de percepção humana, cognição e teoria das cores e guiariam o mapeamento de atributos dos dados para dimensões visuais, tornando cada iteração mais efetiva. Dessa forma, os sistemas futuros tendem a embutir algum tipo de ajuda ao usuário, ou na forma de sistemas especialistas que automatizam o processo de gerar as visualizações, ou na forma de regras que ajudam o usuário na seleção de mapeamentos e técnicas adequadas.

3.2.1. Mapeamento por Cores

A técnica de mapeamento por cores é muito usada e de extrema importância em Computação Gráfica, particularmente em Visualização Científica. Trata-se de uma técnica unidimensional que mapeia uma parte da informação (isto é, um valor escalar) em uma especificação de cor. Contudo, a apresentação da informação de cor não está limitada a representações unidimensionais. Muitas vezes usa-se informações de cor mapeadas em objetos 1D, 2D e 3D [Sch97].

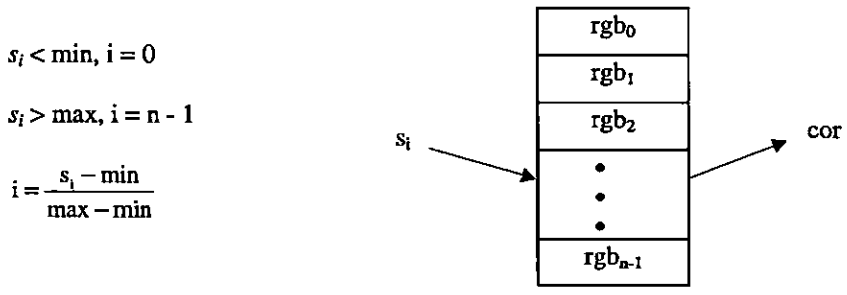


Figura 3.3 - Mapeamento linear de valores escalares através de tabelas de cores [Sch97].

O mapeamento por cores é implementado indexando uma tabela de cores (*Lookup Table*), sendo que os valores escalares são usados como índices para a tabela. Essa tabela consiste de um vetor de cores (especificadas em termos de seus componentes vermelho, verde e azul, ou de primitivas de outros modelos). Associados à tabela existem um valor mínimo e um valor máximo (min, max), e os valores escalares são mapeados no intervalo que varia do mínimo ao máximo. Valores escalares maiores do que o valor máximo do intervalo são mapeados para a cor correspondente ao valor máximo, valores escalares menores que o mínimo são mapeados à cor correspondente ao valor mínimo. Assim, para cada valor escalar s_i, o índice i na tabela de cores com n entradas é obtido como ilustrado na figura 3.3, em caso de um mapeamento linear.

O ponto chave para a utilização de mapas de cores na visualização de dados escalares é a escolha adequada das cores que compõem a tabela de cores. O uso cuidadoso de cores pode realçar características importantes de um conjunto de dados. Porém, uma tabela de cores inadequada pode destacar detalhes irrelevantes, ou mesmo criar artefatos visuais devido: a interações inesperadas entre os dados, a escolha inadequada de cores e/ou à própria limitação do sistema visual humano [Sch97].

A maioria dos sistemas de visualização atuais fornece uma tabela de cores padrão (normalmente a tabela de cores do arco-íris) e uma ferramenta para a criação de tabelas personalizadas. A tabela resultante, no entanto, pode levar o usuário a inferir estruturas inexistentes nos dados ou a perder detalhes, pois, estes podem ter sido agrupados em uma região mapeada com uma única cor [Tut98].

Um outro elemento importante para a visualização é a sua qualidade estética. Boas visualizações representam dados mantendo um certo equilíbrio entre a comunicação efetiva da informação e uma representação esteticamente agradável. Muitas representações sacrificam as informações para obterem uma boa imagem e nem sempre significativa. Aumentar o grau de conforto e atrair a atenção do observador humano para detalhes mais relevantes melhora a eficácia da comunicação [Sch97].

3.3. Percepção e Visualização

Com o aumento da complexidade das visualizações, devido não apenas à grande quantidade de dados, mas também à expansão da aplicação da visualização em várias áreas, a necessidade

de se considerar aspectos de percepção no processo de representação de dados tornou-se evidente. No projeto de uma visualização, conhecimentos sobre os mecanismos fisiológico, psicológico e psicofísico do sistema visual humano, bem como diferenças culturais e experiências anteriores dos usuários deveriam ser considerados [Lan91; Tul95]. Essa preocupação com os aspectos de percepção pode estar voltada para a maneira como o usuário interpreta o conjunto de dados, ou para as propriedades da imagem (tais como cor, forma e posição) que facilitam a interpretação. Observando esses aspectos, representações melhores podem ser geradas pelos usuários, evitando ambigüidades e confusão visual. Representações que desviam o olho do observador para características não importantes podem causar sobrecarga do canal visual [Rhe95].

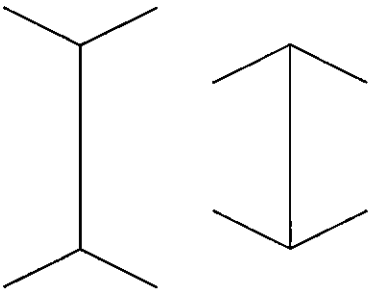


Figura 3.4 - Ilusão de Muller-Lyer [Tul95].

Nem sempre o que o usuário vê em uma imagem (representação visual dos dados) é o que os dados realmente representam. Isso pode ser observado na figura 3.4, que ilustra a ilusão de Muller-Lyer. As duas linhas têm o mesmo tamanho, porém a da direita parece ser menor [Dav75 apud Tul95][1].

Diferentes representações dos dados podem revelar características diferentes. Se o objetivo de uma visualização é mostrar um tipo específico de variação, o projetista da visualização deve selecionar a melhor representação para mostrar esse tipo de variação. Se o objetivo é exploratório, então ele deve assegurar que a representação revele toda a variação inerente aos dados [Tul95]. O uso de mapeamentos redundantes pode ser efetivo: visualizações que representam os valores dos dados usando múltiplos parâmetros de apresentação têm maior potencial de eficácia para retratar os dados do que visualizações que mapeiam cada variável para um único parâmetro [Rhe95].

[1] Davidoff, J. – "Differences in Visual Perception", Crosby Lockwood Staples, london, 1975. Apud Tulloch, K. – "Data Visualization", School of Geografy, University of New South Wales, 1995.

Tukey [Tuk90 apud Tul95][2] apresenta alguns princípios gerais que deveriam ser observados na criação de uma representação gráfica:

- uso de um critério visual, uma escala, para julgar a importância da variação observada;
- cuidado na escolha e expressão de escalas é essencial;
- objetos são percebidos em relação aos outros objetos ao seu redor;
- linhas retas são mais fáceis de perceber do que linhas curvas;
- linhas horizontais são mais fáceis de perceber do que as oblíquas;
- objetos que estão enfatizados juntos são mais fáceis de serem comparados do que os que estão separados;
- objetos de igual importância deveriam ter igual impacto visual;
- material irrelevante pode interferir seriamente com o gráfico;
- movimento é mais efetivo para expressar profundidade tridimensional do que estereopse ou perspectiva;
- os princípios de boas apresentações gráficas são, muitas vezes, conflitantes, exigindo um compromisso entre eles.

Não é necessário retratar a representação com grande realismo, mas o uso de elementos familiares aos usuários ajudam a estabelecer associações com a visualização gerada [Tul95; Rhe95].

Pesquisas têm sido desenvolvidas, com sucesso, para incorporar aspectos de percepção como parte integrante dos sistemas de visualização. Um exemplo é a proposta de Rogowitz e Treinish [Rog93a; Rog93b] de construção de uma arquitetura para incorporar regras perceptuais ao processo de gerar uma visualização. O grande problema da geração automática de representações que consideram aspectos perceptuais é que muitos aspectos da relação do usuário com objetos e imagens de visualização são fracamente definidos, e variam em função de resultados de interações e evolução. Assim, não existe um conjunto fixo de

[2] Tukey, P. - "Visual Perception and Scientific Data Display" in Rueter, L.; Tukey, P.; Maloney, L.; Pani, J.; Smith, S. - "Human Perception and Visualization" in (ed.) Kaufman, A - **Proceedings of the First IEEE Conference on Visualization**, IEEE Computer Society Press, California, 1990. Apud Tulloch, K - "Data Visualization", School of Geografy, University of New South Wales, 1995.

regras para obter uma representação ótima. Mesmo com a definição de várias regras, a percepção humana continua sendo a ferramenta mais poderosa no reconhecimento de padrões. Se uma visualização foi criada sem uma ferramenta que incorpore conceitos de percepção e apresentação efetiva, o uso dessas regras pelo próprio usuário é apropriado.

A utilização de cor em visualização é extremamente valiosa, mas pode ser considerada como uma área particularmente difícil. Embora experimentemos a sensação de cor continuamente essa experiência é, invariavelmente, passiva e geralmente, temos pouco ou nenhum treinamento na manipulação ativa de cores.

Em geral, as dimensões ou eixos de cores não são ortogonais, ou seja, mudar um parâmetro de definição normalmente provoca um deslocamento não linear de percepção nos demais. Isso faz com que o processo de visualização varie de uma pessoa para outra. Considerações especiais podem ser necessárias para usuários com anomalias visuais. Algumas pesquisas na derivação e manipulação de espaços de cores perceptualmente uniformes já estão disponíveis e serão beneficiárias neste ponto. Espaços de cores perceptualmente uniformes também podem ter aplicações na armazenagem, transporte e reprodução de imagens entre diferentes plataformas [Min95].

A percepção de cor é um processo claramente subjetivo e as pessoas diferem muito na sua habilidade de manipular cores efetivamente. Alguns problemas são: a incidência de visão defeituosa das cores entre os usuários finais, a facilidade de sobrecarregar o sistema visual com muitas cores, e a falta de experiência prévia no uso de cores. A chave para o uso efetivo de espaços perceptuais de cores, na prática, é fornecer uma interface apropriada que permita a visualização de um espaço de cores em um dispositivo de apresentação [Rob88]. Um estudo mais completo de uso de espaços perceptuais de cores pode ser encontrado os trabalhos de Robertson e de Meyer [Rob88; Mey80].

Um método para apresentar informação de formar visual é o mapeamento de um ou mais atributos da informação para cor. Assim, cada valor ou intervalo de valores é associado a uma cor. Uma abordagem comum é adotar a escala do arco-íris, cuja associação é baseada na posição da cor no espectro visível (violeta-azul para valores baixos e vermelho para valores altos). Essa escala é baseada no contraste de matiz, sendo que alguns estudos indicaram que uma escala baseada no contraste do brilho é mais efetiva. O brilho é baseado na intensidade luminosa percebida, ao invés da intensidade física, o que é normalmente conhecido como luminância [Owe96].

Se a intenção é analisar precisamente o caráter numérico dos dados ou o modelo a eles inerente, uma escala baseada apenas em variações do matiz será inadequada ou enganosa, visto que variações apenas do matiz não refletem precisamente variações na magnitude. Nesse caso, uma escala baseada variação do brilho ou da saturação pode ser mais apropriada. Por outro lado, se a preocupação é identificar tendências e padrões, ao invés de valores explícitos, uma escala com a rápida variação que pode ser alcançada com mudança de matiz pode ser ideal.

Owen [Owe96] afirma que estudos mostram que objetos com cores brilhantes em um fundo preto parecem maiores do que os mesmos objetos desenhados com cores escuras em um fundo claro. Isso tem duas implicações: é necessário cuidado com a quantidade de dados visuais apresentados, e objetos pequenos podem se tornar mais visíveis se forem coloridos e brilhantes contra um fundo escuro. A associação de uma escala que torna uma certa região mais brilhante pode melhorar a percepção de profundidade dessa região. Esta é outra razão para escolher uma escala baseada no contraste de brilho.

Outro método usado para aumentar o contraste de brilho é usar técnicas de morfologia matemática para "raspar" a superfície. O efeito é o de se criar uma região muito estreita de valores de dados iguais a zero em torno de cada superfície. Isso causa uma grande graduação na densidade da luz, que aumenta o contraste. Escolhas diferentes de mapeamento de cores também podem ser usadas para mostrar ou esconder diferentes aspectos de um conjunto de dados [Owe96].

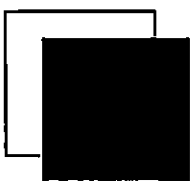


Figura 3.5 - Exemplo de oclusão [Owe96].

Nossa experiência afeta o modo como percebemos uma imagem. Por exemplo, associamos objetos que são fisicamente maiores com valores de dados mais altos. Assim, se um objeto está na frente de outro, como determinado pelas sensações de profundidade, então

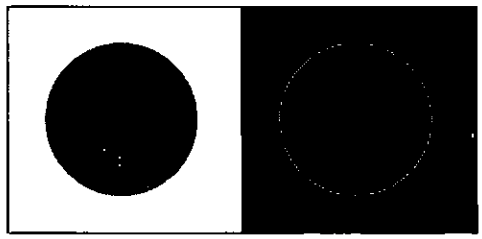


Figura 3.6 - Efeito do contraste de brilho na percepção do matiz [Owe96].

ele deve corresponder a um valor de dados mais alto. Uma sensação de profundidade é dada pela oclusão, como na figura 3.5. Como o quadrado preto sobrepõe-se ao quadrado branco consideramos que o quadrado preto está à frente, e tem um valor mais alto.

Outro problema é a introdução de descontinuidade aparente em dados contínuos pelo mapeamento por cores. O contraste de brilho afeta a percepção do matiz. Na figura 3.6, o círculo cinza à esquerda parece ser mais escuro do que o círculo à direita, mas eles são idênticos.

3.4. Visualização Baseada em Regras

Há várias propostas para melhorar o processo de geração de visualizações, seja fornecendo regras que auxiliam o gerador na escolha das técnicas e parâmetros adequados, sou automatizando o processo de geração. Uma dessas abordagens é sugerida por Senay e Ignatius [Sen94, Owe96], que propõem a incorporação de um sistema baseado em conhecimento aos sistemas de visualização. Nessa abordagem, o conhecimento é embutido no sistema na forma de um "sistema especialista" que seria utilizado para automatizar o processo de gerar visualizações. O VISTA (*Visualization Tool Assistant*) é um sistema especialista que ajuda os usuários a aplicar uma técnica de visualização apropriada para um conjunto de dados, e permite que eles modifiquem interativamente o projeto da visualização. Através da especificação do conjunto de dados pelo usuário, o sistema gera uma estratégia de visualização apropriada para esse conjunto e renderiza a imagem resultante.

Foley e outros [Fol94a; Owe96] também propõem a automatização do processo de geração de visualizações. O *Glyphmaker* é um sistema que permite ao usuário construir e organizar *glyphs* para representar dados multivariados. Os *glyphs* são objetos gráficos cujos atributos visuais (posição, tamanho, orientação, etc.) são associados a atributos dos dados. O sistema oferece uma biblioteca de *glyphs*, e permite que os usuários definam os seus próprios

glyphs interativamente. O sistema permite também que o usuário associe interativamente atributos de variáveis de interesse a atributos do *glyph* para apoiar a exploração dos padrões e relações contidas nos dados.

Uma outra abordagem proposta por Robertson [Rob91; Owe96] se baseia no paradigma da cena natural (NSP - *Natural Scene Paradigm*). Esse paradigma procura representar as variáveis através de propriedades de objetos que podem ser imediatamente reconhecidas. Segundo o autor, é importante que o observador consiga construir um modelo mental a partir da visualização, cujos atributos visuais representem atributos dos dados de uma maneira bem definida. O NSP usa modelos claros e facilmente compreensíveis como estruturas e cenas 3D, representa variáveis de dados através de propriedades reconhecíveis dos objetos ou cenas, e usa técnicas gráficas de simulação de cenas para induzir o reconhecimento de propriedades pelo observador.

Dentre várias abordagens, destacamos a proposta por Rogowitz e outros [Rog93a; Rog93b]. Essa abordagem é descrita com mais detalhes, pois fornece a base do projeto de pesquisa a ser desenvolvido. Esses autores propõem uma arquitetura baseada em regras que ajuda o usuário no processo de geração de uma visualização restringindo as operações possíveis sobre o seu conjunto de dados, denominada PRAVDA (*Perceptual Rule-based Architecture for Visualizing Data Accurately*). Os autores descrevem a natureza das regras que regem o processo e a arquitetura de um sistema de visualização baseado em regras genérico.

Os sistemas de visualização atuais suportam uma grande variedade de escolhas para mapear, manipular e renderizar dados. Tipicamente, os mecanismos para fazer essas seleções são baseados na estrutura inerente aos dados. Os autores sugerem que: 1) as operações de visualização sejam dirigidas em termos de operações em representações de mais alto nível dos dados, denominada metadados; e 2) a introdução da noção de regras de visualização, que o usuário invoca para obter ajuda no processo de criar uma visualização. A ajuda fornecida pelas regras seria baseada em princípios da visão humana, cognição e percepção visual codificados nas mesmas. Se selecionadas, as regras restringiriam os possíveis mapeamentos dos atributos dos dados em dimensões visuais. Os metadados forneceriam uma caracterização de mais alto nível do dados, cujas informações seriam utilizadas como entrada pelas regras, além de fornecer ao usuário uma representação mais intuitiva das mesmas.

O propósito das regras de visualização é assegurar que estruturas dos dados sejam representadas com fidelidade na imagem, e que artefatos perceptuais não sejam erroneamente interpretados como características dos dados [Rog93b]. Para alcançar esse objetivo, os autores acrescentam que é importante entender como o observador humano processa as variações espaciais de luminância e cor, e como variações na representação física de luminância e cor afetam a sua interpretação.

São definidas duas classes de regras. Na classe I estão incluídas as regras para a criação de visualizações que representem fielmente a estrutura dos dados, ou seja, fazem um mapeamento isomórfico de atributos dos dados em dimensões perceptuais de forma a assegurar que as características visuais como a cor, o tamanho e a forma dos dados não variem [Rog93a; Rog93b]. A classe II contém regras que transformam intencionalmente a estrutura dos dados de forma a realçar características dos mesmos.

Essas regras podem considerar ainda a tarefa a ser apoiada pela visualização. As tarefas foram divididas em duas classes, sendo que a primeira inclui tarefas de exploração e análise, e a segunda consiste de tarefas de apresentação. Nas tarefas de exploração e análise é importante manter uma representação fiel da estrutura dos dados, minimizando os artefatos visuais. Nas tarefas de apresentação o objetivo é, muitas vezes, transmitir informações de forma não ambígua e chamar a atenção para características importantes nos dados. Um exemplo citado por Rogowitz e Treinish [Rog93a] menciona a aplicação de tarefas de enfatização em imagens obtidas por sensoriamento remoto, ajudando na identificação de regiões de cultivo de diferentes culturas. Essas regiões precisam ser mostradas sem distorção do contexto dos dados. Já em um mapeamento isomórfico, uma duplicação no intervalo de variação dos dados (*dynamic range*), por exemplo, deve ser percebida como uma duplicação na intensidade da dimensão visual associada.

Segundo Rogowitz e Treinish [Rog95], para representar precisamente a estrutura contida nos dados é importante entender a relação entre a estrutura dos dados e a representação visual. Para dados nominais (dados que pertencem a um certo conjunto de itens, por exemplo [Minas Gerais, São Paulo, Rio de janeiro, Paraná]), os objetos devem ser distinguíveis, mas como os dados não estão ordenados, não deveria haver uma ordenação perceptual na representação visual associada. Para dados ordinais (relacionados entre si segundo uma ordem), objetos devem ser perceptualmente discrimináveis, e a ordenação entre eles deveria ser aparente na representação. Para dados do tipo intervalo, incrementos iguais

nos valores dos dados devem aparecer como incrementos iguais na magnitude percebida da representação. Em dados do tipo razão, os valores aumentam e diminuem monotonicamente em torno de um zero ou outro valor de referência, que deveria ser preservado na representação dos dados.

Em muitos sistemas de visualização o mapa de cores *default* pinta de azul o menor valor associado a uma variável, e de vermelho o maior, com um mapeamento linear do intervalo dos dados entre o azul e o vermelho. Esse mapa de cores não produz uma variação contínua do matiz ao longo do intervalo de dados, ao contrário, produz uma representação em faixas de cores [Lev92]. Além disso, mapear uma variável contínua utilizando um mapa de cores perceptualmente discreto pode introduzir artefatos visuais na imagem resultante.

Essa arquitetura proposta está construída em um modelo funcional de dados com organizações lógicas e físicas, e um esquema para relacioná-las [Rog93b]. As estruturas de dados do modelo orientam as operações de visualização, que são restringidas pelas regras de visualização. A regra de visualização apropriada é selecionada com base nos metadados. Metadados relevantes ao processo de visualização podem incluir estatísticas (como intervalo de variação e desvio padrão), se os dados são discretos ou contínuos e freqüência espacial.

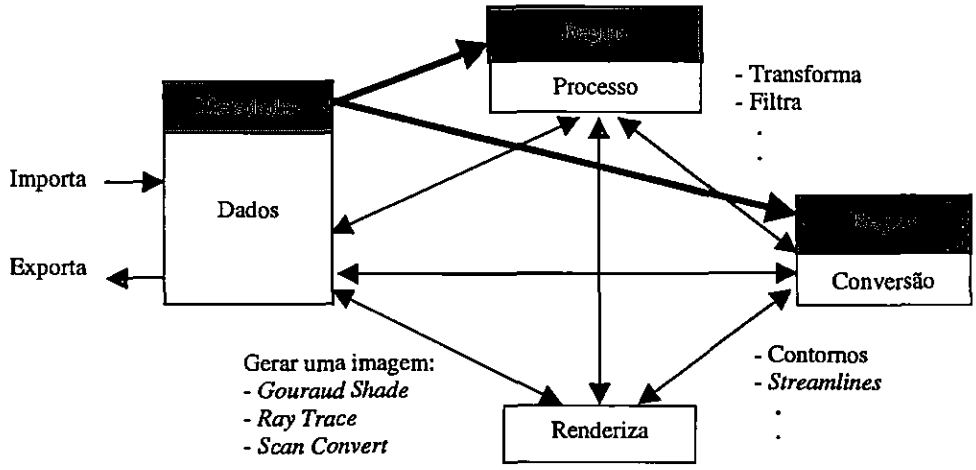


Figura 3.7 - Arquitetura dos sistemas de visualização modernos com a inclusão de regras [Rog93b].

A figura 3.7 mostra a adição de uma camada complementar (ilustrada em cinza) aos sistemas atuais (figura 3.2) para auxiliar o usuário na aplicação das ferramentas disponíveis. Essa camada corresponde à inclusão dos metadados e do conjunto de regras adotadas pelo sistema. Com a inclusão dessa camada complementar, que não altera o processo

de geração das visualizações, os metadados são transmitidos para as regras de visualização, que restringem o conjunto de operações disponibilizadas ao usuário. Assim, a partir de um conjunto de opções mais restritas, porém mais adequadas, o usuário pode selecionar uma estratégia efetiva para visualizar seus dados.

A visualização baseada em regras aumenta o nível de abstração do processo de visualização de duas maneiras. Primeiro, o usuário interage com a ferramenta de visualização em termos de uma caracterização de alto nível dos dados. Segundo, essa interação é mediada por regras empíricas baseadas na visão humana, em aspectos de cognição e teoria das cores, que guiam o usuário na seleção de operações de visualização apropriadas.

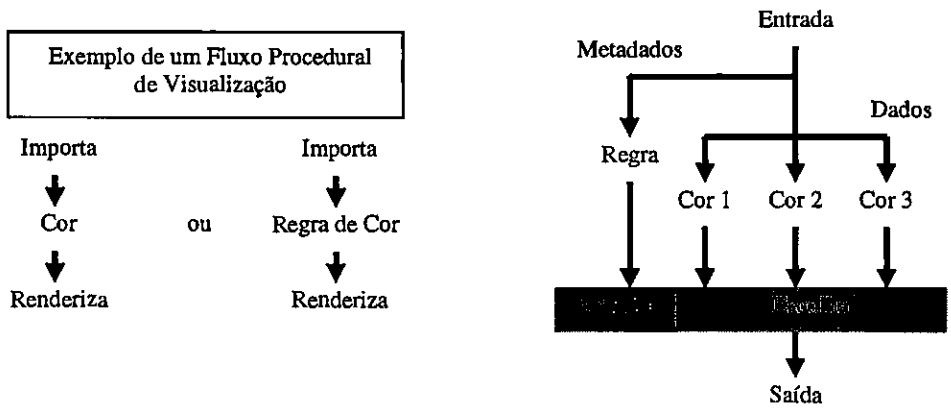


Figura 3.8 - Arquitetura de uma operação baseada em regras [Rog93b].

Nessa arquitetura, cada regra consiste de uma operação convencional restrita, dirigida à estrutura dos dados; ou seja, as regras são aplicadas em uma base de operações e o usuário é livre para escolher entre as operações baseadas em regras ou as tradicionais. A figura 3.8 mostra as operações tradicionais e as baseadas em regras em uma aplicação de mapeamento de cores. O processo é o mesmo para a seleção do mapa de cores *default* ou o baseado em regras. Os dados são importados, uma operação de mapeamento de cores é aplicada e a imagem é renderizada. O lado direito ilustra como a regra está implementada. A operação baseada em regras fornece vários candidatos a mapas de cores, cada qual apropriado para dados com certas características. Os metadados, que contêm informação sobre essas características relevantes, seguem para a regra, que seleciona o mapa apropriado. Os metadados podem ser fornecidos juntamente com os dados (por exemplo, se a variável é contínua ou discreta) ou podem ser computados se necessário pelas regras (como o intervalo

de variação - *dynamic range*) [Rog93a, Rog93b].

3.5. PRAVDAColor

A partir da arquitetura baseada em regras discutida na seção anterior, Bergman e outros [Ber95; Rog95] apresentam os princípios gerais para implementar tal grupo de operações de visualização baseada em regras para a seleção de mapas de cores. A arquitetura foi utilizada para criar uma ferramenta que foi incorporada ao *IBM Data Explorer*. Através de um painel de controle a ferramenta computa os metadados relativos à freqüência espacial e ao tipo dos dados (ordinal, nominal, intervalo ou razão) e pede ao usuário para fornecer a tarefa de visualização (isomórfica, segmentação ou enfatização). Os dados importados para o DX (*Data Explorer*) seguem para um módulo denominado *PRAVDAColor*.

Os dados (escalares) são classificados quanto ao seu tipo (intervalo ou razão) e a sua freqüência espacial (alta ou baixa). As variáveis contínuas são mapeadas levando em consideração as características espaciais dos dados, o que permite a seleção de uma representação visual que mapeie fielmente as estruturas contidas nos dados. Essas características devem ser consideradas porque a sensibilidade visual humana para a variação espacial é diferenciada para os mecanismos de matiz e de luminância. O mecanismo de luminância é adequado para freqüências espaciais altas. Por isso, os mapas de cores que incluem variação no componente de luminância podem representar adequadamente informação com alta freqüência espacial. O mecanismo de variação no matiz é adequado para dados com freqüências espaciais mais baixas. Logo, mapas de cores baseados em saturação, que apresentam variações na magnitude do matiz, são inadequados para transmitir informação de alta freqüência espacial, mas são adequados para representar variações espaciais em escalas maiores [Ber95].

A figura 3.9 descreve a classificação utilizada para a geração de mapas de cores, em que L indica luminância, M indica matiz e S indica saturação. A coluna mais à esquerda contém a classificação do tipo de dado, que pode ser de intervalo ou razão; e pode apresentar alta ou baixa freqüência espacial. As demais colunas contêm recomendações para a criação de mapas de cores apropriados tanto para o tipo dos dados como para a tarefa

Tipo de Dado	Freqüência Espacial	Tarefa de Representação		
		Isomorfismo	**Segmentação**	**Enfatização**
Razão (zero verdadeiro)	Baixa	L: uniforme M: pares oponentes ou complementares S: aumentando monotonicamente a partir do cinza	- Número par de segmentos - Muitos segmentos	- Variação maior para características enfatizadas
	Alta	L: aumentando monotonamente M: pares oponentes ou complementares S: aumentando monotonicamente a partir do cinza	- Número par de segmentos - Menos segmentos	- Variação menor para características enfatizadas
Intervalo	Baixa	L: uniforme M: pares oponentes S: aumentando monotonicamente a partir do cinza	- Muitos segmentos	- Variação maior para características enfatizadas
	Alta	L: aumentando monotonamente M: variação pequena ou uniforme de matiz S: diminuindo monotonamente	- Menos segmentos	- Variação menor para características enfatizadas
Ordinal	Baixa	L: uniforme M: variação segundo o círculo de matizes S: uniforme	- Menor número de segmentos	- Maior luminância na área enfatizada
	Alta	L: aumentando monotonicamente M: variação segundo o círculo de matizes S: uniforme	-Maior número de segmentos	- Maior saturação na área enfatizada
Nominal	Baixa / Alta	L: uniforme M: variação segundo o círculo de matizes S: uniforme	- Menos do que 7 segmentos	- Maior luminância ou saturação na área enfatizada

Legenda: L - Luminância; M - Matiz, S – Saturação.

Figura 3.9 - Classificação de mapas de cores baseada no tipo dos dados, tarefa de representação e princípios de percepção [Ber95].

de representação. Cada célula contém as características dos mapas de cores que, segundo os autores, atendem aos requisitos da tarefa considerando o tipo dos dados.

Se os metadados indicam que os dados produzirão variações de alta freqüência espacial, então as regras oferecem ao usuário mapas de cores que contêm um componente de luminância variando monotonicamente. Nesses mapas estão incluídos mapas em escala de cinza e também mapas que variam na saturação (ou seja, mapas do verde escuro ao verde claro, do ciano escuro ao lavanda claro) [Berg95].

Se nos dados predominam freqüências espaciais baixas, a informação sobre sua variação espacial é transmitida com mais eficácia pelo canal matiz, assim as regras ofereceriam ao usuário mapas de cores com variações na saturação de um matiz ou variações na saturação de dois matizes, tal como o par de cores oponentes azul/amarelo (azul saturado, através do cinza isoluminante, até o amarelo saturado). Dados contendo pequenos detalhes e variações graduais através do espaço poderiam ser mapeados para mapas de cores que combinam essas características, tal como verde muito saturado para verde pouco saturado.

O *PRAVDAColor* consiste de dois componentes principais. O primeiro é uma macro que seleciona o conjunto de mapas de cores a ser apresentado ao usuário com base nas características dos dados e na tarefa de representação especificada pelo usuário. Essa macro, chamada *ColorMapLookup*, examina duas características dos dados: freqüência espacial e presença ou ausência de um zero (*zero-crossing*). Uma análise simplificada (e aproximada) da freqüência espacial, que consiste em computar o desvio padrão normalizado da diferença entre a malha de dados originais e uma malha filtrada por passa-baixa é usada para classificar os dados como sendo de alta ou baixa freqüência espacial. Quando o desvio padrão normalizado é menor ou igual a 0.1 os dados são considerados de baixa freqüência (essa decisão foi baseada em resultados empíricos).

Usando os valores computados e a tarefa de representação, o *ColorMapLookup* computa um índice em uma tabela (*lookup*), o qual é usado para determinar um conjunto de mapas de cores a ser lido na memória. Se os dados contêm um *zero-crossing* (isto é, os valores mínimo e máximo têm sinais diferentes), são considerados do tipo razão. Caso contrário, são considerados do tipo intervalo. O segundo componente do *PRAVDAColor* é um seletor/apresentador de mapa de cores interativo. Esse componente está implementado como um módulo no DX, denominado *ColorMapPick*, escrito em C usando Motific/Xlib. O *ColorMapPick* permite ao usuário selecionar interativamente seqüências de mapas de cores e

também alterar dinamicamente a associação dos valores do mapa de cores aos valores dos dados.

3.6. Considerações Finais

Neste capítulo apresentou-se o processo de geração de visualização no qual o usuário é parte ativa e essencial para a obtenção de visualizações bem sucedidas, bem como a técnica de mapeamento por cores que permite a associação entre cores e valores dos dados. A necessidade de incorporar aspectos de percepção nesse processo se tornou evidente com a evolução dos recursos computacionais.

Várias abordagens têm sido propostas por vários pesquisadores com o intuito de ajudar o usuário durante o processo de geração de uma visualização. Dentre elas, destacamos a arquitetura PRAVDA, na qual o usuário pode selecionar regras que o auxiliam no processo de mapear atributos dos dados em representações visuais. Essas regras selecionam as operações de visualização apropriadas de acordo com os metadados, que descrevem características de alto nível dos dados e são baseadas em princípios de percepção, cognição e teoria das cores.

A arquitetura PRAVDA foi utilizada para construir uma ferramenta que apoia a operação de mapeamento por cores. O módulo *PRAVDAColor* possui um conjunto de regras que seleciona o mapa de cores mais adequado ao tipo dos dados e ao tipo de tarefa de visualização do usuário. A proposta da arquitetura PRAVDA e o módulo *PRAVDAColor* é ajudar o usuário na escolha de mapeamento dos atributos dos dados de forma que as representações geradas comuniquem adequadamente o conteúdo dos dados. Essa arquitetura, assim como o módulo de mapeamento por cores, serviram de base para o desenvolvimento de um projeto de mestrado [Tut98b] que foi estendido como parte deste trabalho.

O próximo capítulo apresenta alguns aspectos de interação que são relevantes em sistemas de visualização, a classificação dos sistemas existentes e a descrição de três deles, com ênfase no VTK que é utilizado neste trabalho.

Capítulo IV

Interfaces em Sistemas de Visualização

4.1. Considerações Iniciais

A interface de um sistema de visualização, como a de qualquer outro tipo de sistema, é o seu "convite de entrada". É ela que suporta a interação com o usuário, sendo responsável por apoiar a execução de tarefas e pela apresentação dos resultados. Se a interface não for adequada, eficiente e efetiva, o sistema terá pouca utilidade, mesmo que ele seja eficiente e ofereça bons recursos ao usuário. As interfaces dos sistemas de visualização evoluíram em conjunto com a arquitetura desses sistemas. Inicialmente, elas eram baseadas em linguagens de comando, tendo evoluído para interfaces gráficas e de programação visual que não exigem a escrita de *scripts* a não ser quando da criação de novas funções.

Os estudos de aspectos relevantes em interfaces com o usuário em sistemas de visualização apontaram a necessidade de incorporar aspectos de percepção, cognição e teoria das cores a essas interfaces. Isso porque a quantidade de informações apresentadas em uma visualização é muito grande. Caso não haja uma maneira de organizar e estruturar adequadamente essas informações, elas podem confundir o usuário e conduzi-lo a

interpretações incorretas.

Neste capítulo são apresentadas na seção 4.2 uma arquitetura genérica de sistemas de visualização e uma classificação desses sistemas baseada na arquitetura e no tipo de interface oferecida. Na seção 4.3 discute-se os aspectos de interação relevantes em uma interface, dentre os quais aspectos cognitivos, perceptuais e relacionados ao uso de cores. Na seção 4.4 são discutidos três sistemas de visualização atuais: o *IRIS Explorer* e o *IBM Open Visualization Data Explorer* (OpenDX), enfatizando suas interfaces com o usuário; e o *Visualization Toolkit* (VTK) que foi utilizado neste trabalho. Na seção 4.5 são apresentados os comentários finais.

4.2. Arquitetura dos Sistemas de Visualização

A arquitetura de um sistema de visualização descreve os componentes do sistema e a relação entre eles. A arquitetura de um sistema de visualização genérico, segundo Gallagher [Gal95], é mostrada na figura 4.1.

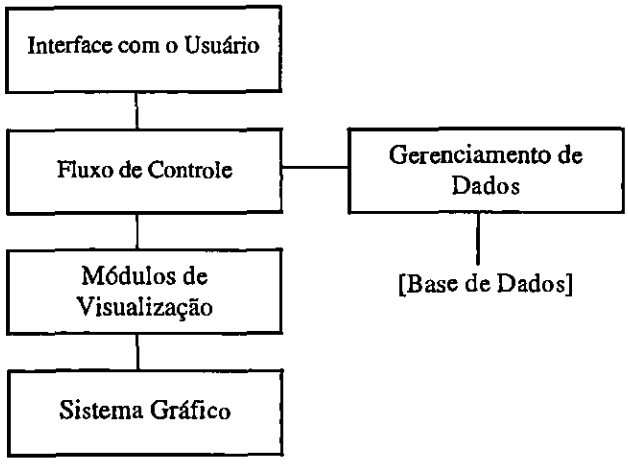


Figura 4.1 - Arquitetura de um sistema de visualização genérico [Gal95].

O componente de Interface com o Usuário define o mecanismo pelo qual o usuário se comunica com o sistema. Isso pode ocorrer na forma de uma linguagem de comandos ou de uma interface gráfica, como discutido de forma mais abrangente na seção 4.3.

O componente de Gerenciamento de Dados abrange os mecanismos que importam os dados de fontes extemas para o sistema, e os gerenciam intemamente. Ele inclui o acesso às base de dados e o gerenciamento da memória intema.

Um Módulo de Visualização é um conjunto de componentes de software que utiliza uma técnica específica de visualização, tal como a extração de isolinhas de uma superfície ou extração de uma isosuperfície de um volume. Os componentes desse módulo são mostrados na figura 4.2. O exemplo mais simples de um módulo desse tipo seria uma sub-rotina em alguma linguagem de programação, como uma função em C. As entradas são os argumentos, o corpo de computação é o código executável e as saídas são argumentos de saída que, em geral, servem de entrada para outra sub-rotina de módulo.

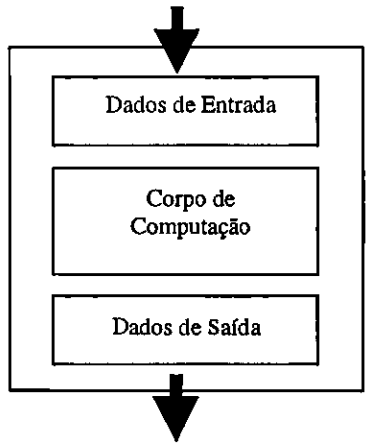


Figura 4.2 - Exemplo de módulo de visualização [Gal95].

O componente Controle de Fluxo define como os módulos de visualização são executados e como os dados fluem entre os módulos do sistema. Alguns autores [Gal95; Bro92] definem três categorias que refletem a evolução dos sistemas: bibliotecas (geradores de aplicação cuja interface é baseada em uma linguagem de programação), sistemas *turnkey*, e geradores de aplicação com uma interface de programação visual. Essas categorias não são rígidas, ou seja, os sistemas de visualização existentes podem ser classificados em mais de uma categoria. Esse é o caso do VTK (*Visualization Toolkit*), que pode ser considerado um gerador de aplicação, apesar de não possuir interface gráfica. Em geral, os *softwares* na primeira categoria requerem menos memória e menor capacidade de processamento e de armazenamento para serem executados, o que os toma convenientes para o uso em computadores pessoais ou ambientes cliente/servidor. Os *softwares* na categoria mais recente, geradores de aplicação, são mais convenientes para supercomputadores ou estações de

trabalho (*workstations*) [Bro92]. Por serem a primeira e a segunda categorias mais antigas, muitas aplicações foram desenvolvidas usando-as e muitos produtos se encaixam nessas classes. No futuro, ferramentas de visualização usando técnicas mais modernas aparecerão, mas atualmente essas técnicas ainda estão em um estágio experimental e precisarão de algum tempo para se consolidar.

A seguir, serão discutidas cada uma dessas categorias, exemplificando alguns sistemas existentes em cada caso.

4.2.1. Bibliotecas de Visualização

Segundo Brodlie e outros [Bro92], o uso de bibliotecas de visualização é o método mais tradicional de criar novas maneiras de visualizar e analisar dados. As bibliotecas interagem diretamente com o *hardware* gráfico. O usuário é obrigado a fornecer quase todos os componentes do *software* para suportar a sua aplicação: o programa principal, a interface com o usuário; além de manipular os dados e mapear a geometria. A maioria das bibliotecas básicas fornecem somente uma interface para os dispositivos gráficos, e algumas de mais alto nível manipulam entidades mais sofisticadas. Essas bibliotecas representam conjuntos de rotinas na forma de APIs (*Application Programming Interfaces*) que permitem ao usuário introduzir componentes de visualização nos seus programas.

As bibliotecas, em geral, adotam uma interface baseada em uma linguagem. O usuário constrói a visualização escrevendo um programa em um ambiente que suporta uma linguagem interpretada, similar a uma linguagem de comandos com macro funções construídas para suportar uma aplicação particular. Se a linguagem em questão é uma linguagem de programação genérica, então a biblioteca será mais flexível e extensível [Gal95] e as aplicações mais eficientes.

As principais vantagens das bibliotecas de visualização são sua flexibilidade e controle direto. A desvantagem é o fato de exigir grande quantidade de tempo investido na escrita e suporte ao código. Alguns exemplos são:

i) *Visual3* (MIT): fornece sub-rotinas escritas em FORTRAN e possui técnicas gráficas para CFD (*Computational Fluid Dynamic*).

ii) *CardinalVision* (Wilsonville, EUA): fornece uma biblioteca de objetos

escritos em C++ para campos escalares, vetoriais e tensoriais associados a malhas de elementos finitos.

iii) *FOCUS* (*Visual Kinematics*, EUA): fornece desenvolvimento de bibliotecas para gerenciamento de dados, interface com usuário, técnicas gráficas e de visualização para problemas de fluidos ou sólidos.

iv) VTK (*Visualization Toolkit*): é uma biblioteca completa, disponível com código fonte, implementada para sistemas Unix e Windows. Tem a possibilidade de programação em Tcl/Tk, C++ e Java, o que lhe permite ser classificado como um gerador de aplicação [Oli97; Sch99].

4.2.2. Sistemas Turnkey

Um sistema de visualização *turnkey* fornece uma interface e módulos que permitem ao usuário gerar visualizações sem qualquer programação [Gal95; Bro92]. Para os sistemas nessa categoria, o usuário fornece os dados e as instruções para o programa principal. O sistema fornece o programa principal e "renderiza" a imagem resultante [Bro92]. Esses sistemas são, geralmente, fáceis de usar e permitem a um usuário inexperiente visualizar rapidamente seus dados. Alguns sistemas são construídos para um domínio de aplicação particular, com interfaces projetadas especificamente para esse domínio de aplicação.

O projeto de um sistema *turnkey* deve fornecer funcionalidades suficientes para satisfazer os requisitos da maioria de seus usuários. Isto porque a funcionalidade de um sistema *turnkey* não pode ser modificada pelo usuário, embora alguns sistemas permitam ao usuário escrever um módulo leitor para a importação de dados [Gal95; Bro92]. Essa funcionalidade fixa é vista como uma desvantagem, já que é impossível satisfazer, a priori, os requisitos de todos os usuários. Por outro lado, a facilidade de uso e de construção de aplicações particulares podem superar essas desvantagens.

Alguns exemplos de sistemas nessa categoria são:

i) *FieldView* (*Inteligent Light*, EUA): fornece um ambiente de visualização para dinâmica dos fluidos.

ii) *Data Visualizer* (*Warefront Technologies*, EUA): fornece um ambiente de visualização de dados de elementos finitos generalizados. O usuário pode

trabalhar com múltiplas técnicas de visualização simultaneamente.

iii) SSV (*Sterling Software*, EUA): fornece recursos para visualização de dinâmica de fluidos (animação, produção e gravação). Possui vários módulos para visualização escalar e vetorial em malhas híbridas, sob o comando do usuário, através de interfaces gráficas.

4.2.3. Geradores de Aplicação

Geradores de aplicação fornecem ao usuário um conjunto de ferramentas para gerar uma visualização. Também é fornecido um conjunto de módulos básicos de visualização, assim como a possibilidade de se criar outros módulos [Gal95]. Os autores classificam os geradores de aplicação segundo o modo como uma aplicação é construída: usando uma interface de programação visual e/ou uma interface baseada em linguagem. A maioria dos geradores de aplicação adotam interfaces de programação visual [Bro92] que permitem ao usuário construir aplicações interagindo com uma representação bidimensional dos módulos de visualização.

Usando uma ferramenta interativa para construir um grafo direcionado, o usuário pode selecionar um conjunto de módulos de uma paleta, conectá-los, e criar aplicações complexas de visualização. Novos módulos para executar tarefas particulares a uma aplicação do usuário podem ser escritos e adicionados a essa paleta [Gal95; Bro92]. Os nós do grafo são módulos de visualização, e as linhas são os caminhos pelos quais os dados seguem de um módulo a outro, como mostra a figura 4.3 [Gal95]. As saídas de um ou mais módulos podem ser usadas como entradas de outros módulos, os quais são executados sempre que algum de seus dados de entrada mudar.

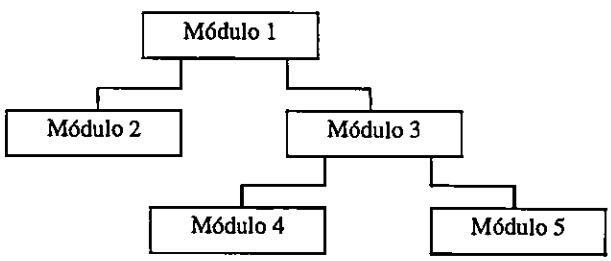


Figura 4.3 - Exemplo de aplicação modular de visualização [Gal95].

A implementação de geradores de aplicação pode ser bastante complexa. Isso

se deve ao fato de que cada módulo pode ser executado como um processo separado e, assim, o sistema de visualização deve suportar alguma forma de comunicação entre os processos. Se um módulo é um processo separado, um usuário pode escrever, compilar e "linkar" somente o módulo com o qual está trabalhando, sem ter que recompilar a aplicação inteira. Módulos também podem ser executados em máquinas remotas num ambiente heterogêneo de computadores [Gal95].

Gallagher [Gal95] também afirma que os geradores de aplicação são um excelente meio para prototipar uma aplicação de visualização. Eles são mais difíceis de se usar do que os sistemas *turnkey*, pois exigem mais estudos por parte do usuário, já que esse precisa adquirir dois conhecimentos básicos: 1) como construir a aplicação e 2) como escrever um módulo, se necessário. Esses sistemas são genéricos, apesar de exigirem mais tempo e recursos para serem usados. Alguns exemplos são:

i) AVS (*Advanced Visual Systems*, EUA): dividido em quatro módulos (um visualizador de imagens, um de geometria, um de volume e um editor de rede), possui uma interface de programação visual.

ii) *IBM Open Visualization Data Explorer* (OpenDX): sistema que segue a mesma filosofia de Fluxo de Dados (saídas de um módulo são entradas de outros) dos demais sistemas dessa classe. Possui um editor visual de programas e várias ferramentas interativas para exploração. Também permite ao usuário a criação dos seus próprios módulos usando C ou FORTRAN.

iii) *IRIS Explorer*: permite selecionar módulos pré-compilados com uma abordagem de seleção visual simples através do mouse. Módulos foram construídos em C e FORTRAN, e são selecionados através de um dos componentes do software: o *Librarian*. Os outros dois componentes são um Editor de Mapas, que permite editar a rede de módulos, e o *DataScribe*, responsável pela conversão de formatos entre o *Explorer* e outros programas.

Dentre estes, o *IRIS Explorer* e o OpenDX serão descritos em maiores detalhes na seção 4.4. Como dito anteriormente, essa classificação em três categorias se baseia na evolução das arquiteturas dos sistemas e, conseqüentemente, no tipo de interface que oferecem aos usuários. Parte-se agora para uma descrição de aspectos relevantes nas interfaces de sistemas de visualização que as tornam mais efetivas.

4.3. Aspectos de interação

O acesso à funcionalidade de um sistema de visualização pode se dar de duas formas básicas: linguagem de comando ou interface gráfica. Além do tipo de acesso às funções do sistema, um outro fator que influencia profundamente a eficiência na interpretação de dados e fenômenos é o tipo de técnicas de exploração e interpretação que o sistema oferece ao usuário para explorar os dados e gráficos. Essas técnicas de exploração dependem basicamente da funcionalidade e da qualidade da interface oferecida pelo sistema. As técnicas de exploração mais comuns são [Oli97]:

- Sondas (*probes*): uma das ferramentas mais poderosas de análise, uma sonda é um objeto geométrico que serve para coletar informações relevantes e "navega" pelo volume de acordo com o movimento do *mouse*. Nessa classe estão os extratores de geometria, que selecionam subvolumes a serem mapeados, e os apontadores, que fornecem valores numéricos em pontos das malhas.

- Calculadoras: necessárias como ferramenta de análise e, em geral, incluem processamento dedicado, como cálculo de fluxo e de gradientes.

- Gráficos: geração de gráficos deve estar sempre disponível para apresentação de variáveis selecionadas pelo usuário. Eles são ferramentas clássicas, úteis e bem conhecidas, e fornecem apoio à análise de dados.

- Mapas de cores: extremamente úteis para gerar representações gráficas de dados multivariados, ou para visualizar uma mesma variável de muitas maneiras diferentes. As cores devem ser cuidadosamente escolhidas para refletir corretamente tendências e padrões nos dados. Interfaces devem incluir apoio à definição interativa de mapas de cores.

- Realidade Virtual (RV): com o avanço da tecnologia, dispositivos desse tipo podem ser usados para processamento gráfico e análise de informação em tempo real. Em muitas circunstâncias, a sensação de presença fornecida pelos dispositivos de RV pode auxiliar a interpretação de dados multidimensionais.

Muitos aspectos influenciam a percepção dos objetos e sinais presentes nas imagens apresentadas ao usuário. Em Brodlie e outros [Bro92], os autores apresentam algumas questões associadas à interação com o usuário em visualização. Essas questões estão agrupadas segundo os aspectos envolvidos:

- Questões Cognitivas - relativas aos modelos mentais que o usuário gera ao estudar e analisar o problema e ao propor soluções para o mesmo.

- Questões Perceptuais - como as pessoas processam informações como cor, profundidade e movimento.

- Fatores Humanos - os que tentam fazer com que o usuário se sinta confortável com o sistema, tais como representações de escala, orientação e tempo, e fornecimento de ajuda.

Algumas dessas questões perceptuais e cognitivas são discutidas a seguir.

4.3.1. Questões Cognitivas e Perceptuais

A razão de ser dos sistemas de visualização é ajudar pessoas a resolver problemas. Assim, a área de psicologia cognitiva, que é o estudo de como as pessoas resolvem problemas, é altamente relevante. Hayes [Hay79 apud Bro92][1] postulou que as semânticas da representação de um problema podem determinar quão difícil ele é. Estudos mostraram que, dado um problema isomórfico (idêntico na forma), mas com representações semânticas diferentes, as pessoas não conseguem aplicar a solução de um problema ao outro (o seu isomorfo). Isso significa que o usuário não emprega o que aprendeu ao resolver o primeiro problema na solução do segundo, ainda que ambos sejam idênticos na forma, mas não na representação [Bro92, Owe96]. Essa falha para resolver um problema transferindo conhecimento entre ele e seu isomorfo ocorre quando a representação da tarefa é alterada. Na essência, as pessoas vêem os problemas como se fossem diferentes [Bro92]. Segundo Brodlie, esse tipo de questão deveria ser de importância central no projeto de sistemas de visualização. Os sistemas devem ser desenvolvidos com um entendimento claro dos processos cognitivos dos usuários.

As limitações perceptuais do usuário inexperiente não deveriam controlar a variedade e sofisticação das possíveis visualizações. Em outras palavras, como em qualquer sistema do qual se espera a capacidade de modelar e interpretar fenômenos complexos, há uma curva de aprendizado inevitável que os usuários finais serão forçados a percorrer.

[1] Hayes, J. - "Cognitive Psychology and Interaction", Methodology of Interaction, Elsevier North-Holland, 1979. Apud Brodlie, K. W.; Carpenter, L. A; Earnshaw, R. A; Gallop, J. P.; Hubbold, R. J.; Mumford, A M.; Osland, C. D.; Quarendon, P.; "Scientific Visualization - Techniques and Application", Springer-Verlag, 1992.

Possivelmente, o usuário possuirá o conhecimento do domínio e a experiência acumulada em uma aplicação particular. Isso permite a reunião de muitas apresentações complexas que, embora sem significado para um estranho, podem capturar apropriadamente os relacionamentos relevantes para o usuário.

Há várias questões perceptuais importantes em sistemas de visualização, sendo que o uso de cor é uma delas [Owe96], como discutido na seção 3.3. O sistema de percepção visual humano é muito complexo e o seu funcionamento deve ser considerado no projeto e uso de sistemas de visualização. As pessoas podem interpretar uma mesma imagem de forma bem diferente, dependendo da psicologia e da experiência de cada um. Para melhorar o processo de visualização, é preciso entender como o sistema visual percebe e identifica a informação. Assim, entendendo melhor a percepção visual, pode-se melhorar o grau de informação dos dados apresentados, aumentar a fidelidade das representações visuais e tomar o processo de observação da apresentação visual mais rápido e mais fácil.

4.4. Interfaces de Alguns Sistemas

Nesta seção são descritos uma biblioteca de visualização, o VTK, e a interface de dois sistemas bastante conhecidos e citados na literatura, o *IRIS Explorer* e o *Data Explorer*. Os últimos oferecem ferramentas poderosas para a visualização, além de uma interface gráfica e de uma linguagem de comandos.

4.4.1. Visualization Toolkit

O VTK (*Visualization Toolkit*) [Sch99] é uma biblioteca de visualização multiplataforma disponível via Web que consiste de algoritmos e estruturas de dados para visualização, compatível com as plataformas Unix e Windows. O software foi desenvolvido em C++ de acordo com o paradigma de orientação a objetos e tem a possibilidade de programação em Tcl/TK, C++, Java e Phyton. Uma camada de código cria uma interface entre as funções do VTK e do Tcl/TK. O *toolkit* é extensível, pois o seu código fonte está disponível. Isso permite o desenvolvimento de novas aplicações com uma interface gráfica própria, a alteração das técnicas já implementadas e a inclusão de novas técnicas de visualização.

- *vtkRenderer*: coordena o processo de renderização envolvendo fontes de luz, câmeras e atores;
- *vtkLight*: uma fonte de luz para iluminar a cena;
- *vtkCamera*: define a posição de observação e outras propriedades da cena;
- *vtkActor*: representa um objeto renderizado na cena, mas suas propriedades e posição no sistemas de coordenadas do usuário;
- *vtkProperty*: define as propriedades da aparência de um ator incluindo cor, transparência e propriedades de tonalização tais como coeficientes de reflexão especular e difusa.
- *vtkMapper*: define a representação geométrica de um ator.

4.4.1.1. A classe vtkLookupTable

A classe *vtkLookupTable* do VTK define um objeto gráfico que permite estabelecer um mapeamento de valores escalares para cores ou de cores para valores escalares, gerando uma tabela de cores pela da criação de instâncias da classe. Essas tabelas são definidas pela especificação de um número de entradas e de parâmetros de variação das cores. Os parâmetros podem ser definidos através dos modelos HSV (*hue*-matiz, *saturation*-saturação, *value*-luminância) ou RGB (*red*-vermelho, *green*-verde, *blue*-azul). Além disso, pode-se especificar ainda um valor de transparência (*alpha*), que varia de 0 a 1. Para gerar uma tabela utilizando o modelo HSV deve-se especificar o intervalo de variação de cada parâmetro, que pode variar de 0 a 1 ao longo de toda a tabela. As cores da mesma são definidas por uma função linear aplicada aos valores iniciais e finais dos componentes HSV e de transparência especificados. A geração de uma tabela *default* no VTK, utilizando a linguagem C++, pode ser feita como exemplificado nos dois trechos de código a seguir.

```
vtkLookupTable *lut = new vtkLookupTable;
      lut->SetHueRange(0.0, 0.6667);
      lut->SetSaturationRange(1.0, 1.0);
      lut->SetValueRange(1.0, 1.0);
      lut->SetAlphaRange(1.0, 1.0);
      lut->SetNumberOfColors(256);
      lut->Build();

vtkLookupTable *lut = new vtkLookupTable;
      lut->SetNumberOfColors(256);
      lut->Build();
```

A geração de visualizações no VTK é feita através da criação de *pipelines* de visualização, que correspondem a uma seqüência de transformações de dados de entrada até a renderização na tela, como mostra a figura 4.4.

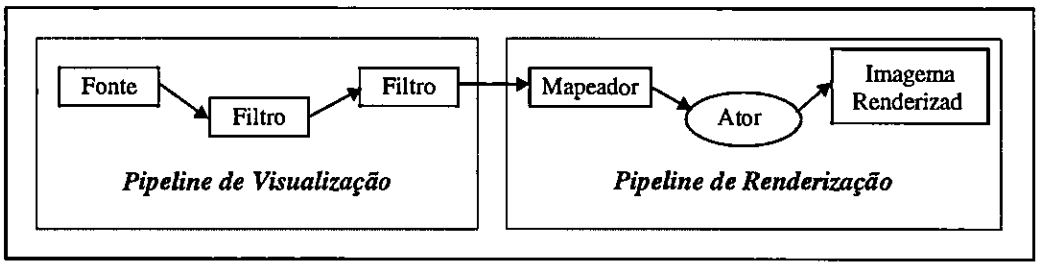


Figura 4.4 – Pipelines de um programa VTK.

Os objetos do VTK, instâncias das classes, estão divididos em duas categorias: objetos de dados (implementam os tipos de dados, como malhas e vetores) e objetos de processo (executam a criação da imagem e a transformação de dados). As conexões entre esses objetos são feitas através de comandos que unem as saídas de um objeto à entrada de outros.

O VTK possui um conjunto de classes com funcionalidades específicas, como as classes dos tipos fontes, filtros, mapeadores e gráficos. Os objetos da classe fonte iniciam o *pipeline* de visualização, fazendo a leitura de arquivos de dados externos ou criando dados a partir de variáveis de instância, e gerando um ou mais conjuntos de dados de saída. Os objetos da classe filtro recebem pelo menos um conjunto de dados de entrada e geram um ou mais conjuntos de dados de saída. Os mapeadores finalizam o *pipeline* de visualização, recebendo um ou mais objetos de entrada, e fazendo o mapeamento de dados para um determinado dispositivo ou para uma biblioteca gráfica. Os objetos gráficos representam a funcionalidade do núcleo de renderização. Esses objetos permitem a manipulação de fontes de luz, câmeras e atores, das propriedades dos atores e dos atributos de renderização (como, por exemplo, a cor de fundo do renderizador). Existem vários objetos que ser utilizados no processo de criação de uma cena, os oito objetos básicos mais usados são:

- *vtkRenderMaster*: coordena os métodos independente e cria uma janela de renderização (nas versões anteriores à versão 2.0);
- *vtkRenderWindow*: gerencia uma janela no monitor e permite que um ou mais renderizadores desenhem em uma instância dessa classe;

O primeiro trecho de código, no qual são utilizados os valores *default* para os parâmetros matiz, saturação, luminância, transparência e para o tamanho da tabela, é equivalente ao segundo trecho. A tabela gerada por esses códigos é mostrada na figura 4.5a.

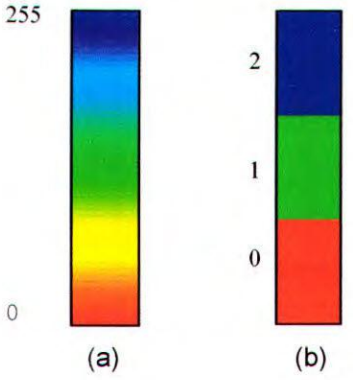


Figura 4.5 - Tabelas de cores geradas pelo VTK

A geração de tabelas usando as primitivas do modelo RGB é feita especificando os componentes R, G, B de cada entrada da tabela individualmente. É preciso especificar o número de entradas da tabela, construí-la e, então, inserir os componentes R, G e B e a transparência de cada cor. Desse modo, é possível gerar tabelas personalizadas que não podem ser expressas como funções lineares de valores HSV. O exemplo a seguir mostra a criação de uma tabela com 3 entradas, correspondendo às cores vermelha, verde e azul, na linguagem C++. A tabela gerada por esse trecho de código é ilustrada na figura 4.5b.

```
vtkLookupTable *lut = new vtkLookupTable;
      lut->SetNumberOfColors(3);
      lut->Build();
      lut->SetTableValue(0, 1.0, 0.0, 0.0, 1.0);
      lut->SetTableValue(1, 0.0, 1.0, 0.0, 1.0);
      lut->SetTableValue(2, 0.0, 0.0, 1.0, 1.0);
```

4.4.2. IRIS Explorer

O IRIS Explorer [IRI99] é um gerador de aplicações que está disponível em plataformas como *workstations* da *Silicon Graphics*, *workstations* da *Sun-SPARC*, computadores da *CRAY Research* ou mesmo em um PC (*Personal Computer*). Esse sistema fornece um conjunto de módulos que podem ser vistos como uma biblioteca de sub-rotinas. O usuário liga os diversos

módulos para construir um programa visual (também chamado de mapa, que é uma coleção de módulos que contêm uma série de operações associadas a um conjunto de dados). Esse programa visual será executado para gerar os resultados desejados. A cena resultante do processo de visualização pode ser gravada no formato VRML. Os usuários podem escrever seus próprios módulos com a ajuda de uma API (*Application Programming Interface*) que permite o acesso a todos os tipos de dados internos e, em muitos casos, permite reconfigurar a interface gráfica com o usuário. Em contraste com os sistemas *turnkey*, o *IRIS Explorer* suporta extensões funcionais e geração de aplicações. Além disso, é possível construir macros a partir de grupos de módulos conectados que sejam usados freqüentemente. Há o *ModuleBuilder*, que ajuda o usuário na escrita de módulos específicos. Esse sistema também oferece uma linguagem de *script* textual, a qual é baseada na linguagem *Scheme*. Ele possui três componentes principais:

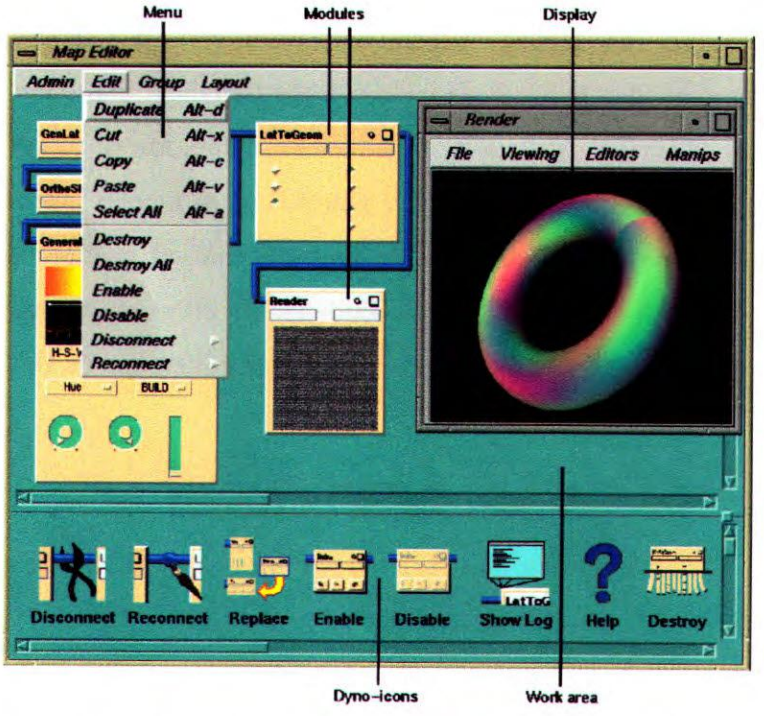


Figura 4.6 - Editor de mapas do IRIS Explorer [IRI96].

- *Map Editor* (Editor de mapas), que é uma área de trabalho para criar e modificar os mapas (programas visuais) (figura 4.6).

- *Module Builder* (Gerador de Módulos), que permite ao usuário criar seus próprios módulos de visualização.

O *IRIS Explorer* usa o gerenciador de janelas padrão *Motif* da *Silicon Graphics*, para construir a interface gráfica com o usuário. Entretanto, outros gerenciadores de janelas podem ser usados. O conjunto de *widgets* consiste em botões, *sliders*, *dials*, *typeins*, *file browsers* e editores de mapas de cores.

Existem diferentes tipos de interação com o usuário. O menu de opções gerais está disponível a partir de menus *pull down*. No ambiente de programação visual, o Editor de Mapas, técnicas de "arrastar e colar" (*drag and drop*) são utilizadas para selecionar os módulos da biblioteca e inseri-los no mapa (programa). A saída e entrada dos módulos são conectadas a outros módulos por um mecanismo de seleção e clique. O acesso aos parâmetros dos módulos pode ser feito de duas maneiras: pela janela do painel de controle ou do mini painel de controle, os quais são empilhados no módulo de apresentação do Editor de Mapas. Os valores dos parâmetros dos módulos podem ser alterados usando o mouse e, quando um controle mais fino dos valores é necessário, pode-se utilizar os *typeins*. Técnicas similares são usadas no *DataScribe* e no Gerador de Módulos. Os objetos na janela de visualização do módulo de renderização podem ser movidos, rotacionados e terem seu tamanho alterado utilizando-se o *mouse*. O posicionamento da câmera só é possível por manipulação direta via *mouse*.

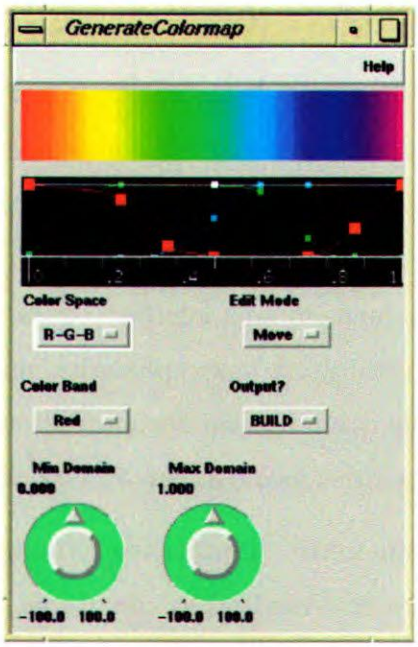


Figura 4.7 - Módulo *GenerateColormap* do *IRIS Explorer* [IRI96].

Através dos programas visuais, os dados podem ser manipulados de diferentes maneiras para gerar visualizações. Módulos controladores de *loop* fornecem mecanismos para construir *loops* simples e aninhados. Os módulos fornecidos pelo *IRIS Explorer* variam extremamente em função e complexidade.

O módulo responsável pela definição de mapas de cores é o *GenerateColormap* (figura 4.7). O painel de controle desse módulo permite selecionar um espaço de cores, manipular as faixas de cores e enviar os resultados para o módulo de renderização (*Render*), no qual os objetos dos dados visualizados são coloridos de acordo com seus valores. O *GenerateColormap* possui duas portas de entrada que aceitam vários tipos de malhas (*lattice*) de dados e vários parâmetros de entrada que são conectados a *widgets* no painel de controle do módulo. A primeira porta de entrada aceita um mapa de cores, o que significa que o usuário pode ler, de um arquivo, mapas de cores que contenham parâmetros pré-estabelecidos. A segunda porta aceita dados na forma de malhas, significando que limites máximo e mínimo do domínio podem ser estabelecidos no mapa de cores. A porta de saída produz um mapa de cores na forma de uma malha 1D, que pode ser conectada a qualquer módulo que aceite um mapa de cores como entrada.

4.4.3. IBM Open Visualization Data Explorer

Assim como o *IRIS Explorer*, o *IBM Open Visualization Data Explorer* (OpenDX) [IBM99] é um gerador de aplicações para criar visualizações de alto desempenho. Com esse sistema, o usuário pode definir um processo de visualização por meio de um programa visual, conectando os módulos de forma análoga à do *IRIS Explorer*; ou criar novos módulos em linguagem C ou FORTRAN. Há também uma interface baseada em linguagem de comandos para os usuários que preferem construir suas aplicações num estilo mais tradicional de programação. O OpenDX permite que o usuário documente seus programas visuais, de forma que futuros usuários desses programas tenham acesso a essa documentação.

A figura 4.8 mostra como as principais ferramentas e interfaces do OpenDX estão posicionadas no processo de visualização, de forma a indicar o seu papel nesse processo. Todas as ferramentas listadas abaixo da linha em negrito, exceto os módulos 6 e 10, são acessadas diretamente e manipuladas no Editor Visual de Programas (5), ilustrado na figura 4.9.

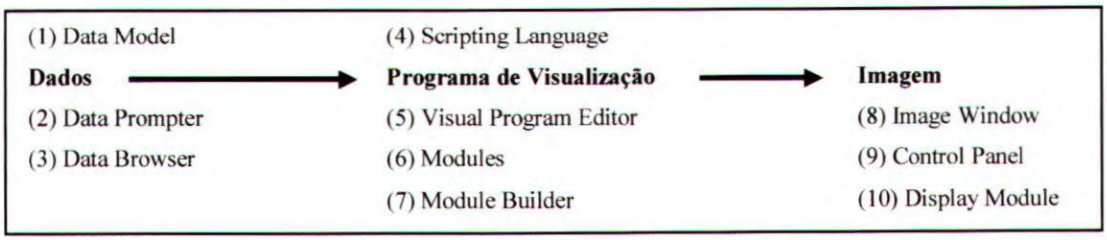


Figura 4.8 - Ferramentas do DX no processo de visualização.

A interface gráfica do OpenDX foi construída sobre os padrões X *Window* e *Motif*. O acesso e a troca dos valores dos parâmetros são feitos por meio dos *typeins* na janela do módulo de painel de controle. Pode-se conectar *widgets* aos parâmetros que são alterados com mais freqüência e, então, colocá-los no painel de controle personalizado. O conjunto de *widgets* consiste de *steppers*, *sliders*, *dials* e *typeins*. Um programa visual pode ser executado sem necessidade do editor de programas. Nesse caso, somente os parâmetros do painel de controle personalizado estão acessíveis.

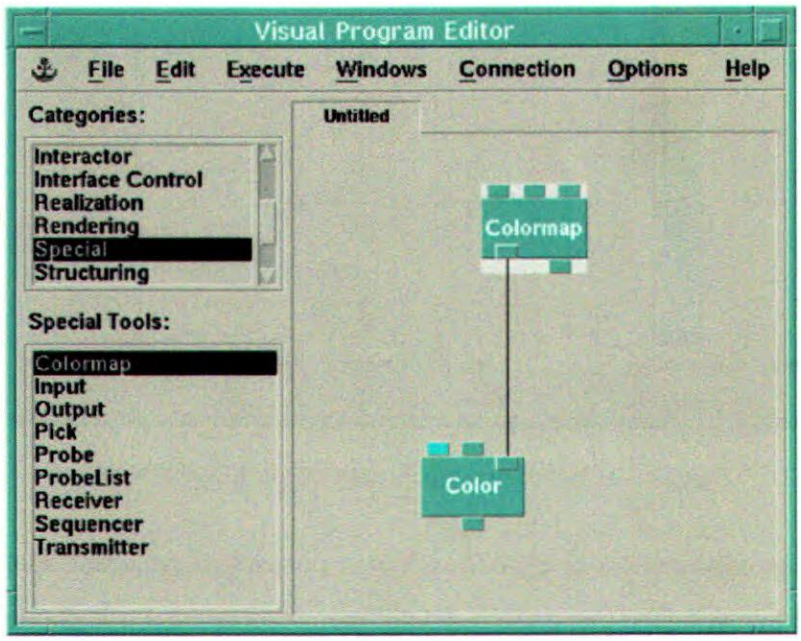


Figura 4.9 - Editor visual de programas do OpenDX [IBM96].

O posicionamento da câmera na janela de visualização pode ser feito usando o mouse ou digitando os valores exatos nos *typeins* correspondentes. Os objetos na janela de visualização não podem ser posicionados usando o mouse. Para movimentá-los, rotacioná-los ou alterar seus tamanhos, deve-se usar o módulo correspondente, o que é uma desvantagem do OpenDX. O *menu* pode ser acessado através de teclas de atalho, o que é conveniente para

usuários mais experientes.

O OpenDX também permite que o usuário mapeie cores para valores específicos dos dados, através da interface ilustrada na figura 4.10, a qual exibe o Editor de mapa de cores (*Colormap Editor*). Este controla o mapeamento da opacidade dos dados, que é o grau de transparência da imagem em relação ao fundo. Em resumo, o Editor de mapa de cores permite:

- controlar o intervalo dos dados sobre o qual o mapeamento ocorre;
- selecionar as cores que serão mapeadas para o intervalo dos valores;
- selecionar as opacidades que serão mapeadas para os intervalos dos valores.

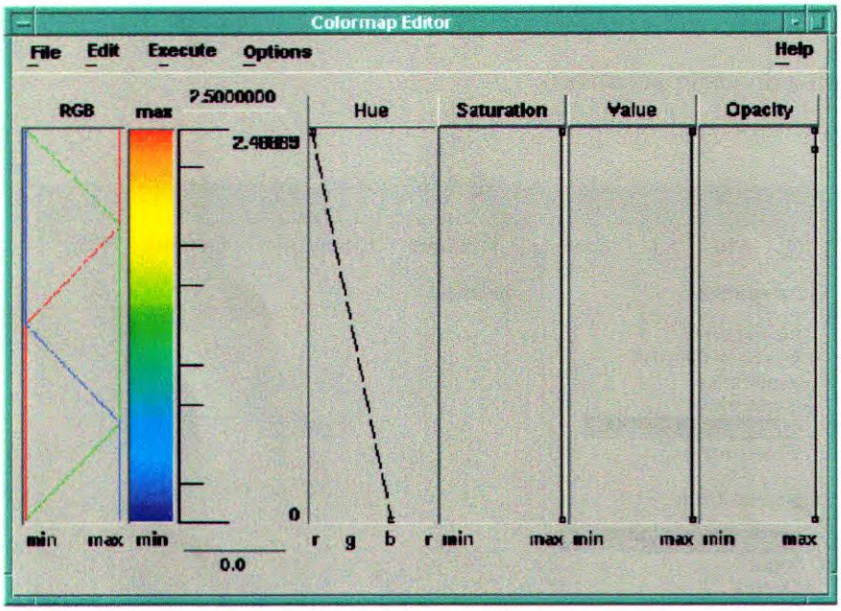


Figura 4.10 - Editor de mapa de cores do OpenDX [IBM96].

Esse editor também especifica cores no espaço de cores Matiz, Saturação e Valor (HSV). Na figura 4.10 é mostrado o *default* para cada um dos três parâmetros do espaço de cores HSV. As três áreas, *Hue*, *Saturation* e *Value*, podem ser alteradas independentemente uma das outras, através de operações com o mouse.

4.6. Considerações Finais

Os sistemas de visualização sofreram uma evolução natural desde o seu surgimento. Os sistemas atuais fornecem flexibilidade, uma interface gráfica amigável, possibilidade de programação visual ou escrita e abertura para novas funcionalidades. Porém, exigem bastante conhecimento do usuário.

Todas as funcionalidades de um sistema são utilizadas por meio da sua interface, que deve ser adequada ao tipo de tarefa de exploração desejada. As interfaces dos sistemas atuais tentam englobar conhecimentos sobre cognição e percepção para facilitar a interação usuário-computador. Os aspectos cognitivos envolvem a exploração do conhecimento/experiência do usuário em outras áreas, permitindo que esse conhecimento seja reaproveitado no contexto do projeto de uma visualização. Os aspectos perceptuais exploram a percepção humana de cor, forma, profundidade para auxiliar o usuário a, por exemplo, correlacionar objetos e identificar regiões.

Neste capítulo descreveu-se a arquitetura genérica dos sistemas de visualização e a classificação dos sistemas existentes em três categorias (bibliotecas, *turnkey* e geradores de aplicação). Alguns aspectos da interação usuário-computador foram abordados e descreveu-se três sistemas de visualização existentes, o VTK, o *IRIS Explorer* e o *IBM Open Visualization Data Explorer*, com ênfase em seus módulos de geração de tabelas de cores.

No próximo capítulo é apresentado o módulo *vtkRuleLookupTable* desenvolvido segundo a arquitetura e a ferramenta apresentados no capítulo 3, as alterações que este módulo sofreu durante a realização deste trabalho e a implementação da interface, denominada *vtkRLTInterface*, que é a proposta deste trabalho.

Capítulo V

Implementação da Interface para o Módulo de Mapeamento por Cores Baseado em Regras

5.1. Considerações Iniciais

O módulo de mapeamento por cores desenvolvido por Tutida [Tut98] e constitui a base deste trabalho se baseia na abordagem e na ferramenta descrita por Rogowitz e outros, descrita na seção 3.4, e na ferramenta correspondente descrita na seção 3.5. As funções e procedimentos implementam a taxonomia proposta pelos mesmos autores e apresentada na figura 3.9. Porém, nem todos os parâmetros analisados foram especificados explicitamente na taxonomia o que exigiu algumas pressuposições, como veremos no decorrer do capítulo.

Este capítulo está organizado da seguinte maneira: na seção 5.2 descreve-se a implementação da classe *vtkRuleLookupTable* desenvolvida por Tutida (implementação original); na seção 5.3 apresenta-se a nova versão desta classe desenvolvida por Fodra [Fod98]; na seção 5.4 relata-se algumas alterações efetuadas sobre a nova versão da classe. Finalmente, na seção 5.5 descreve-se a implementação da interface.

5.2. A Classe vtkRuleLookupTable (original por Tutida)

A partir da arquitetura proposta por Rogowitz e outros [Rog93a, Rog93b] e das descrições de um módulo de mapeamento de cores desenvolvido por esses autores [Ber95, Ber96, Rog96] baseado nessa arquitetura, um módulo (classe) para mapeamento de cores apoiado por regras foi desenvolvido para o VTK, em sua versão 1.3. Esse módulo, denominado *vtkRuleLookupTable,* foi implementado como uma classe derivada da classe *vtkLookupTable.* A figura 5.1 ilustra os principais métodos e os atributos da classe utilizando a notação do OMT (*Object Modeling Technique*) [Rum91].

As regras perceptuais, nesse módulo, estão implementadas pelos métodos *IsomorphicRules, SegmentationRules, HighlightingRules,* correspondendo, respectivamente, à regras para as tarefas de representação isomórfica, de segmentação e de enfatização. As tabelas de cores são geradas pelos métodos *DefineIsomorphicTable, DefineSegmentationTable* e *DefineHighlightingTable.*

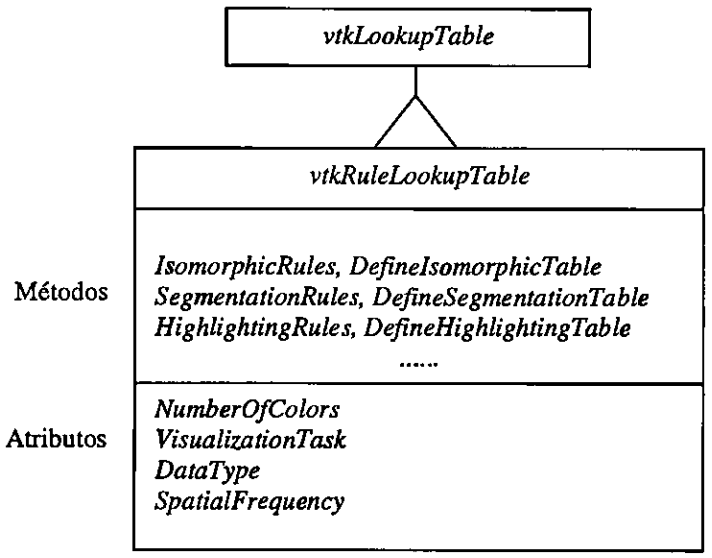


Figura 5.1 – Esquema da classe *vtkRuleLookupTable* com principais métodos e atributos [Tut98b]

Existem outros métodos de apoio associados à classe, como um método de conversão de parâmetros de cor do modelo HSV para o modelo RGB; métodos que permitem a gravação e recuperação de tabelas de cores; e um método de mapeamento de valores de dados em índices da tabela. O método *HSVToRGB()* faz a conversão dos parâmetros H, S e V associados a cada entrada da tabela para os parâmetros R, G e B correspondentes. O método

WriteRuleLookupTable(nome_arquivo) imprime em um arquivo as informações sobre o conjunto de dados que gerou a tabela e os parâmetros RGB de cada entrada da tabela. A tabela armazenada em arquivo pode ser recuperada com o método *ReadRuleLookupTable(nome_arquivo)* e aplicada a outros conjuntos de dados. Nesse caso, a tabela pode não ser adequada ao novo conjunto de dados, visto que não é feita uma verificação e um confrontação entre as características do conjunto de dados e as da tabela recuperada. O método *MapDataValueToIndex(valor)* verifica o intervalo de variação dos dados e estima a posição na qual o valor do dado será mapeado na tabela de cores do VTK. Esse método é utilizado na criação de tabelas de cores de segmentação e de enfatização, nas quais os segmentos ou intervalos são especificados em termos dos valores de dados que os delimitam.

Os matizes das tabelas definidos pela classe possuem códigos que variam de 1 a 6, de acordo com a ordem em que as cores são representadas em um círculo de matizes. Nesse círculo, a cor vermelha corresponde a 0° ou 360° (código 1), amarelo a 60° (código 2), verde a 120° (código 3), turquesa a 180° (código 4), azul a 240° (código 5) e magenta a 300° (código 6). Pares de cores oponentes ou complementares são definidos pelo código de cada uma das cores, sem uma ordenação fixa. Por exemplo, um par composto pelas cores verde e vermelha é especificado pelo código 31, em que o 3 corresponde ao verde e o 1 ao vermelho.

Os metadados que informam a tarefa de visualização e tipo do dado também são especificados pelo programador através de códigos numéricos. Os códigos definidos para a especificação da tarefa são: 1 para tarefas de isomorfismo; 2 para de segmentação e 3 para de enfatização. A classe permite, ainda, a geração da tabela de cores do arco-íris especificada pelo código 4, e da tabela com a escala de cinzas, especificada pelo código 5. O tipo dos dados (intervalo ou razão) é especificado pelo usuário, sendo o código 1 para o tipo razão e 2 para o tipo intervalo. Na taxonomia em que a classe foi baseada há, ainda, os tipos de dados ordinal e nominal, que não foram implementados nessa versão. O intervalo de variação (*range*) de dados é obtido do próprio conjunto de dados, verificando-se o menor e o maior valores no mesmo, e a freqüência espacial dos dados é determinada pelos métodos da classe *SpatialFrequency*.

No caso de ser especificado um código incorreto, o programa envia uma mensagem de erro e assume um valor *default*. Os valores *default* para matiz são: vermelho para tabelas formadas por um único matiz e o par vermelho-verde para tabelas formadas por 2

matizes. O valor *default* para tarefa de visualização é 1 (tarefa de isomorfismo) e o *default* para o tipo de dados é 1 (tipo razão).

5.2.1. Tarefa de Isomorfismo

Nas tabelas de cores para a tarefa de isomorfismo, projetadas para produzir uma representação fiel das estruturas contidas nos dados, as regras são implementadas pelo método privado *IsomorphicRules()*, e o método privado *DefineIsomorphicTable()* é responsável por construir a tabela no modelo HSV para essa tarefa. O usuário deve definir a tarefa de visualização com o método *SetVisualizationTask(tarefa)*, o número de entradas da tabela com o método *SetTableSize(num_entradas)*, o tipo dos dados com o método *SetType(tipo_dado)*, o matiz ou par de matizes com o método *SetTableColor(código_cor)*, o nome do arquivo de dados a ser visualizado com o método *DataCharacteristic(nome_arquivo)*, aplicar as regras com o método *ApplyRules()*, e então construir a tabela no modelo RGB com o método *Build()*. O usuário pode definir valores de opacidade para as posições inicial e o final da tabela com o método *SetTableOpacity(opac_inicial, opac_final)*. Esses valores variam entre 0 e 1, e são incrementados ou decrementados monotonicamente nas posições intermediárias da tabela. Um exemplo, em C++, que ilustra a criação de um tabela de cores para a tarefa de isomorfismo é fornecido a seguir:

```
vtkRuleLookupTable* lut = new vtkRuleLookupTable;
    lut -> SetVisualizationTask(1);     //tarefa de isomorfismo
    lut -> SetTableSize(50);            //tamanho da tabela = 50
    lut -> SetType(1);                  //dados do tipo razão
    lut -> SetTableColor(31);           //cores verde-vermelho
    lut -> DataCharacteristic(nome_arquivo);
    lut -> ApplyRules();
    lut -> Build();
```

Para tarefas de isomorfismo as tabelas de cores para dados do tipo razão são formadas por 2 matizes que compõem pares de cores oponentes (vermelho-azul, azul amarelo, turquesa-vermelho, magenta-verde). Nesta implementação, essas tabelas são divididas em duas porções de tamanhos iguais, cada uma associada a um matiz. Para dados de baixa freqüência espacial, a luminância é mantida constante e a saturação possui valor máximo (igual a 1) nos extremos da tabela e valor mínimo (igual a 0) no meio da tabela. Para dados de alta freqüência espacial, a saturação e a luminância possuem valor máximo nos extremos na tabela e valor mínimo no meio da tabela. As tabelas para dados do tipo intervalo são formadas

por um único matiz. Se freqüência espacial é baixa, a luminância é mantida constante (1) e a saturação varia monotonicamente de 0 a 1. Se a freqüência espacial é alta, a saturação é mantida constante (1) e a luminância varia monotonicamente de 0 a 1. Exemplos de tabelas para tarefa de isomorfismo podem ser vistos na figura 5.2.

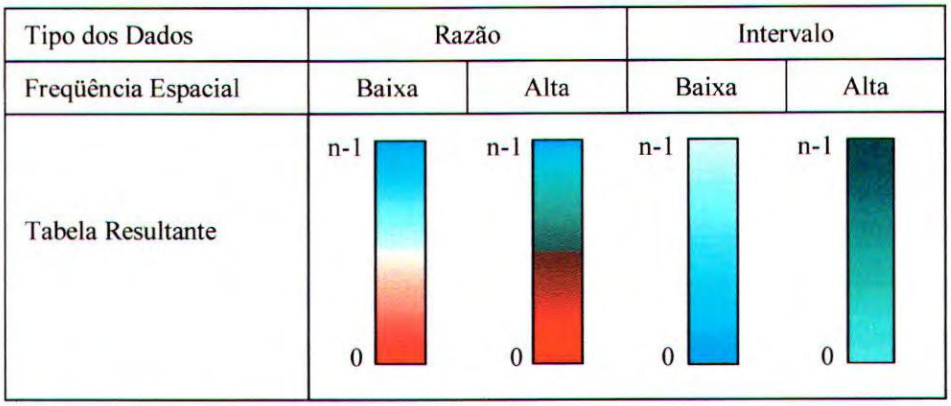


Figura 5.2 – Classificação de tabelas para a tarefa de isomorfismo [Tut98b].

5.2.2. Tarefa de Segmentação

Em tarefas de segmentação, as tabelas são projetadas para delinear regiões visualmente e, portanto, são compostas por segmentos de cores com características diferenciadas que devem corresponder a diferentes intervalos de interesse nos dados. A quantidade de segmentos que a tabela pode conter é definida de acordo com o tipo dos dados e a freqüência espacial. Nesta implementação o número máximo de segmentos permitidos em qualquer caso foi fixado (arbitrariamente) em 6. Para cada segmento deve ser especificado o seu início; e os parâmetros matiz, saturação e luminância de cada cor. Nesta implementação, o início de cada segmento é especificado em termos dos valores de dados correspondentes ao início de um intervalo de interesse, sendo que o início do primeiro segmento deve corresponder ao menor valor do conjunto de dados. O final de cada segmento coincide com o início do próximo, sendo que o último vai até o maior valor do conjunto de dados.

O método privado *SegmentationRules()* define o número máximo de segmentos que a tabela pode conter, de acordo com o tipo e a freqüência espacial dos dados, e o método privado *DefineSegmentationTable()* constrói as tabelas no modelo HSV para essa tarefa. O usuário deve especificar a tarefa de visualização com o método *SetVisualizationTask(tarefa)*, o número de entradas da tabela com o método *SetTableSize(num_entradas)*, o nome do

arquivo a ser visualizado com o método *DataCharacteristic(nome_arquivo)*, o tipo dos dados com o método *SetType(tipo_dado)*, o número de segmentos com o método *SetNumberOfSegments(num_segmentos)*, o início de cada segmento (ou seja, o valor do dado a ser mapeado no início desse segmento da tabela), bem como os seus parâmetros de matiz, saturação e luminância, com o método *SetSegmentationTable(início_segmento, código_cor, saturação, luminância)*. Em seguida, deve aplicar as regras com o método *ApplyRules()* e construir no modelo RGB com o método *Build()*. Se o usuário especificar um número de segmentos incorreto, o programa envia uma mensagem de erro e assume um número de segmentos *default* que depende do tipo e da freqüência espacial dos dados. Se o usuário não especificar um código de cor para algum segmento, ou se especificar o mesmo matiz mais de uma vez, ou valores dos segmentos fora do intervalo de variação dos dados, o programa envia uma mensagem de erro e assume a tabela de segmentação *default*. Caso o usuário especifique um valor de saturação ou de luminância inválido, o programa envia uma mensagem de erro e assume o valor 1,0 para ambos. O método *SetSegmentationTable(início_segmento, código_cor, saturação, luminância)* assume que o fim de um segmento coincide com o início do próximo. Com o método *SetSegmentationTable(início_segmento, código_cor, saturação, luminância, opacidade)* o usuário pode, também, definir um valor de opacidade entre 0,0 a 1,0 para cada segmento. Para um conjunto de dados escalares inteiros, uma possível tabela de cores com 3 segmentos pode ser criada conforme o trecho de código C++ a seguir:

```
vtkRuleLookupTable * lut = new vtkRuleLookupTable;
    lut -> SetVisualizationTask(2); //tarefa de segmentação
    lut -> SetTableSize(50);        //tamanho da tabela = 50
    lut -> SetType(2);              //tipo dos dados intervalo
    lut -> SetNumberOfSegments(3); //número de segmentos = 3
    lut -> DataCharacteristic(nome_arquivo);
    lut -> SetSegmentationTable(0, 1, 1, 1); //início = 0, cor
                                //vermelha, saturação e luminância =1
    lut -> SetSegmentationTable(4, 2, 1, 1); //início = 4, cor amarela
    lut -> SetSegmentationTable(8, 5, 1, 1); //início = 8, cor azul
    lut -> ApplyRules();
    lut -> Build();
```

A figura 5.3 mostra a distribuição do número de segmentos segundo o tipo dos dados e sua freqüência espacial. Se os dados são do tipo razão, as tabelas são formadas por um número par de segmentos, de forma a garantir a ocorrência de uma transição no zero. Para dados de baixa freqüência espacial pode ser aplicada uma tabela com vários segmentos, e para dados de alta freqüência espacial recomenda-se tabelas com poucos segmentos. Um exemplo

de algumas tabelas de segmentação geradas pela classe *vtkRuleLookupTable* de acordo com o tipo e a freqüência espacial dos dados é apresentado na figura 5.4

Tipo dos Dados	Razão		Intervalo	
Freqüência	Baixa	Alta	Baixa	Alta
Número de segmentos	4	2	6	3

Figura 5.3 – Número de segmentos para tarefas de segmentação, segundo o tipo e a freqüência espacial dos dados.

5.2.3. Tarefa de Enfatização

As tabelas para a tarefa de enfatização são projetadas para atrair a atenção do usuário para regiões dos dados que apresentam certas características de interesse. Nos caso, o usuário deve identificar intervalos de valores dos dados a serem enfatizados perceptualmente na representação visual.

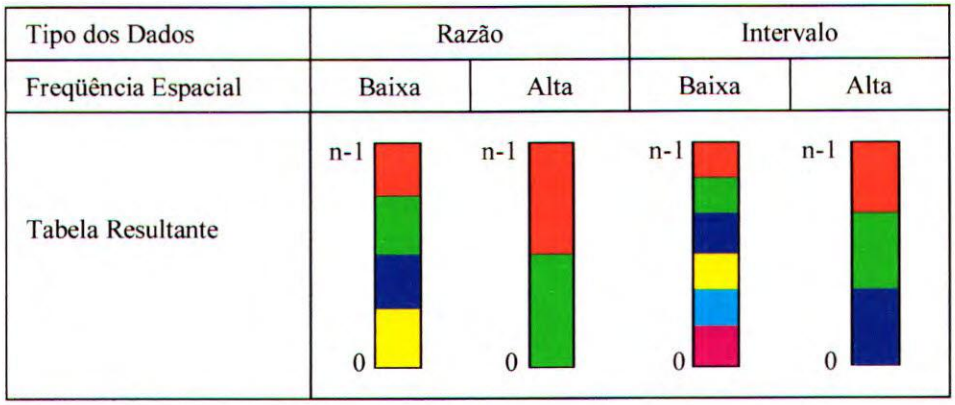


Figura 5.4 – Classificação de tabelas para a tarefa de segmentação [Tut98b].

Um fato essencial para a criação de tabelas de cores que enfatizam uma regra de um conjunto de dados é a não existência de interferência entre os diferentes componentes da cor utilizados numa representação. Se, por exemplo, o matiz for utilizado para diferenciar os elementos de um conjunto de dados, a variação da luminância ao longo de um intervalo de mesmo matiz faz com que este seja enfatizado sem que haja detrimento de outra informação. As tabelas para a tarefa de enfatização são compostas por 2 matizes, sendo que os parâmetros de saturação e luminância permanecem constantes ao longo da tabela. A taxonomia utilizada como referência não faz restrições quanto ao tipo de dado para a criação de tabelas para essa

tarefa, apenas sugere que o intervalo enfatizado seja menor para dados de alta freqüência espacial e maior para dados de baixa freqüência. Nesta implementação, o usuário define um intervalo de valores a seu critério, ou seja, ele pode enfatizar uma única região de interesse e não se verifica se o tamanho do intervalo é adequado considerando a freqüência espacial dos dados.

O método privado *HighlightingRules()* define o número de matizes que a tabela de cores pode conter, que nesta implementação foi fixado arbitrariamente em 2, e o método privado *DefineHighlightingTable()* constrói a tabela para essa tarefa usando o modelo HSV. O usuário deve especificar a tarefa de visualização com o método *SetVisualizationTask(tarefa)*, o número de entradas da tabela com o método *SetTableSize(num_entradas)*, o nome do arquivo a ser visualizado com o método *DataCharacteristic(nome_arquivo)*, o par de matizes que irá compor a tabela de cores com o método *SetTableColor(código_cor)*, o intervalo de valores a ser enfatizado com o método *SetHighlightingInterval(início_intervalo, fim_intervalo)*, aplicar as regras com o método *ApplyRules()* e construir a tabela usando o modelo RGB com o método *Build()*. O intervalo de valores deve estar dentro do intervalo de variação do conjunto de dados, caso contrário o programa envia uma mensagem de erro e assume a tabela de enfatização *default*. A tabela *default*, que pode ser especificada com o método *SetDefaultHighlightingTable()*, é formada pelas cores azul e amarelo, e enfatiza uma faixa correspondente aos 10 valores centrais do conjunto de dados. O usuário pode definir valores de opacidade para o intervalo enfatizado e para as outras entradas da tabela com o método *SetIntervalOpacity(intervalo, opacidade)*. A definição de uma tabela de cores para a tarefa de enfatização, para um conjunto de dados escalares inteiros com intervalo de variação [0, 100] e intervalo de valores a serem enfatizados entre [45, 55], pode ser feita conforme o seguinte código em C++:

```
vtkRuleLookupTable* lut = new vtkRuleLookupTable;
    lut ->SetVisualizationTask(3);        //tarefa de enfatização
    lut -> SetTableSize(50);              //tamanho da tabela = 50
    lut -> SetType(2);                    //dados do tipo intervalo
    lut -> SetHighlightingInterval(45, 55) //intervalo de 45 a 55
    lut -> SetTableColor(25);             //par de cores amarelo-azul
    lut -> DataCharacteristic(nome_arquivo);
    lut -> ApplyRules();
    lut -> Build();
```

A figura 5.5 ilustra uma tabela de enfatização pela classe *vtkRuleLookupTable*.

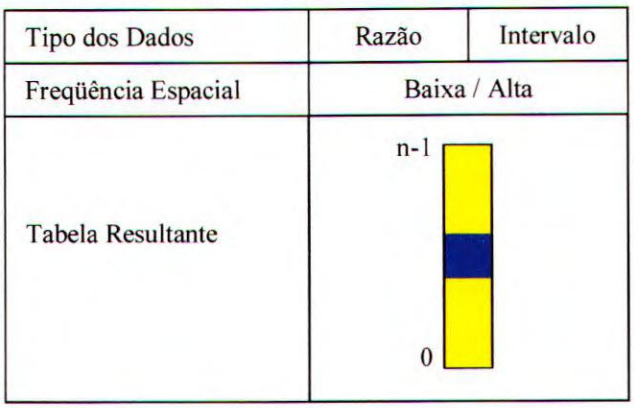


Figura 5.5 – Classificação de tabelas para a tarefa de enfatização [Tut98b].

5.2.4. Determinação da Freqüência Espacial

A classificação da freqüência espacial do conjunto de dados (alta ou baixa) foi implementada com a aplicação de um filtro passa-baixa convencional [Gon78] generalizado para trabalhar com conjuntos de dados 2D e 3D. Essa operação de filtragem resulta em uma suavização das altas variações de freqüência contidas no conjunto de dados. A implementação do filtro passa-baixa utiliza uma máscara de dimensão 3 e cria uma matriz composta pela média dos vizinhos da matriz original. Uma nova classe chamada *SpatialFrequency*, independente da classe *vtkRuleLookupTable*, implementa o procedimento de classificação da freqüência. Essa classe inclui métodos de leitura do conjunto de dados, de filtragem e de análise da freqüência espacial. A figura 5.6 ilustra os principais métodos e atributos da classe, utilizando a notação do OMT [Rum91].

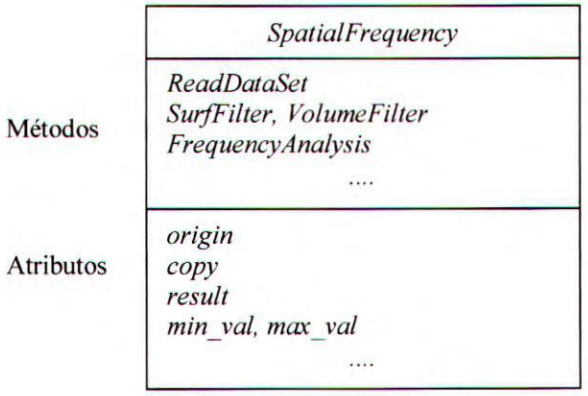


Figura 5.6 – Esquema da classe SpatialFrequency com os principais métodos e atributos [Tut98b].

O método privado *ReadDataSet()* faz a leitura do arquivo de dados, armazena os valores lidos em uma matriz (2D ou 3D, de acordo com as dimensões especificadas no cabeçalho do arquivo) e obtém os valores mínimo e máximo do conjunto (intervalo de variação dos dados). A operação de filtragem, implementada pelos métodos *SurfFilter()*, para conjuntos de dados bidimensionais, e *VolumeFilter()*, para conjuntos de dados tridimensionais, resulta na eliminação das freqüências altas do conjunto de dados e o efeito dessa operação sobre a imagem resultante da visualização seria a suavização das bordas, que correspondem aos dados de alta freqüência. O método público de análise da freqüência, denominado *FrequencyAnalysis(stream)*, implementa uma operação que calcula a diferença entre a matriz que contém os dados originais e a matriz que contém os dados filtrados, e a seguir calcula o desvio padrão normalizado dessa diferença, segundo o intervalo de variação dos dados. O valor do desvio padrão será maior quando o conjunto de dados original contiver informação de alta freqüência; e menor, caso contrário. Caso o resultado seja menor ou igual a 0.1 os dados são classificados como de baixa freqüência espacial; caso contrário são classificados como de alta freqüência, como sugerido por Bergman e outros [Ber95; Ber96]. Essa é uma forma aproximada de resolver o problema, que evita o cálculo de transformadas de Fourier. A análise seria mais precisa se realizada por uma análise de Fourier, mas o custo computacional associado seria alto, e a experiência indica que o cálculo aproximado é suficiente nesse caso.

5.2. A Nova Classe vtkRuleLookupTable (por Fodra [Fod98])

A primeira implementação da classe *vtkRuleLookupTable* apresenta algumas incompatibilidades com o padrão de programação adotado no VTK. Por exemplo, a classe *SpatialFrequency*, usada para determinar a distribuição espacial do dados, executa essa tarefa através do método *DataCharacteristic(nome_arquivo)*, que retorna a classificação da freqüência espacial (HIGH ou LOW). Entretanto, segundo a filosofia do VTK classes que acessam arquivos de dados devem ser implementadas como sub-classes da classe genérica *vtkReader*, o que não foi feito. Adicionalmente, se o arquivo de dados é lido através da classe do VTK denominada *vtkStructuredPointsReader*, a informação sobre a freqüência espacial poderia ser calculada sem necessidade de acesso adicional ao arquivo. Por esses motivos, decidiu-se pela reestruturação do cálculo da freqüência espacial. Para isso, foi criada uma

nova classe denominada *vtkStructuredPointsMetadata*, derivada da classe *vtkStructuredPoints*, contendo um método *GetFrequency()* que calcula a freqüência espacial dos dados. Desta maneira, a classe *SpatialFrequency* tornou-se desnecessária. A nova classe do VTK que permite ler os dados e determinar a sua distribuição espacial é utilizada da seguinte forma:

```
vtkStructuredPointsReader reader;
vtkStructuredPointsMetadata * lpSPM;

reader.SetFilename("nome_arquivo.vtk");
lpSPM = (vtkStructuredPointsMetadata*) reader.GetOutput();
```

Note que a saída do *vtkStrucutredPointsReader* é um objeto da classe *vtkStructuredPoints* que, portanto, deve ser convertido para um objeto da classe *vtkStructuredPointsMetadata* por um *type casting*, para que os métodos desta classe possam ser utilizados. Criou-se, também, um novo método para a classe *vtkRuleLookupTable* que obtém a classe a freqüência espacial dos dados. Esse método foi denominado *SetFrequency(frequência)* e sua utilização é exemplificada a seguir:

```
vtkRuleLookupTable lut;
lut.SetFrequency(lpSPM->GetFrequency());
```

O método *DataCharacteristic(nome_arquivo)* da classe *SpatialFrequency* também calculava o intervalo de variação dos dados, informação utilizada por vários métodos da classe *vtkRuleLookupTable*. Assim, a classe *vtkStructuredPointsMetadata* inclui também o método *GetRange(intervalo)*, que obtém o intervalo de variação do conjunto de dados [Fod98], e à classe *vtkRuleLookupTable* foi acrescentado o método *SetRange(intervalo)*. Ele pode ser utilizado da seguinte forma:

```
lut.SetRange(lpSPM->GetPointData()->GetScalars()->GetRange());
```

Um outra mudança ocorreu na fomra como é feito o cálculo da freqüência espacial. Como os dados estão representados em uma classe do VTK, é possível utilizar filtros do próprio VTK para a realização do cálculo (segundo a mesma abordagem anterior), em especial o *vtkImageMedian*, que substitui cada ponto pela média dos valores dos pontos ao seu redor. Os dados são convertidos para a classe *vtkImageSource* utilizando o filtro *vtkStructuredPointsToImage*. A diferença entre os dois objetos é calculada utilizando o filtro

vtkImageArithmetic, e esta diferença é convertida para um objeto da classe *vtkStructuredPoints* com o filtro *vtkImageToStructuredPoints*. Com esse último objeto é calculada a freqüência espacial aproximada. Se o desvio normalizado dos dados for menor que 0.1, a freqüência espacial é baixa, caso contrário, é alta. Essa nova implementação mostrou-se mais robusta que a anterior.

As mudanças mencionadas foram efetuadas como parte de um programa de Iniciação Científica [Fod98]. Apesar de serem muitas e de grande importância, essas não foram as únicas modificações que a classe *vtkRuleLookupTable* sofreu. Em ambas as implementações os métodos de escrita e recuperação de uma tabela em arquivo não chegaram a ser testados. Como parte deste trabalho esses métodos foram testados e alguns problemas foram detectados no método de recuperação de uma tabela. Esses problemas, advindos de erro de implementação, foram sanados.

5.2.1. Extensão da Nova Classe

Outra alteração na classe *vtkRuleLookupTable*, ainda para a versão 1.3 do VTK, foi a extensão da tabela de segmentação com base em um estudo realizado por Healey e Enns [Hea96a; Hea96b]. Esse estudo descreve uma técnica para a escolha de uma tabela de múltiplas cores para visualização de dados que tem como objetivo maximizar o número de cores disponíveis ao usuário, mas de forma a permitir uma detecção precisa e rápida de elementos em qualquer uma das cores. A questão central é "como escolher cores que assegurem boa identificação entre os elementos de dados durante a tarefa de visualização?" Essa tarefa pode envovler, por exemplo, a identificação e localização de um elemento de interesse. Segundo Healey e Enns, várias pesquisas sugerem que a escolha efetiva de cores para visualização de dados depende de pelo menos a análise de 3 critérios distintos: a distância entre as cores, a separação linear e a categoria das cores. A distância entre duas cores é a distância Euclidiana entre elas, medida da mesma forma que em um modelo de cores perceptualmente balanceado, conforme descrito na seção 2.7.5, equação 2.17. A separação linear é a capacidade de separar linearmente cores alvo de cores não alvo no espaço de cores definido pelo modelo de cores utilizado. As categorias das cores são as regiões de cores nomeadas no espaço de cores definido pelo modelo de cores (região de vermelho, de verde, de azul, de púpura etc.). Os autores realizaram vários experimentos com um número variado de cores: estudo de três (verde,

amarelo e púrpura), cinco (vermelho, verde amarelado, verde, azul e púrpura), sete (vermelho, amarelo, verde amarelado, verde, ciano, azul, púrpura)e nove cores (vermelho, laranja, amarelo, verde amarelado, verde, ciano, azul, púrpura e púrpura avermelhado). Eles realizaram esse experimento para investigar a relação entre o número de cores apresentado e o tempo necessário para determinar a presença de um elemento alvo. Uma análise dos dados coletados durante os experimentos levou-os à conclusão de que o número máximo de cores isoluminantes que podem ser apresentadas de uma só vez, enquanto permitindo uma identificação rápida e precisa das mesmas, é 7 (sete), sendo elas: vermelho, laranja, amarelo, verde, ciano, azul e púrpura.

No módulo *vtkRuleLookupTable,* a tabela de segmentação foi desenvolvida contendo 6 matizes isoluminantes (vermelho, amarelo, verde, ciano, azul e magenta) especificados no modelo de cores HSV, escolhidos por serem as 6 cores equidistantes que definem o círculo de cores que é a base da representação do modelo utilizado. Decidimos, então, acrescentar o matiz laranja à tabela. Desta forma, o número máximo de segmentos, que era 6 (seis), passou a 7 (sete). As cores no *vtkRuleLookupTable* foram codificadas de 1 a 6, seguindo a mesma ordem em que foram apresentadas no início do parágrafo. Como o matiz laranja se localiza entre as regiões dos matizes vermelho e o amarelo num círculo de cores (que seria a base do cone que representa o modelo de cores HSV), a maneira correta de codificá-lo seria com o número 2 (que é o código do amarelo). Para não alterar toda a codificação dos matizes, optamos por codificá-lo como 7 (sete). Desta maneira, bastou acrescentar algumas linhas ao programa original, sem a necessidade de alterar todo o código.

Uma outra alteração feita, foi a inclusão de uma função que permite ao usuário alterar qualquer tabela gerada pela classe *vtkRuleLookupTable*. Tal função é necessária para dar maior flexibilidade ao módulo e à interface, fornecendo recursos para usuários experientes gerarem outras tabelas, além das que são fornecidas pelo módulo. As informações da tabela gerada são armazenadas em uma estrutura *array* com, no máximo, 256 entradas. Os parâmetros de entrada dessa função são: um intervalo que deve estar contido no intervalo de variação do conjunto de dados, um matiz que deve ser um dos sete matizes possíveis, um intervalo de opacidade, um intervalo de variação de saturação e um intervalo de variação de brilho. Ela obtém os índices de entrada da tabela correspondentes ao início e ao fim do intervalo de dados e associa a cada índice do intervalo o matiz especificado e a variação da saturação, do brilho e da opacidade. É importante lembrar que, após a chamada a esta função,

devemos reconstruir a tabela chamando a função *Build()*. A função de alteração da tabela foi denominada *ChangeTableInterval(float rangeInit, float rangeEnd, int color, float opacInit, opacEnd, float satInit, satEnd, float brightInit, brightEnd)* e sua utilização se dá como no seguinte trecho de código em C++:

```
...
lut->ApllyRules() //Aplica as regras
lut->Build();           //Constroe a tabela

lut->ChangeTableInterval(45, 55, 3, 0, 1, 1, 1, 0, 1 ); //Altera intervalo
lut->Build();           //Reconstrói a tabela
```

Nesse trecho alteramos o subintervalo de dados [45, 55] de uma tabela que supomos ter 100 entradas, e cujo intervalo de variação de dados é de 0 a 100. Nesse trecho, alteramos o subintervalo para a cor verde (código 3), variamos a opacidade entre 0 e 1, mantemos a saturação constante (1 a 1) e variamos o brilho entre 0 e 1.

5.3. Implementação da Interface

Este trabalho teve como objetivo o desenvolvimento de uma interface gráfica para um módulo de definição de tabelas de cores baseado na classe *vtkRuleLookupTable*. Essa interface deveria fornecer ao usuário um processo mais intuitivo e simples de geração de tabelas de cores embutindo as regras perceptuais discutidas na seção 3.4.

A interface para o módulo de visualização, chamada *vtkRLTInterface,* foi desenvolvida no ambiente de programação *Microsoft Visual C++* [Kru97], versão 5.01 para a plataforma Windows. As janelas de diálogo foram projetadas utilizando os recursos da *Microsoft Foundation Class* (MFC), que contêm basicamente dois tipos de janelas, *modal* e *modaless*, ambas utilizadas neste trabalho. As janelas do tipo *modal* são aquelas que, quando ativas, não permitem ao usuário acessar outras janelas do mesmo aplicativo até que ela seja desativada (fechada). Já as do tipo *modeless*, permitem ao usuário continuar tendo acesso a outras janelas do mesmo aplicativo quando ativas [Kru97].

O *vtkRLTInterface* não permite ao usuário lançar mão de todos os recursos do *VTK*, mas apenas de recursos básicos para visualização de volumétrica (visualização de um conjunto de dados tridimensional), incluindo renderização volumétrica direta (DVR –

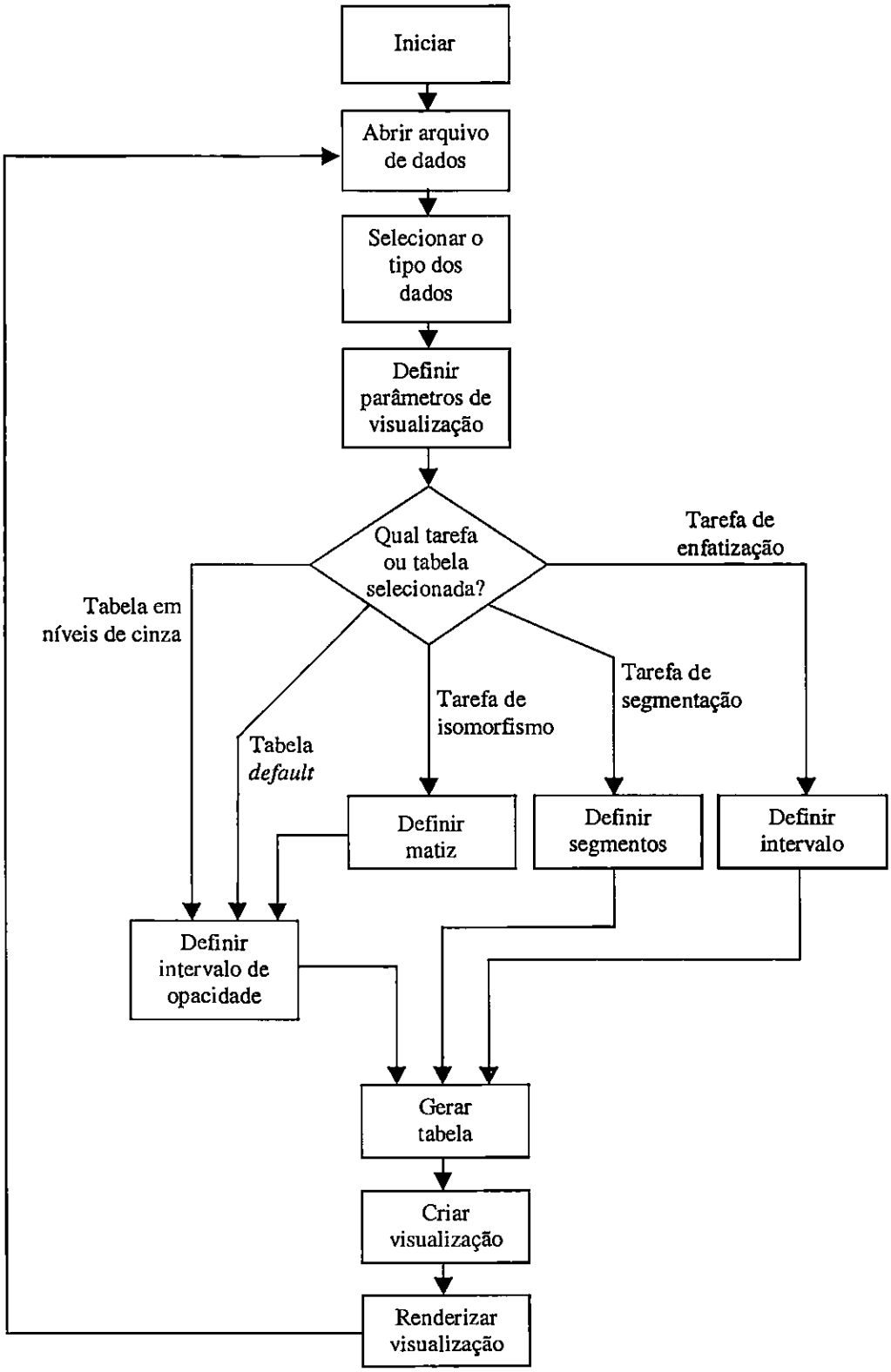


Figura 5.7 - Esquematização simplificada do funcionamento do aplicativo.

Directing Volume Rendering) e extração de isosuperfícies (visualização de superfícies extraídas do volume em valores específicos), em conjunto com a técnica de mapeamento por cores usando a *vtkRuleLookupTable*. Como já vimos, o *VTK* é uma biblioteca que possui uma vasta gama de funcionalidades e, apesar de ser interessante o desenvolvimento de uma interface gráfica genérica para acesso a todas elas, tal solução não seria viável no escopo de um projeto de mestrado. Não é permitido ao usuário acrescentar funcionalidades a interface, que fornece uma seqüência de passos a serem seguidos para gerar uma visualização, sendo que um deles é a definição da tabela de cores. O diagrama da figura 5.7 descreve, de maneira geral e simplificada, como o aplicativo funciona.

A classe que controla os eventos e variáveis da janela principal (figura 5.8), denominada *CVtkRLTInterfaceDlg*, é responsável por manipular todas as informações especificadas pelo usuário nas janelas do aplicativo. Por exemplo, ao selecionar um arquivo, o usuário utiliza a janela ilustrada na figura 5.9, que é tratada pela classe *CDataSetDlg*; mas nada acontece até que o botão OK seja pressionado. Quando esse botão é acionado, a janela é automaticamente fechada e a classe *CVtkRLTInterfaceDlg* recebe e trata as informações especificadas pelo usuário. O tratamento de dos eventos na maioria da janelas ocorre de forma semelhante, exceto para a janela que contém a representação gráfica da tabela de cores gerada. A classe *CVtkRLTInterfaceDlg* é responsável apenas por apresentar a janela na tela. Os eventos que essa janela manipula são tratados por sua classe, denominada *CTabelaDlg*. Incluímos nesta classe as chamadas às funções do VTK, exceto aquelas referentes à leitura do arquivo do conjunto de dados, que são chamadas pela classe *CVtkRLTInterfaceDlg*, dentro da função *OnOpen()*.

A função principal, aquela que transfere as informações da visualização especificadas pelo usuário para uma instância da classe *CTabelaDlg*, está implementada na classe *CVtkRLTInterfaceDlg*. Ela implementa a maioria dos passos mostrados no diagrama da figura 5.7, exceto aqueles que se referem ao arquivo de dados e à geração da visualização. Essa função, denominada *SetRLTVariables()*, é chamada sempre que o usuário termina de especificar todos os parâmetros necessários para a geração de uma tabela. O código dessa função, em C++, pode ser visto no Apêndice A.

5.3.1. Funcionamento da Interface

A partir da janela principal do programa (figura 5.8), que é do tipo *modeless*, o usuário inicia o processo de geração de uma visualização. O primeiro passo é selecionar um arquivo contendo o conjunto de dados a ser visualizado. Isso é feito pressionando o primeiro botão à esquerda (figura 5.8). A janela da figura 5.9 é, então, apresentada. Somente após definido o arquivo dos dados é que o usuário inicia o processo de visualização.

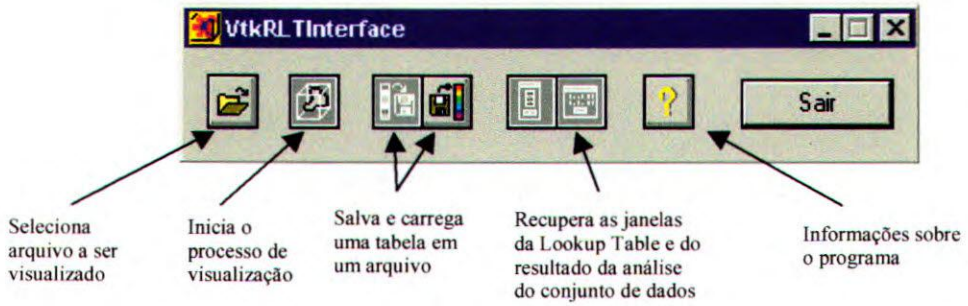


Figura 5.8 – Janela principal do vtkRLTInterface.

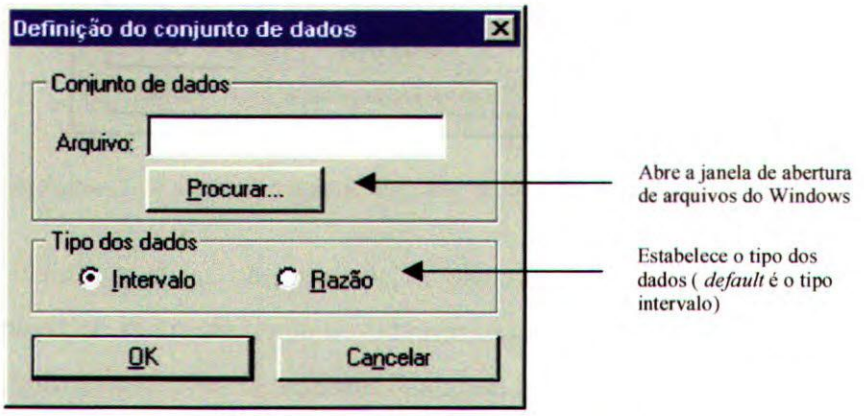


Figura 5.9 – Janela de definição do arquivo e do tipo dos dados.

A janela de definição do conjunto de dados é do tipo *modal* e oferece mais do que uma simples opção de escolha do arquivo. O tipo de dados contido no arquivo é muito importante no processo de visualização baseado em regras, como discutido na seção anterior. O tipo *default* é o tipo intervalo, por ser mais comum. Caso o usuário não selecione um arquivo, uma mensagem de erro é apresentada. Após a leitura dos dados, o aplicativo aciona os métodos da classe vtkStructuredPointsMetada para análise do conjunto de dados e

apresenta uma janela contendo informações sobre o mesmo para conhecimento do usuário, como apresentado na figura 5.10.

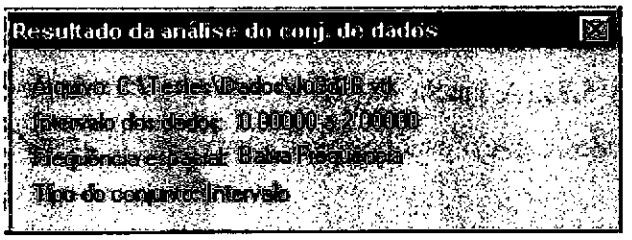


Figura 5.10 – Resultado da análise do conjunto de dados.

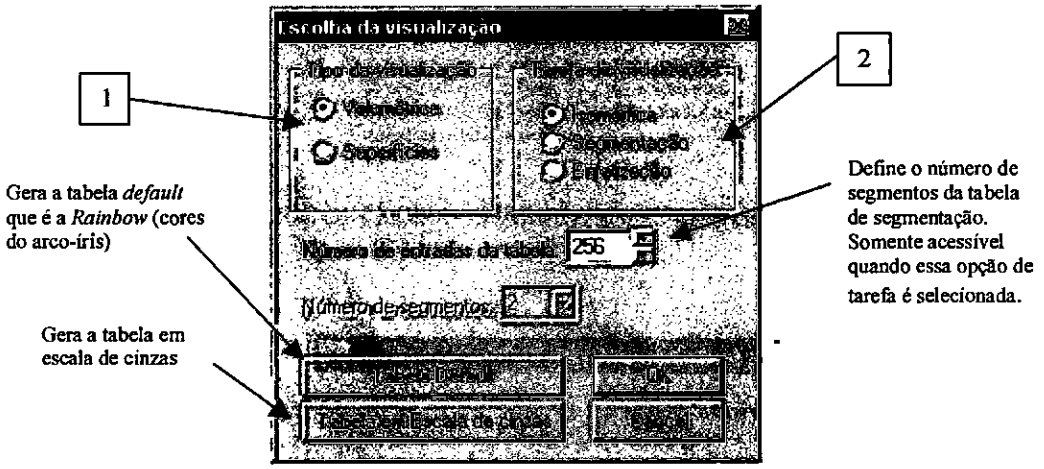


Figura 5.11 – Janela de definição do tipo e da tarefa de visualização desejada.

O início do processo de visualização se dá por meio do pressionamento do segundo botão à esquerda, que apresenta o diálogo ilustrado na figura 5.11. Nele, o usuário define o tipo de visualização desejada (visualização volumétrica direta ou por isosuperfícies) (regra 1 na figura), o tipo de tarefa de visualização desejada (regra 2 na figura). Tem-se as 3 opções de tarefas e tabelas associadas, além da possibilidade de gerar uma tabela *default* e uma em escala de cinzas. O diálogo permite definir o número de entradas da tabela que será gerada, que equivale ao número de cores da tabela, sendo 255 o máximo permitido. Caso a opção de tarefa seja a segmentação, um diálogo para definição do número de segmentos torna-se acessível, que permite gerar uma tabela contendo de 2 a 7 segmentos.

5.3.1. Renderização Volumétrica Direta

Caso a tarefa selecionada tenha sido a de Isomorfismo, ou a tabela *default* , ou ainda a escala de cinzas, o diálogo na figura 5.12 é apresentado. Nele, o usuário determina os valores das opacidades inicial e final da tabela, que podem variar de 0,0 (zero) a 1,0 (um). Caso o usuário deseje que a opacidade seja constante ao longo de toda tabela, basta definir o início e o fim do intervalo com o mesmo valor. O *default* para esse diálogo é o apresentado na figura, com opacidade constante em 1,0.

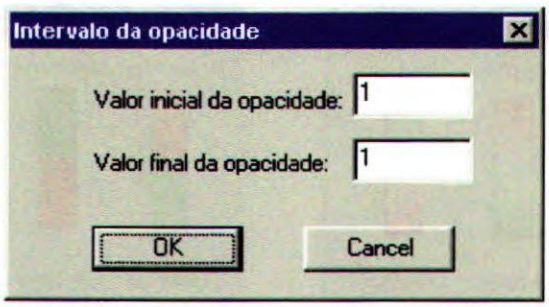
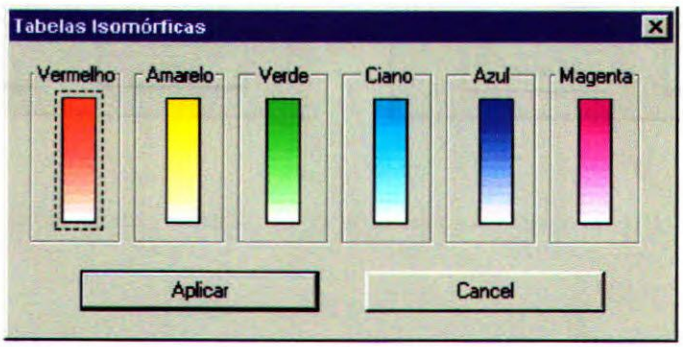
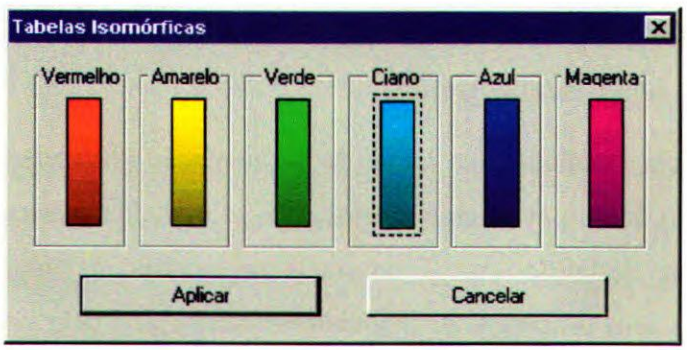


Figura 5.12 – Definição do intervalo de opacidade da tabela.



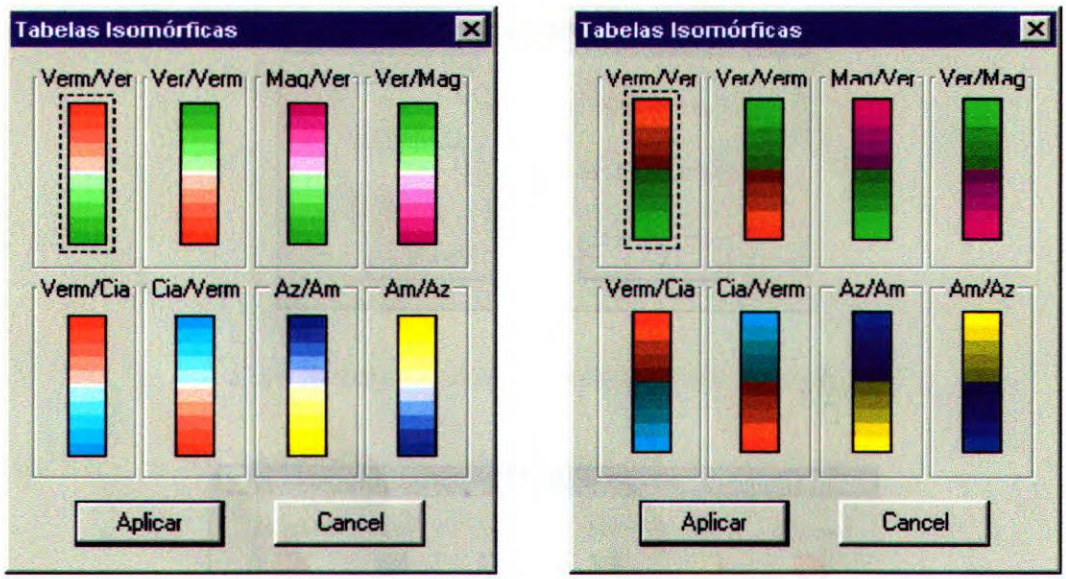
(a)



(b)

Figura 5.13 – Opções de tabelas para tarefas de isomorfismo para dados do tipo intervalo.

A partir daí, caso a tarefa selecionada seja a de isomorfismo, será apresentada ao usuário uma das 4 opções de diálogo contendo as tabelas disponíveis nessa categoria. A decisão de qual diálogo será apresentado depende do tipo e da freqüência espacial dos dados. Caso os dados sejam do tipo intervalo e de baixa freqüência o diálogo da figura 5.13a é apresentado; se o tipo é intervalo e a freqüência é alta o diálogo apresentado é o da figura 5.13b. Caso os dados sejam do tipo razão e de baixa freqüência o diálogo da figura 5.14a é apresentado; e se a freqüência é alta o diálogo da figura 5.14b é apresentado.



(a) (b)

Figura 5.14 – Opções de tabelas para tarefas de isomorfismo para dados do tipo razão.

Esses diálogos finalizam o processo de geração de uma visualização por renderização volumétrica direta assumindo uma tarefa isomórfica. Ao pressionar o botão "Aplicar" em quaisquer das quatro janelas, o usuário autoriza a criação da tabela e apresenta a janela que contém sua representação gráfica.

Caso seja escolhida uma tarefa de segmentação (segunda opção na regra 2 do diálogo da figura 5.11), basta um diálogo adicional. Esse diálogo, apresentado na figura 5.15, contém as opções de definição dos 7 possíveis segmentos de uma tabela, que ficam habilitados de acordo com o número de segmentos selecionado pelo usuário no diálogo da figura 5.11. Como opção inicial, o intervalo de variação dos dados foi dividido em intervalos de tamanhos iguais, de acordo com o número de segmentos escolhido pelo usuário. Esses

valores de início e fim de segmentos podem ser alterados, desde que estejam dentro o intervalo dos dados. O usuário deve especificar os valores iniciais, a opacidade e a cor desejada para cada intervalo. Ao ser selecionada a cor para um intervalo, o retângulo rotulado "Cor" que aparece antes da paleta de cores é colorido com a cor selecionada, possibilitando ao usuário visualizar as cores associadas aos intervalos.

Assim como na visualização isomórfica, ao pressionar o botão "Aplicar", o usuário continua o processo de visualização gerando a tabela e apresentando na tela a janela que contém a sua representação gráfica.

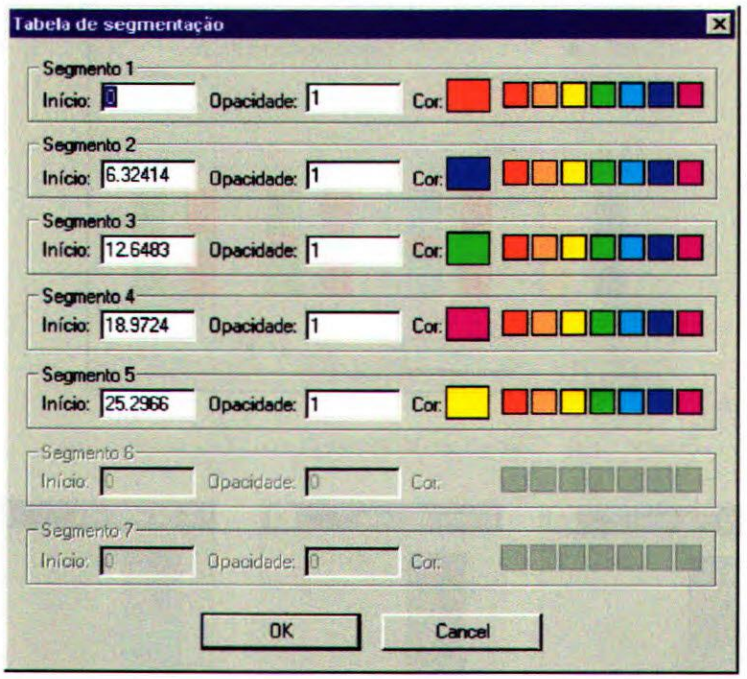


Figura 5.15 – Diálogo de definição dos segmentos da tabela de segmentação.

Caso seja escolhida a tarefa de enfatização (terceira opção na regra 2 da figura 5.11), o diálogo apresentado ao usuário pode ser visto na figura 5.16. Nesse diálogo, todas as opções de tabelas possíveis são apresentadas, sem restrição quanto ao tipo dos dados. O usuário deve especificar o intervalo que deseja enfatizar, e as opacidades do intervalo enfatizado e do restante da tabela. A janela apresenta as oito opções de tabelas para tal tarefa. Estas opções estão agrupadas segundo o par de matizes utilizado. Apenas uma delas pode ser selecionada. Caso o usuário pressione mais de um botão, será utilizada a opção selecionada por último.

As janelas de diálogo que apresentam as opções de tabelas para uma dada tarefa de visualização são as últimas a serem mostradas antes da apresentação da representação gráfica da tabela gerada, na janela denominada "Tabela Utilizada". Essa representação pode ser editada interativamente pelo usuário. A figura 5.17 apresenta três diálogos contendo, respectivamente, uma tabela gerada para a tarefa de isomorfismo (figura 5.17a), uma tabela de níveis de cinza (figura 5.17b), e uma tabela *default* (figura 5.17c). Esse diálogo permite ao usuário ver a tabela gerada, e decidir por aplicá-la ao conjunto de dados ou alterá-la. No primeiro caso, o botão "Aplicar" deve ser pressionado.

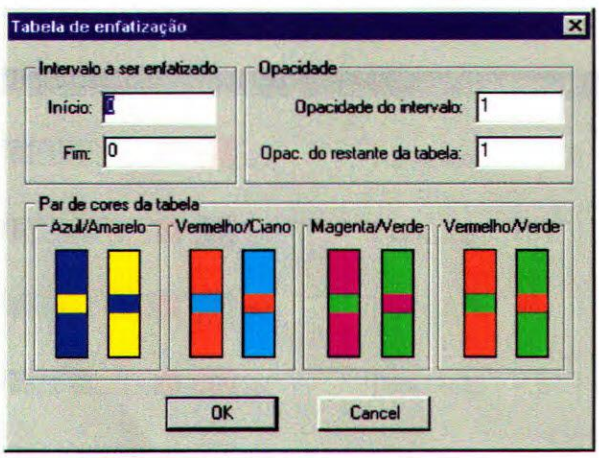
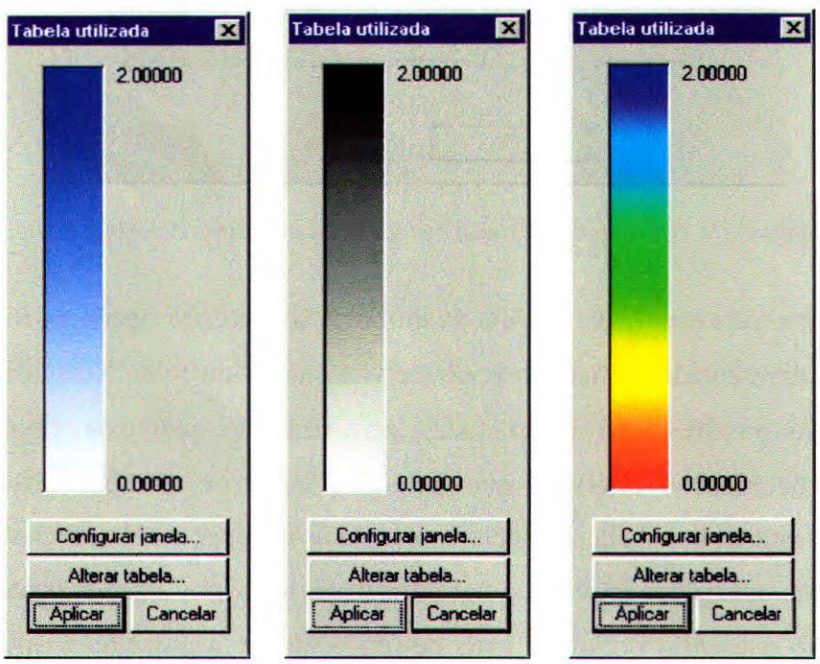


Figura 5.16 – Diálogo de definição da tabela de enfatização.



(a) (b) (c)

Figura 5.17 – Diálogo que contém a tabela gerada.

Muitas vezes, as tabelas geradas não estão exatamente na forma desejada pelo usuário. É importante deixar o controle com o usuário e, por isso, disponibilizamos a opção de alterar a tabela criada. A alteração pode ser feita pressionando o botão "Alterar Tabela...", que apresenta o diálogo da figura 5.18, no qual os parâmetros referentes a brilho, saturação, opacidade e matiz podem ser alterados para um intervalo de valores de dados definido pelo usuário. Caso o usuário queira alterar mais de um intervalo, basta utilizar novamente o botão "Alterar Tabela...".

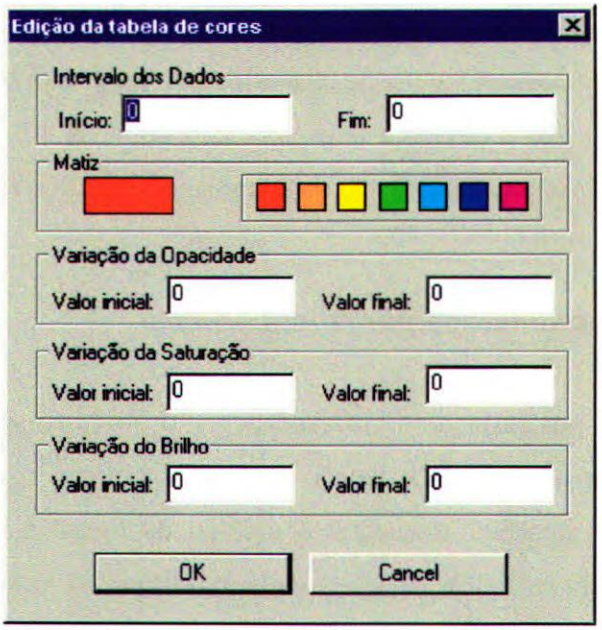


Figura 5.18 – Diálogo de alteração da tabela

As outras opções do diálogo apresentado na figura 5.17 permitem a configuração da janela de visualização e a geração da visualização propriamente dita. Pressionando o botão "Configurar janela.." o diálogo ilustrado na figura 5.19 é apresentado. Nele o usuário pode configurar a cor do fundo da janela de visualização, a cor dos *outlines*, que são as linhas que delimitam o volume de dados e o tamanho do passo de amostragem do *Ray Casting*. O tamanho do passo interfere muito no tempo de renderização da imagem e na sua qualidade final. Um passo muito grande diminui a resolução e, conseqüentemente, a qualidade imagem final.

Ao pressionar o botão "Aplicar" no diálogo da figura 5.17, é feito o *rendering* (apresentação na tela) do volume (ou das isosuperfícies) do conjunto de dados.

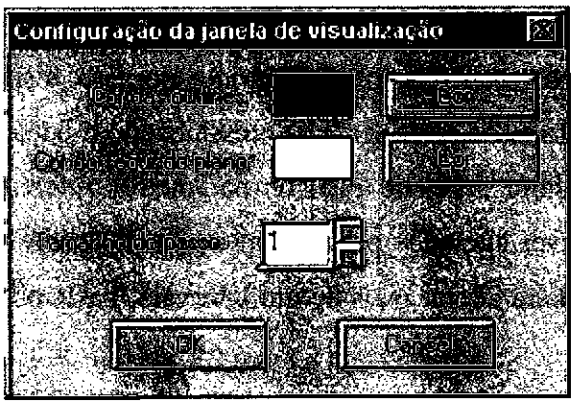


Figura 5.19- Configuração da janela de visualização.

Uma vez criada a janela de visualização, essa não deve ser fechada até a finalização do aplicativo. Isso é devido a uma limitação do VTK 1.3, que já foi solucionada em versões posteriores. Caso o usuário feche essa janela, o aplicativo será finalizado.

5.3.2. Renderização Volumétrica por Isosuperfícies

Em caso de visualização por extração de isosuperfícies, o processo segue o mesmo percurso usado na geração das tabelas para renderização volumétrica direta, com uma única diferença. Ao final do processo, e antes de apresentar o diálogo da figura 5.17, um novo diálogo é mostrado ao usuário (figura 5.20). Nele, o usuário define os valores de contorno e as opacidades para as isosuperfícies. Os contornos selecionados são armazenados em uma lista que é apresentada ao usuário, sendo o número máximo de isosuperfícies igual a 10 (este valor pode ser alterado futuramente). O usuário pode alterar essa lista, bastando, para isso, selecionar o item a ser alterado e utilizar o botão correspondente à operação desejada (adicionar, alterar, remover). Após esse diálogo, é apresentado o diálogo com a representação gráfica da tabela de cores gerada, já apresentado na figura 5.17, com as mesmas propriedades que este apresenta para a visualização volumétrica direta.

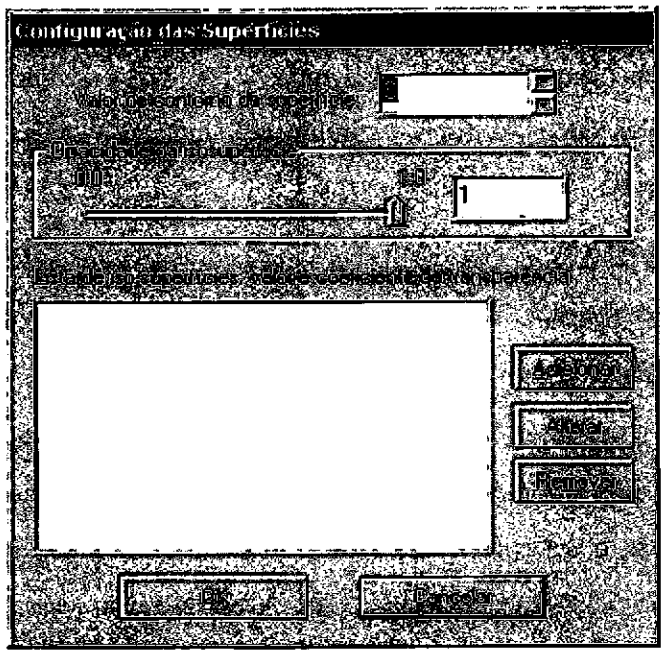


Figura 5.20 – Diálogo de seleção das isosuperfícies.

5.3.3. Características da Implementação

A seguir apresentamos alguns limitações deste trabalho, algumas delas remanescentes da implementação.

1) Alguns parâmetros não foram definidos por Rogowitz e outros [Rog93a; Rog93b], como o tamanho dos intervalos nas tabelas de segmentação e de enfatização.

2) O trabalho original dos autores citados no parágrafo anterior classifica os dados em quatro tipos: intervalo, razão, ordinal e nominal. Nessa implementação, tabelas são sugeridas apenas para os dois primeiros tipos.

3) Para a tarefa de isomorfismo com dados do tipo razão, as tabelas possuem como limiar o valor central da tabela, que corresponde ao zero. O ideal seria que o usuário pudesse escolher o limiar.

4) Apenas um intervalo de enfatização pode ser especificado, e o tamanho desse intervalo não é verificado em relação ao tamanho do intervalo de dados, conforme sugerido pela taxonomia proposta por Rogowitz e outros [Rog93a, Rog93b] descrita no capítulo 3.

5) As tabelas de cores geradas possuem um tamanho máximo de 256 entradas por ser esse o limite de entradas das tabelas geradas pelo VTK. Nas versões atuais do VTK, esse conceito de tabelas de cores foi substituído pelo conceito mais genérico de função de transferência de cor.

6) Essa versão do *VtkRLTInterface* utiliza a versão do módulo *vtkRuleLookupTable* para o VTK 1.3. Essa versão está muito ultrapassada, sendo necessária a sua atualização para, no mínimo, a versão 2.3.

7) Uma vez gerada a visualização, o usuário não deve fechar a janela do VTK que contém a imagem. Essa é uma limitação do próprio VTK 1.3, que permite apenas uma janela de renderização. Caso o usuário deseje criar outras visualizações, deve executar os passos descritos na seção 5.3.2 sem fechar a janela. A visualização será atualizada nessa mesma janela.

8) O VTK possui um amplo conjunto de técnicas de visualização que podem ajudar o usuário a entender os seus dados. No entanto, o *VtkRLTInterface* incorpora DVR e extração de isosuperfícies com mapeamento por cores. Em trabalhos futuros, deve-se estudar a incorporação de outras técnicas de visualização escalar e vetorial, ou pensar em uma forma de generalizar a interface.

9) A configuração da janela de visualização permite apenas especificar a cor do fundo da janela, a cor dos *outlines*, e o tamanho do passo de amostragem do *Ray Casting*. Seria interessante que o usuário também pudesse configurar o posicionamento e o movimento de câmeras, por exemplo.

10) Os tratamentos de alguns erros não foram implementados. Preocupamos primeiramente em desenvolver a interface considerando apenas a sua utilização correta. À medida em que os erros foram sendo identificados, procuramos tratá-los. No entanto, o aplicativo não foi maciçamente testado e alguns erros devem surgir.

5.4. Considerações Finais

Neste capítulo descrevemos o trabalho desenvolvido por Tutida [Tut98] durante o seu programa de mestrado, que resultou na *vtkRuleLookupTable*. A seguir, descrevemos uma

nova implementação desta classe, reestruturada de acordo com a filosofia do VTK e as alterações incluídas nesse trabalho. Descrevemos a implementação da interface para o módulo desenvolvido, denominada *VtkRLTInterface*, como utilizá-la e suas limitações.

No próximo capítulo apresentamos alguns casos de testes executados utilizando a interface e as conclusões deste trabalho.

Capítulo VI

Testes

6.1. Considerações Iniciais

Os testes realizados tiveram por objetivo verificar o funcionamento da interface e sua integração aos módulos já criados, bem como das alterações introduzidas no módulo *vtkRuleLookupTable*. Foram geradas tabelas em escala de cinzas, a tabela *default* do VTK e as tabelas de isomorfismo, segmentação e enfatização. Foram utilizados três conjuntos de dados, dois volumétricos e um bididmensional, do tipo escalar. Todos consistem de malhas estruturadas regulares, representadas por instâncias da classe *vtkStructuredPoints*. Nos testes, fora geradas imagens por renderização volumétrica ou extração de superfícies usando as classes do VTK 1.3 que implementam tais técnicas, usando diferentes tabelas de cores disponibilizadas por meio da interface.

Este capítulo apresenta na próxima seção os resultados dos testes obtidos com a utilização das diferentes tabelas de cores aos conjuntos de dados.

6.2. Casos de Teste

6.2.1. Dados Marítimos

Este caso de teste consiste de um conjunto de dados com diferentes níveis de concentração da substância química *Dimethyl Sulphide* (DMS) nas águas do Mar do Norte ao longo de um período de tempo, obtidas por medições [Day97].

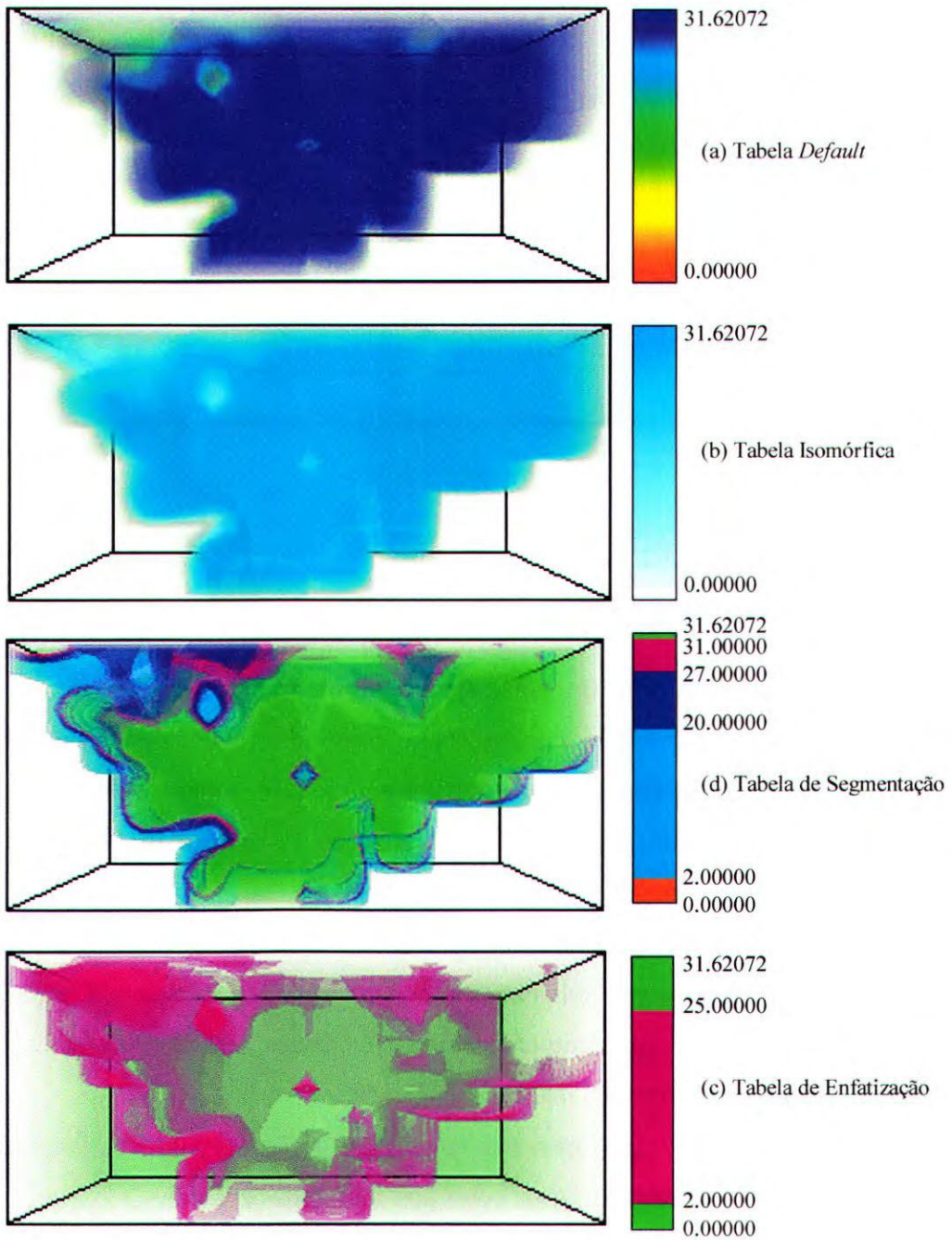


Figura 6.1 - Visualizações do volume de dados marítimos utilizando diferentes tabelas de cores.

Esse conjunto de dados foi classificado como sendo de baixa freqüência espacial e do tipo intervalo. O intervalo de variação dos dados é [0,0; 31,62072]. A figura 6.1 ilustra os resultados da utilização de 4 tabelas de cores diferentes na visualização desse conjunto, todas elas com 256 entradas. Os dados visualizados por renderização volumétrica direta, usando a classe *vtkVolume* do VTK. A análise, nesse caso, tinha o objetivo de detectar alterações temporais e posicionais na distribuição dessa substância nas águas do Mar do Norte.

A figura 6.1a ilustra uma visualização obtida com a aplicação da tabela de cores *default* (*rainbow*) do VTK, cuja opacidade varia de 0,0 a 0,5. Essa tabela não é muito útil para ajudar a detectar alterações no valores pois há muitos valores de dados concentrados em um intervalo muito próximo ao valor limite do conjunto de dados. Uma tabela isomórfica também não se mostra muito útil nesse caso em que o objetivo é detectar alterações. Uma tabela desse tipo foi criada (figura 6.1b) para efeito de testar a interface, mas a variação monotônica de um único matiz gera o mesmo problema ocorrido na tabela *default*.

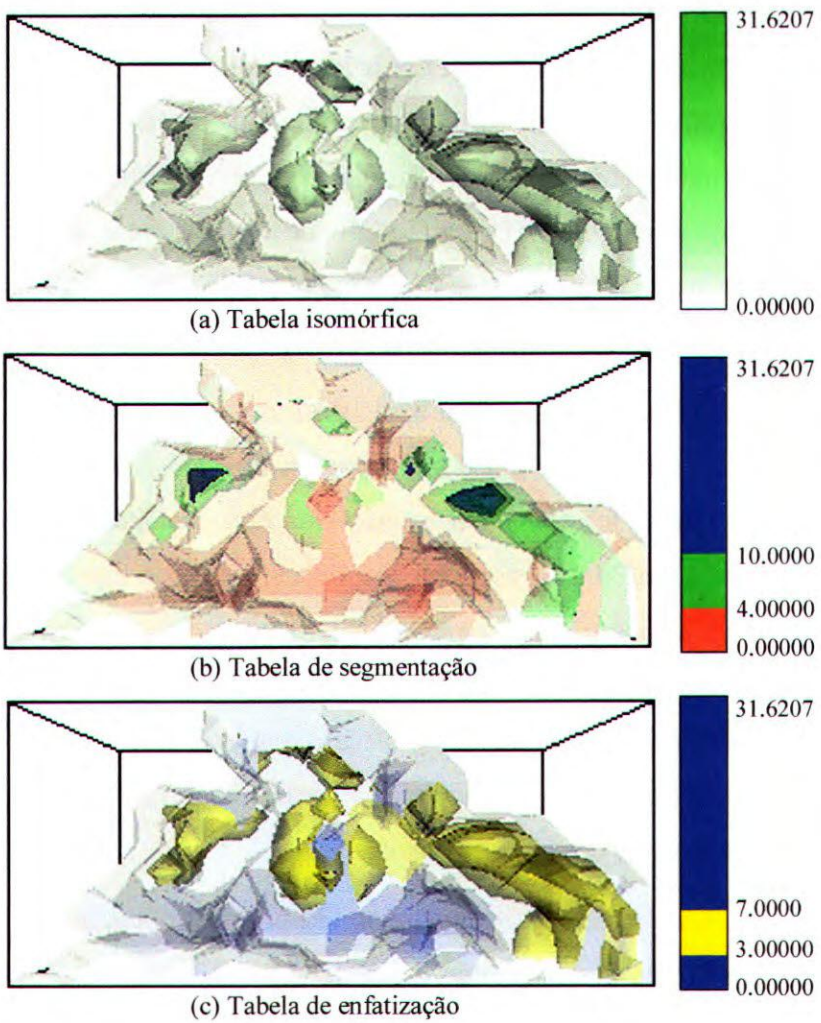
A figura 6.1c mostra uma visualização que usa uma tabela com 5 segmentos representados pelas cores vermelho, ciano, azul, magenta e verde. A cor vermelha foi associada aos valores na faixa de 0,0 a 2,0 que foram renderizados totalmente transparente (opacidade 0,0,). A cor ciano foi associada a um valor de opacidade 0,3 e utilizada para representar valores entre 2,0 e 20,0. A cor azul tem opacidade 0,3 e representa valores entre 20,0 e 27,0. A cor magenta possui valor de opacidade 0,3 e representa valores entre 27,0 e 31,0 e finalmente, a cor verde possui valor de opacidade 0,1 e representa valores entre 31,0 e 31,6027. A vantagem de usar essa tabela é que o usuário tem liberdade de definir os intervalos de interesse de forma a ressaltar as variações que deseja identificar. Com a interface, é possível redefinir os intervalos e observar os resultados.

A figura 6.1d mostra o resultado obtido com a aplicação de uma tabela de enfatização usando o par verde-magenta, cujo intervalo enfatizado corresponde aos valores de 2,0 a 25,0 em magenta, com valor de opacidade 0,3. O restante da tabela foi mapeado na cor verde, com valor de opacidade 0,02. Novamente, a possibilidade de alterar a regra enfatizada e seus parâmetros facilita a tarefa de atingir uma solução adequada.

A figura 6.2 apresenta visualizações do mesmo conjunto de dados utilizando extração de isosuperfícies e diferentes tabelas de cores. Foram geradas 3 isosuperfícies que

correspondem aos valores de concentração iguais a 1,0, 4,0 e 12,0, com opacidades 0,1, 0,3 e 0,5, respectivamente.

A figura 6.2a apresenta o resultado da aplicação de uma tabela isomórfica com a cor verde às 3 superfícies. Na figura 6.2b apresentamos o resultado da utilização de uma tabela com 3 segmentos representados pelas cores vermelho, verde e azul. A cor vermelha corresponde ao segmento que varia de 0,0 a 4,0. A cor verde corresponde aos valores de 4,0 a 10,0. A cor azul corresponde aos valores de 10,0 a 31,6207.



(a) Tabela isomórfica

(b) Tabela de segmentação

(c) Tabela de enfatização

Figura 6.2 - Visualizações de superfícies de dados marítimos utilizando diferentes tabelas de cores.

A figura 6.2c ilustra o resultado da utilização de uma tabela de enfatização cujo intervalo enfatizado corresponde aos valores entre 3,0 e 7,0. O restante da tabela foi mapeado na cor azul.

6.2.2. Dados de Neurônios

O conjunto de dados utilizado neste caso de teste foi obtido a partir da síntese da forma de estruturas neurais. A síntese é realizada por gramáticas estocásticas que operam sobre uma série de regras de produção bem definidas, derivadas a partir de caracterizações estatísticas e fractais de cada tipo de neurônio [Ces97 apud Tut98b][1]. Esse conjunto de dados tridimensional foi cedido pelo grupo de Visão Cibernética do IFSC-USP (Instituto de Física de São Carlos da Universidade de São Paulo). O conjunto de dados original consiste em uma descrição do neurônio na forma de uma poligonal bi ou tridimensional definida em um espaço discreto. Nesse espaço discreto, valores 0 ou 1 são colocados para indicar a presença ou ausência do neurônio, respectivamente. Os dados foram processados com a aplicação de uma Gaussiana, gerando valores escalares entre 0 e 1. Para manipulação no VTK, foi gerada uma malha regular, sendo que nos pontos introduzidos para tornar a malha regular foi inserido um valor inválido, no caso 2 [Tut98b].

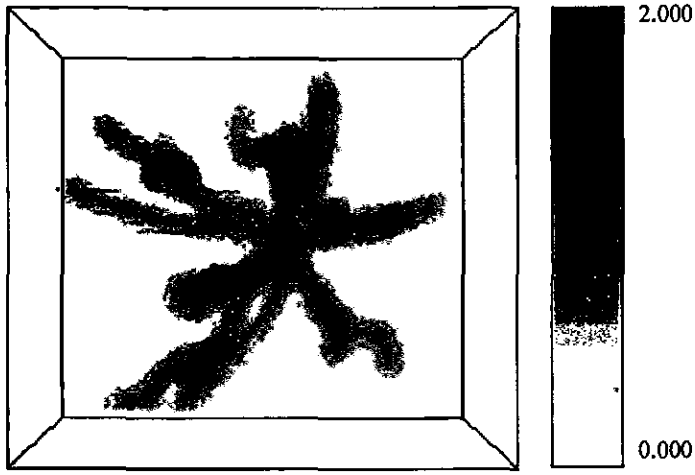


Figura 6.3 - Visualização de neurônio utilizando escala de cinzas.

A figura 6.3 mostra a imagem resultante da aplicação de uma escala de cinzas a um desses conjuntos de dados, visualizados por renderização volumétrica direta, com opacidade variando de 1,0 a 0,0, ao longo da tabela. A figura 6.4 mostra os resultados obtidos com a aplicação de diferentes tabelas de cores à renderização volumétrica direta dos dados. O conjunto de dados foi analisado como sendo de baixa freqüência espacial, do tipo intervalo, e

[1] Cesar Jr., R. M.; Costa, L. F. - "Semi-automated Dendogram Generation for Neural Shape Analysis", in **Proceedings of Sibgraph 97**, IEEE CS Press, 1997, pp. 147-154. Apud Tutida, S.M. – "Mapeamento por Cores Baseado em Aspectos de Percepção Visual", Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Tese, 1998.

contém valores escalares do tipo *float* variando de 0 a 2. Foram utilizadas tabelas de cores com 20 entradas.

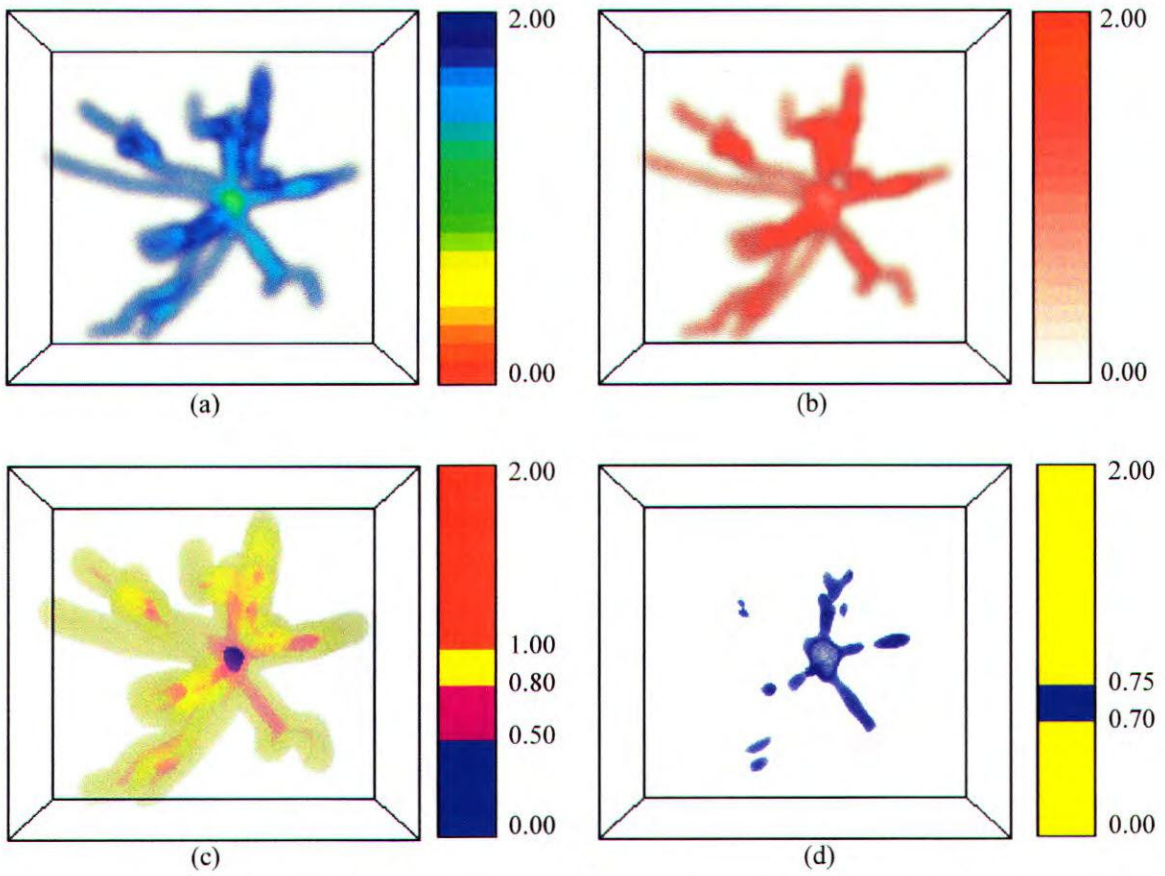
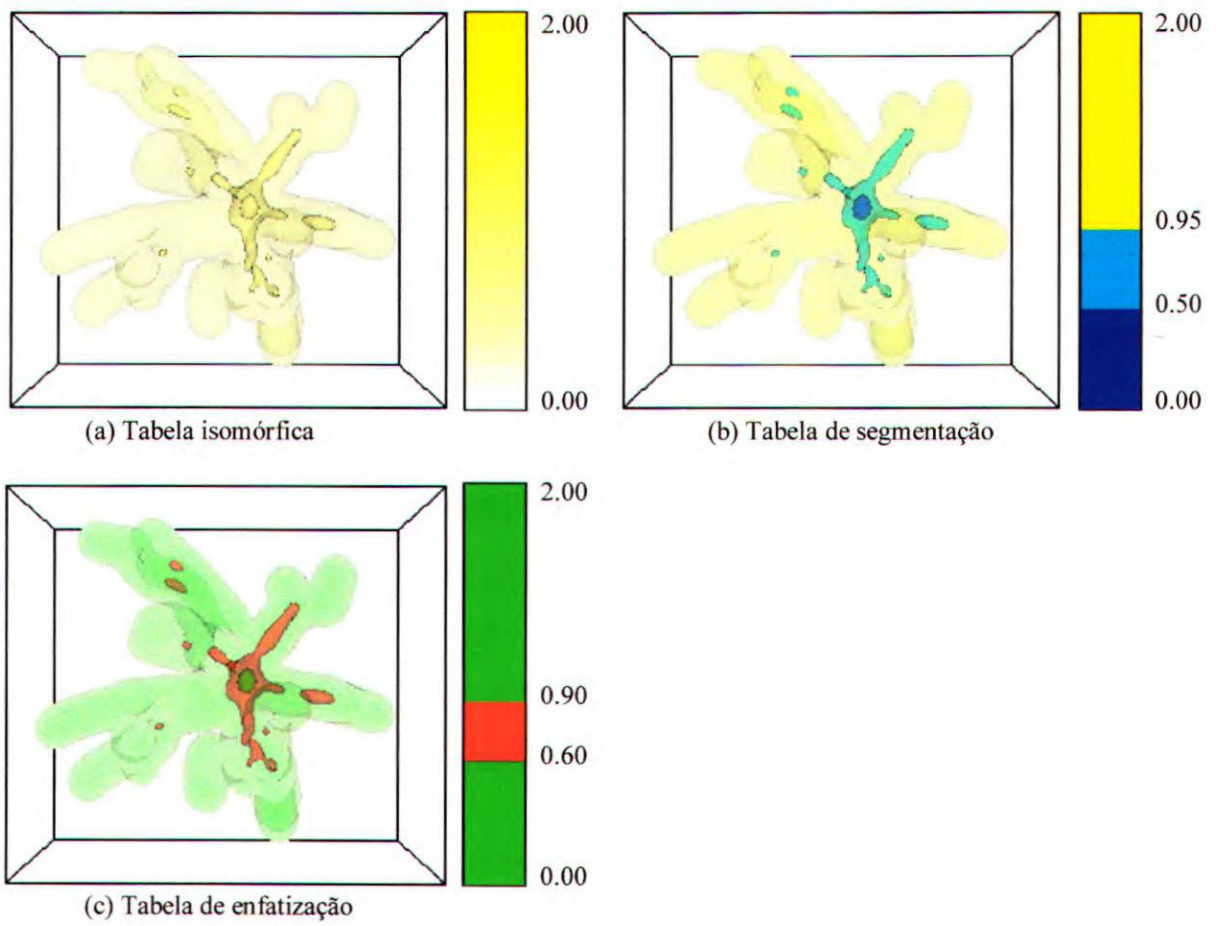


Figura 6.4 - Visualizações do volume do conjunto de dados de neurônios utilizando diferentes tabelas.

A imagem da figura 6.4a ilustra o resultado obtido com a aplicação da tabela *default* do VTK , a *rainbow*, com a opacidade variando linearmente no intervalo de 1,0 a 0,0. A figura 6.4b mostra o resultado obtido com a aplicação da tabela de cores isomórfica. Utilizamos uma tabela com a cor vermelha e com valores de opacidade decrescendo de 1,0 a 0,0 . Neste tabela e na anteriores, escolhemos esse intervalo de variação de opacidade porque o valor 2,0, que corresponde ao espaço em branco da figura, não constitui informação relevante, e é mapeado de totalmente transparente.

A figura 6.4c mostra a imagem resultante da aplicação de uma tabela de segmentação com as cores azul, magenta, amarela e vermelha. A cor azul possui valor de opacidade 0,5 e representa valores entre 0,0 e 0,50. A cor magenta possui valor de opacidade 0,3 e representa valores entre 0,50 e 0,80; e a cor amarela possui valor de opacidade 0,2 e representa valores entre 0,80 e 1,00; e a cor vermelha possui valor de opacidade 0,0 e representa o valor 2,0.

A figura 6.4d mostra o resultado da aplicação de uma tabela de enfatização, sendo que o intervalo enfatizado varia entre 0,70 a 0,75 e está representado pela cor azul com valor de opacidade 0.3. Os demais valores da tabela foram mapeados na cor amarela com valor de opacidade 0.0, tornando-os transparentes.



(a) Tabela isomórfica

(b) Tabela de segmentação

(c) Tabela de enfatização

Figura 6.5 - Visualizações de superfícies de neurônios utilizando diferentes tabelas.

A figura 6.5 mostra os mesmos dados visualizados por extração de superfícies utilizando diferentes tabelas de cores. Foram geradas 3 superfícies que contornam os valores 0,45, 0,8 e 1,0, do centro para fora, com opacidades 0,4, 0,3 e 0,2, respectivamente.

A figura 6.5a ilustra o resultado da utilização de uma tabela isomórfica usando a cor amarela. A figura 6.5b ilustra uma visualização gerada utilizando uma tabela de 3 segmentos, o primeiro é mapeado com a cor azul ao intervalo entre 0,0 e 0,50; o segundo com a cor ciano no intervalo entre 0,50 a 0,95; e o terceiro com a cor amarela para o intervalo de dados que varia entre 0,95 e 2,00. Na figura 6.5c foi utilizada uma tabela de enfatização com

o par de cores oponentes vermelho-verde, cujo intervalo enfatizado varia de 0,60 a 0,90 e foi mapeado com a cor verde.

6.2.3. Dados de Topografia

O terceiro caso de teste utilizou de dados topográficos do continente africano. Esse conjunto de dados bidimensional foi obtido da rede Internet e convertidos para o formato de arquivos do VTK. Esse conjunto de dados foi incluído nos testes porque a interface permite a visualização de conjunto de dados 2D. Para gerar uma visualização desse tipo, o usuário deve seguir os mesmos passos para a criação de uma visualização por renderização volumétrica direta.

A figura 6.6 ilustra a aplicação de diferentes tabelas cores a esse conjunto de dados, classificado como de baixa freqüência, do tipo intervalo e com valores escalares do tipo *float* variando entre 0,01176 e 0,92953. Foram utilizadas tabelas com 10 entradas.

A figura 6.6a ilustra a aplicação da tabela em escala de cinza, e a figura 6.6b ilustra a aplicação da tabela de cores *default* do VTK. A figura 6.6c mostra a imagem resultante da aplicação da tabela isomórfica. Nessa imagem, valores de saturação menores do vermelho correspondem a alturas menores e, valores de saturação maiores indicam alturas maiores.

Na figura 6.6d é ilustrada a aplicação de uma tabela com 4 segmentos. A cor vermelha representa valores entre 0,01176 e 0,2412, a cor amarela corresponde ao intervalo de 0,2412 a 0,4706, a cor azul corresponde ao intervalo de 0,4706 a 0,700 e a cor laranja representa valores entre 0,700 e 0,92953.

Na figura 6.6e apresentamos o resultado a aplicação de uma tabela de enfatização com o par de matizes vermelho e ciano. Nessa visualização procuramos enfatizar a regra correspondente ao intervalo mapeado em azul na figura 6.6d.

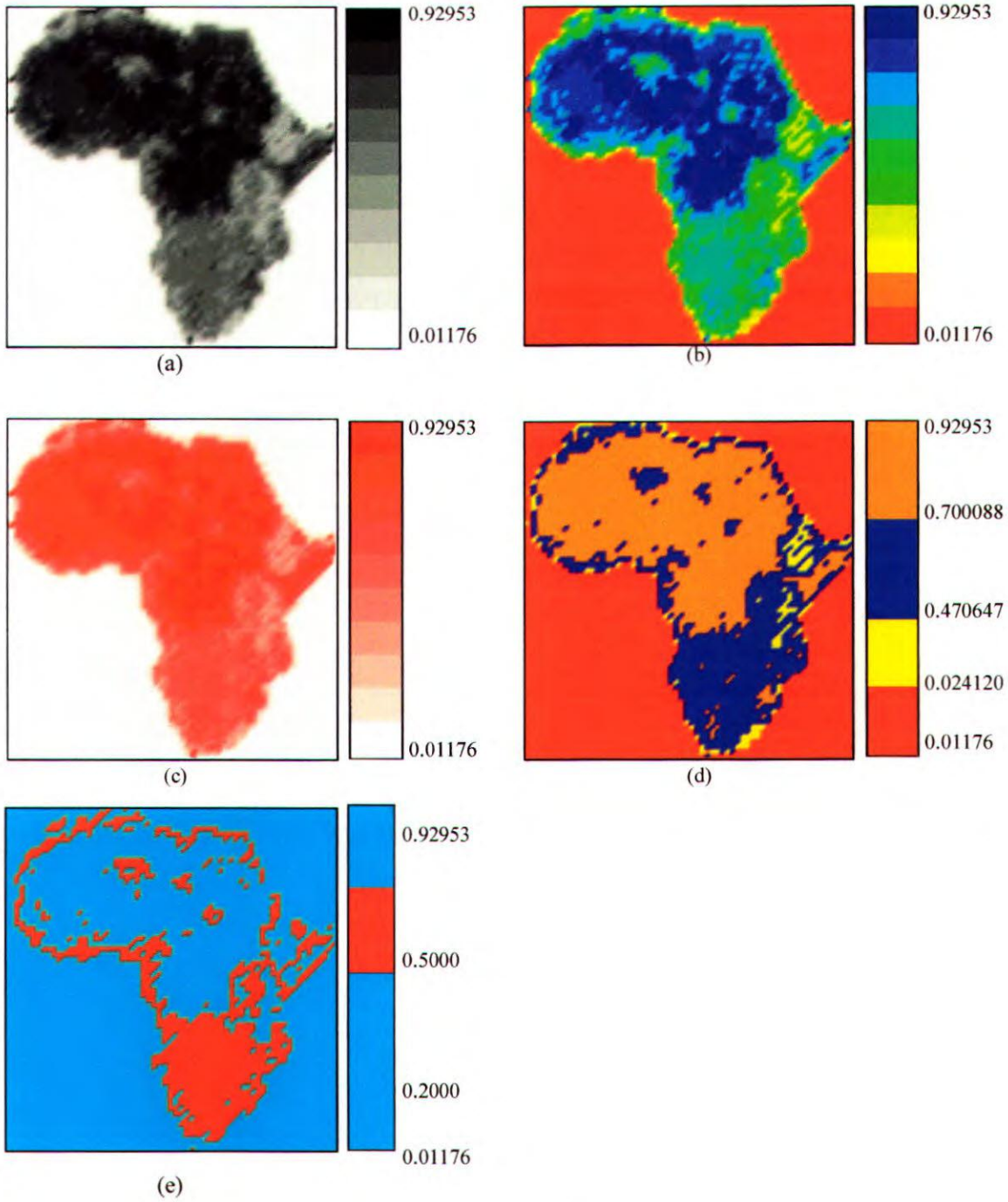


Figura 6.6– Visualização de dados topográficos utilizando diferentes tabelas de cores.

6.3. Considerações Finais

Utilizando a interface para gerar os resultados acima pudemos verificar a sua funcionalidade. De fato, por meio da interface, a geração de visualizações torna-se mais rápida e efetiva. Duas grandes vantagens da interface são: o usuário pode enxergar as opções de tabelas; e pode alterar essas tabelas como bem entender e visualizar a alteração antes de aplicar a tabela de cores ao conjunto de dados. Isso não é possível quando se usa o módulo *vtkRuleLookupTable*, além da necessidade de reescrita do trecho de código correspondente à especificação da tabela.

Capítulo VII

Conclusões

Os conjuntos de dados foram utilizados com o intuito de gerar as visualizações e obter o resultado da aplicação das tabelas a eles, de forma a verificar a correção do software. Os testes executados ajudaram na identificação de falhas na implementação e a ilustrar o potencial da abordagem de definição interativa de mapeamentos de cores, partindo de tabelas que podem ser efetivas em situações específicas.

A interface criada, aliada ao módulo *vtkRuleLookupTable*, agiliza o processo de geração de visualizações. Porém, requer do usuário conhecimento do funcionamento da técnica de mapeamento por cores. A interface permite ao usuário escolher a tabela a ser gerada, mas nem sempre as tabelas geradas pelo módulo são as mais adequadas eficácia total. Por exemplo, a tabela de isomorfismo com o matiz amarelo não transmite com clareza a variação dos valores dos dados, portanto, cuidado deve ser tomado com a utilização desse matiz. Outro fato detectado durante a realização dos testes se refere ao número de entradas da tabela. Uma escolha inadequada do número de entradas irá influenciar o resultado da visualização. Infelizmente, não foi feito um estudo da relação entre o intervalo de variação

dos dados e o número de entradas da tabela não foi feito, mesmo porque, o número de entrada no VTK (versões 1.0 e 1.3) é limitado a 256. Esse limite é facilmente ultrapassado pelo intervalo de variação dos dados e toma-se difícil estabelecer alguma regra que os correlacione.

A tarefa de segmentação sofreu alterações com a inclusão do matiz laranja no conjunto de cores possíveis para o mapeamento, com base na literatura mencionada capítulo 5. Como os autores especificaram os parâmetros necessários para a definição do matiz no modelo de cores usado, decidimos especificá-lo com sendo a metade da distância entre os matizes vermelho e amarelo, luminância e saturação máximos, no círculo de cores que representa a base do modelo HSV.

A interface possibilitou uma fácil interação na geração de visualizações usando mapeamentos por cores. O usuário gera suas imagens de forma rápida e sem a necessidade de programação. Para usuários mais experientes, contomamos o fato de o número de tabelas serem restritas àquelas fomecidas pelo módulo adicionando uma função que permite a alteração interativa de uma tabela sugerida pelo módulo.

A interface fomece os recursos básicos necessários para a geração de visualizações volume de dados escalares utilizando renderização volumétrica de dados e extração de superfícies, usando as classes oferecidas pelo VTK. Existem vários outros recursos que podem e devem ser adicionados. Por exemplo, a visualização de certos volumes não permite uma boa análise do conteúdo do interior do mesmo. Há vários recursos que podem ser inseridos ao aplicativo para auxiliar essa tarefa. Um deles seria a utilização de planos de corte que fomecessem ao usuário a informação contida numa fatia do volume. Um outro, um pouco mais arrojado, seria o acoplamento à interface do módulo de sonificação desenvolvido por Salvador [Sal98] que implementa uma sonda sonora para este tipo de análise. Um grande número de técnicas para auxiliar na visualização de dados multidimensionais, tais como *ribbons*, *streamlines*, *streamsurfaces*, *glyph* entre outros, podem também dar suporte à tarefa de geração de uma visualização.

Em aplicativos que visam auxiliar o usuário em alguma tarefa, toda ajuda que possa ser fomecida deve ao alcance. O *VtkRLTInterface* não fomece todos recursos para auxiliar o usuário em sua tarefa, concentrando se apenas nas opções de tabelas disponíveis. A criação de um arquivo que armazenasse os passos seguidos pelo usuário durante a utilização

do aplicativo seria de imensa utilidade. Com ele, o usuário poderia facilmente recordar os passos que executou ao gerar uma tabela específica.

O armazenamento de uma tabela em arquivo e sua posterior recuperação constituem uma característica importante da interface, possibilitando ao usuário gerar novamente uma visualização anterior ou mesmo utilizar essa tabela em outro conjunto de dados. Porém, essa reutilização deve ser cuidadosa, pois a interface não verifica a compatibilidade entre a tabela carregada a partir de um arquivo e o conjunto de dados ao qual ela está sendo aplicada, o que pode produzir visualizações completamente inadequadas.

Os testes iniciais realizados neste trabalho não avaliaram extensivamente a abordagem adotada e nem o conjunto de regras implementadas, mas serviram para mostrar o potencial da abordagem. Os resultados obtidos com a aplicação das diferentes tabelas de cores geradas pelo módulo de mapeamento baseado em regras confirmam que diferentes tipos de tabelas influenciam de modo significativo a imagem resultante. Futuramente, testes adicionais com usuários de um domínio de aplicação com o objetivo de avaliar a abordagem em contextos reais podem ser realizados.

Uma nova implementação do módulo *vtkRuleLookupTable* para a versão 2.0 do VTK foi implementada. No entanto, o VTK já está na sua versão 2.4 e há um esforço do grupo de Computação Gráfica de Processamento de Imagens do ICMC a fim de atualizarmos a versão do módulo para, no mínimo, a versão 2.3. O VTK, a partir da versão 2.0, alterou completamente os métodos de manipulação das tabelas de cores. Foi implementado um conceito mais genérico das tabelas de cores, denominado *vtkColorTransferFunciton*, para a visualização de volumes. A atualização da interface para uma versão mais recente é estritamente necessária para que possamos lançar mão de sua funcionalidade.

Há uma infinidade de recursos que podemos adicionar ao aplicativo e estamos cientes da precariedade do mesmo. Pudemos observar neste trabalho a necessidade de tal ferramenta no âmbito da pesquisa e análise científica e a potencialidade de desenvolvimento de novos recursos para o mesmo. Este trabalho representa um passo na direção de ferramentas que tentam auxiliar o usuário em seu trabalho, levando em consideração mais do que o próprio usuário seria capaz de perceber e sugerir.

Capítulo VIII

Referências Bibliográficas

[AVS99] AVS Application Visualization System (1999, Novembro). AVS Upgrades Flagship Product, AVS/Express 5.0. WWW: Advanced Visual System Inc, http://www.avs.com/company/pr/AVSExpress50.htm.

[AVS95] AVS Application Visualization System (1995, Janeiro). AVS Tutorial. WWW: http://www.cica.indiana.edu/cica/faq/avs/Index.htm.

[Ber95] Bergman, L. D.; Rogowitz, B. E.; Treinish, L. A. – "A Rule-based Tool for Assisting Colormap Selection", **Proceedings of the IEEE Computer Society Visualization '95 Conference**, Maio, 1995, pp. 118-125.

[Ber96] Bergman, L. D.; Rogowitz, B. E.; Treinish, L. A. (1996). A Rule-based Tool for Assisting Colormap Selection. WWW: IBM Research, http://almaden.ibm.com/dx/vis96/proceedings/PRAVDA/index.htm.

[Bok95] Boker, S.M. (1995, February). The Representation of Color Metrics and Mappings in Perceptual Color Space. WWW: University of Notre Dame Home Page, http://www.nd.edu/~sboker/ColorVision2/ColorVision2.html.

[Bou95] Bourgin, D. (1995, April). Color Spaces FAQ. WWW: The WELL, http://www.well.com/user/rld/vidpage/color_faq.html.

[Bro92] Brodlie, K. W.; Carpenter, L. A.; Earnshaw, R. A.; Gallop, J. P.; Hubbold, R. J.; Mumford, A. M.; Osland, C. D.; Quarendon, P. – "Scientific Visualization – Techniques and Application", Springer-Verlag, 1992.

[Day97] Day, A. M. et al - "Visualization and Sonification of Marine Survey Data", in *Visualization & Modeling*, Earnshaw et al (Eds), Academic Press, 1997, pp. 87 a 97.

[Ear97] Earnshaw, R.; Vince, J.; Jones, H. – Visualization & Modeling, Academic Press, 1997.

[Fai98] Fairchild, M. D. – "Color Appearence Models", Addison Wesley, 1998.

[Fod98] Fodra, R. D. – "Geração de Tabelas de Cores em Visualização", Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Relatório 1, Dezembro, 1998.

[Fol84] Foley, J.D.; VanDam, A – "Fundamentals of Interactive Computer Graphics", Addison Wesley, 1984.

[Fol90] Foley, J.D.; VanDam, A; Feiner, S.K.; Hughes, J.H.; Phillips, R.L. - "Computer Graphics: Principles and Practice", Addison Wesley, 1990.

[Fol94a] Foley, J.; Ribarsky, B. – "Next-Generation Data Visualization Tools", **Scientific Visualization – Advances and Challenges**, Academic Press, 1994, pp. 103-127.

[Fol94b] Foley, J.D.; VanDam, A; Feiner, S.K.; Hughes, J.H.; Phillips, R.L. - "Introduction to Computer Graphics", Addison Wesley, 1994.

[Fri91] Friedhoff, R. M.; Benzon, W. – "The Second Computer Revolution", W. H. Freeman and Company, 1991, pp. 10-45.

[Gal95] Gallagher, R. S. – "Computer Visualization – Graphics and Techniques for Scientific and Engineering Analisys", CRC Press, 1995.

[Gom94] Gomes, J.; Velho, L. – "Computação Gráfica: Imagem", IMPA/SBM, 1994.

[Gon78] Gonzalez, R. C.; Woods, R. E. – "Digital Image Processing", Addison Wesley Publishing Company, 1978.

[Gri95] Grinstein, G.; Levkowitz, H. (eds.) – Perceptual Issues em Visualization, Springer-Verlag, 1995.

[Gro94] Groβ, M. – "Visual Computing", Spring-Verlag, 1994.

[Hab90] Haber, R. B.; MacNabb, D. A. - "Visualization Idioms: A Conceptual Model for Scientific Visualization Systems", in Nielson, G. M. (ed) - in *Visualization in Scientific Computing*, ed. Nielson G. M. et al, IEEE Computer Graphic Press, 1990, pp. 74-93.

[Hea94] Hearn, D.; Baker, M. P. – "Computer Graphics", Prentice-Hall International, 1994.

[Hea96a] Healey, C. G.; Enns, J. T. – "A Perceptual Colour Segmentation Algorithm", Technical Report, TR-96-XX, Department of Computer Science, University of British Columbia, 1996.

[Hea96b] Healey, C. G. – "Choosing Effective Colours for Data Visualization", **Proceedings of IEEE Visualization 96**, IEEE CS Press, 1996, pp. 263-270.

[Hun78] Hunt, R. W. G. – "Color Terminology", **COLOR Research and Application**, vol. 3, Number 2, 1978.

[IBM96] IBM Visualization Data Explorer (1996). IBM Visualization Data Explorer User's Guide. WWW: IBM Research, http://www.almaden.ibm.com/dx/docs/html.

[IBM99] IBM Open Visualization Data Explorer (1990, Outubro). IBM Research Visualization Data Explorer – Home. WWW: IBM Research, http://www.research.ibm.com/dx/Index.html.

[IRI99] IRIS Explorer (1999, Janeiro). IRIS Explorer User's Guide. WWW: The Numerical Algorithms Group Ltd, http://www.nag.co.uk/visual/IE/iecbb/DOC/Nt/Doc/UG_NT.htm.

[Kau90] Kaufmann, A. – "3D Volume Visualization", Tutorial Note 12, **Eurographics Technical Report Series**, 1990, pp. 34.

[Kau93] Kaufmann, A.; Cohen, D.; Yagel, R. – "Volume Graphics", **IEEE CG&A**, July, 1993, pp. 51-64.

[Kru97] Kruglinski, D. J. – "Inside Visual C++", 4ª edition, Microsoft Press, 1997.

[Lan91] Lansdowns, J. – "Visual Perception and Computer Graphics", in Earnshaw, R. A. (ed) – "Fundamental Algorithms for Computer Graphics", **The Proceedings of NATO Advanced Study**, 1991.

[Lev92] Levkowitz, H.; Herman, G. T. – "Color Scales for Image Data", **IEEE Computer Graphics and Applications**, Vol. 12, Number 1, January, 1992, pp. 72-80.

[Lev97] Levkowitz, H. – "Color Theory and Modeling for Computer Graphics, Visualiztion, and Multimedia Applications", Kluwer Academic Publishers, 1997.

[Mcc87] McCormick, B. H.; DeFanti, T. A.; Brown, M. D. (eds.) – "Visualization in Scientific Computing", **Computer Graphics**, vol.21, nº 6, November, 1987.

[Mey80] Meyer, G.W.; Greenberg, D.P. – "Perceptual Color Spaces for Computer Graphics", ACM, 1980, pp. 254-261.

[Mey86] Meyer, G.W. – "Tutorial on Color Science", The Visual Computer, Springer-Verlag, 1986, pp. 278-290.

[Min95] Minghim, R. – **On Sound Support for Visualization**, PhD. Thesis, University of East Anglia, Norwich, UK, March, 1995, pp. 273.

[Mon98] Montgomery, G. (1998, Julho). Breaking the Code of Color. WWW: Seeing the World, http://www.hhmi.org/senses/b/b110.htm.

[Oli97] Oliveira, M. C. F.; Minghim, R. – "Uma Introdução à Visualização Computacional", **XVII Jornada de Atualização em Informática**, Agosto, 1997, pp. 85-127.

[Ous97] Ousterhout, J. K. – "Tcl and the TK Toolkit", Adisson Wesley, 9th edition, 1997.

[Owe96] Owen, G. S. – "HyperVis – Teaching Scientific Visualization Using Hypermedia", ACM Siggraph Education Commitee, 1996 (http://www.cs.gasou.edu/~xuc/hypervis/hypervis.htm).

[Rhe95] Rheigans, P.; Landreth, C. – "Perceptual Principles for Effective Visualizations" in Grinstein, G.; Levkowitz, H (ed) – "Perceptual Issues in Visualization", Springer-Verlag, 1995, pp. 59-73.

[Rob85] Robertson, P.K.; O'Calhagham, J.F. – "The application of Scene Synthesis Techniques to the Display of Multidimensional Image Data", ACM Transactions on Graphics, Vol.4, No 4, 1985, pp. 247-275.

[Rob88] Robertson, P. K. - "Visualization Color Gamuts: A User Interface for the Effective Use of Perceptual Color Spaces em Data Displays", IEEE Computer Graphics and Application, 1988, pp. 50-64.

[Rob91] Robertson, P. K. – "A Methodology for Choosing Data Representations", **IEEE Computer Graphics and Applications**, Volume 11, Number 3, May, 1991, pp. 56-66.

[Rog93a] Rogowitz, B. E.; Treinish, L. A. – "Data Structured and Perceptual Structures", **Proceedings of the SPIE/SPSE Symposium on Electronic Imaging**, Vol. 1913, February, 1993, pp. 600-612.

[Rog93b] Rogowitz, B. E.; Treinish, L. A. – "Na Architecture for Rule Based Visualization", **Proceedings of the IEEE Computer Society Visualization '93 Conference**, October, 1993, pp. 236-243.

[Rog96] Rogowitz, B. E.; Treinish, L. A. – "How not lie with visualization", 1996, pp. 1-11, (www.almaden,ibm.com/dx/vis96/proceedings/PRAVDA.index.html).

[Rum91] Rumbaugh, J. et al – "Object-Oriented Modeling and Design", Prentice-Hall, 1991.

[Sal98] Salvador, V. C. L –"Sonificação para Apoio a Tarefas de Visualização", Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo, Tese, 1998.

[Sch99] Schröeder, W.; Martin, K.; Lorensen, B. - "The Visualization Toolkit - An Object-Oriented Approach to 3D Graphics", 2 edition, Prentice-Hall, 1999.

[Sen94] Senay, H.; Ignatius, E. – "A knowledge-Based System for Visualization Design", **IEEE Computer Graphics and Applications**, Vol. 14, Number 6, November, 1994, pp. 36-47.

[Taj83] Tajima, J. – "Optimal Color Display Using Uniform Color Scale", NEC Research & Development, No 70, July, 1983.

[Tul95] Tulloch, K. (1995). Data Visualization, Thesis. WWW: University of New South Wales, http://www.geog.unsw.edu.au/~kirsty/.

[Tut98a] Tutida, S. M. - "Estudo de Aspectos Percepção Visual de Cores em Visualização de Dados Científicos", Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo, Relatório Fapesp, no 2, 1998.

[Tut98b] Tutida, S.M. – "Mapeamento por Cores Baseado em Aspectos de Percepção Visual", Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Tese, 1998.

[Wat93] Watt; A. – "3D Computer Graphics", Second Edition, Addison Wesley, 1993, pp. 427-429.

Apêndice A

Este apêndice contém a função *SetRLTVariables*() que consiste na principal função da interface, sendo responsável por gerar a tabela especificada pelo usuário. Onde *dlgLut* é uma instância da classe *CTabelaDlg*.

```
void CVtkRLTInterfaceDlg::SetRLTVariables()
{
 dlgLut.m_flagCreated = FALSE;  ~ //flag para verificar se janela criada
 dlgLut.m_bReadTableFromFile = FALSE; //flag para verificar se tabela lida de
                                      //um arquivo
 dlgLut.m_nDataType    = m_nDataType;
 dlgLut.m_nFrequency   = m_nSpatialFreq;
 dlgLut.m_nVisc        = m_nVisc;
 dlgLut.m_nNumEntry    = m_nEntry;
 dlgLut.m_nTask        = m_nTask;
 dlgLut.m_nRangeMax    = m_nRangeMax;
 dlgLut.m_nRangeMin    = m_nRangeMin;
 switch (m_nTask)
{

 case ISOMORPHIC: //Isomorfismo
        dlgLut.m_nColorTable = m_nColorTable;
        dlgLut.m_nOpacityMax = m_nOpacMax;
        dlgLut.m_nOpacityMin = m_nOpacMin;
        //Verifica se a janela da tabela já esta criada
        //Caso contrario, destroe e constroe novamente
        if (!dlgLut.m_bFlagCreated) dlgLut.Create();
        else
        {
               dlgLut.CloseWindow();
               dlgLut.Create();
        }
        break;

 case SEGMENTATION: //Segmentacao
        dlgLut.m_nNumSegment = atoi(m_strNumSegment);
        dlgLut.m_nInitSeg1    = m_nIniSeg1;
        dlgLut.m_nOpacSeg1    = m_nOpacSeg1;
        dlgLut.m_nColorSeg1   = m_nColorSeg1;
        dlgLut.m_nInitSeg2    = m_nIniSeg2;
        dlgLut.m_nOpacSeg2    = m_nOpacSeg2;
        dlgLut.m_nColorSeg2   = m_nColorSeg2;
        //Verifica o numero de segmentos especificado e transfere
        //os valores de acordo com esse numero
        if ( dlgLut.m_nNumSegment > 2 )
        (
               dlgLut.m_nInitSeg3   = m_nIniSeg3;
               dlgLut.m_nOpacSeg3   = m_nOpacSeg3;
               dlgLut.m_nColorSeg3 = m_nColorSeg3;
               if ( dlgLut.m_nNumSegment > 3 )
               (
                      dlgLut.m_nInitSeg4   = m_nIniSeg4;
                      dlgLut.m_nOpacSeg4   = m_nOpacSeg4;
                      dlgLut.m_nColorSeg4 = m_nColorSeg4;
                      if ( dlgLut.m_nNumSegment > 4 )
                      (
                             dlgLut.m_nInitSeg5   = m_nIniSeg5;
                             dlgLut.m_nOpacSeg5   = m_nOpacSeg5;
                             dlgLut.m_nColorSeg5 = m_nColorSeg5;
                             if ( dlgLut.m_nNumSegment > 5 )
                             {
                                    dlgLut.m_nInitSeg6   = m_nIniSeg6;
                                    dlgLut.m_nOpacSeg6   = m_nOpacSeg6;
                                    dlgLut.m_nColorSeg6 = m_nColorSeg6;
                                    if ( dlgLut.m_nNumSegment > 6 )
                                    {
                                           dlgLut.m_nInitSeg7   = m_nIniSeg7;
                                           dlgLut.m_nOpacSeg7   = m_nOpacSeg7;
                                           dlgLut.m_nColorSeg7 = m_nColorSeg7;
                                    } //6
                             } //5
```

```
                        } //4
                } //3
        } //2
        //Verifica se a janela da tabela já esta criada
        //Caso contrario, destroe e constroe novamente
        if (!dlgLut.m_bFlagCreated) dlgLut.Create();
        else
        {
                dlgLut.CloseWindow();
                dlgLut.Create();
        }
        break;

case HIGHLIGHTING: //Enfatizacao
        dlgLut.m_nBeginRange = m_nBeginRange;
        dlgLut.m_nEndRange   = m_nEndRange;
        dlgLut.m_nRangeOpacity  = m_nRangeOpac;
        dlgLut.m_nRestTableOpac = m_nRestOpac;
        dlgLut.m_nColorTable    = m_nColorTable;
        //Verifica se a janela da tabela já esta criada
        //Caso contrario, destroe e constroe novamente
        if (!dlgLut.m_bFlagCreated) dlgLut.Create();
        else
        {
                dlgLut.CloseWindow();
                dlgLut.Create();
        }
        break;

case RAINBOW: //Tabela Default
        if (!m_flagOpacCancelClicked)
        {
                dlgLut.m_nOpacityMin = m_nOpacMin;
                dlgLut.m_nOpacityMax = m_nOpacMax;
                //Verifica se a janela da tabela já esta criada
                //Caso contrario, destroe e constroe novamente
                if (!dlgLut.m_bFlagCreated) dlgLut.Create();
                else
                {
                        dlgLut.CloseWindow();
                        dlgLut.Create();
                }
        }
                break;

case GRAYSCALE: //Tabela em escala de cinzas
        if (!m_flagOpacCancelClicked)
        {
                dlgLut.m_nOpacityMin = m_nOpacMin;
                dlgLut.m_nOpacityMax = m_nOpacMax;
                //Verifica se a janela da tabela já esta criada
                //Caso contrario, destroe e constroe novamente
                if (!dlgLut.m_bFlagCreated) dlgLut.Create();
                else
                {
                        dlgLut.CloseWindow();
                        dlgLut.Create();
                }
        }
        break;
} //switch

//Habilita o botao que permite o armazenamento da tabela em arquivo
 CWnd* p;
 p = GetDlgItem( IDC_BUTTON3 ); p->EnableWindow();
}
```