UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

INTERFACE 3D PARA REPRESENTAÇÃO DA DISTRIBUIÇÃO DE CORES DE IMAGENS DIGITAIS EM DIFERENTES ESPAÇOS DE CORES

TRABALHO DE GRADUAÇÃO

Leonardo Crauss Daronco

Santa Maria, RS, Brasil

2007

INTERFACE 3D PARA REPRESENTAÇÃO DA DISTRIBUIÇÃO DE CORES DE IMAGENS DIGITAIS EM DIFERENTES ESPAÇOS DE CORES

por

Leonardo Crauss Daronco

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de

Bacharel em Ciência da Computação

Orientador: Prof. Dr. Marcos Cordeiro d'Ornellas

Trabalho de Graduação Nº 230

Santa Maria, RS, Brasil

2007

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

INTERFACE 3D PARA REPRESENTAÇÃO DA DISTRIBUIÇÃO DE CORES DE IMAGENS DIGITAIS EM DIFERENTES ESPAÇOS DE CORES

elaborado por
Leonardo Crauss Daronco

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:
Prof. Dr. Marcos Cordeiro d'Ornellas
(Presidente/Orientador)

Prof. Dr. Cesar Tadeu Pozzer (UFSM)

Prof. Dr. Felipe Martins Müller (UFSM)

Santa Maria, 05 de março de 2007.

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

INTERFACE 3D PARA REPRESENTAÇÃO DA DISTRIBUIÇÃO DE CORES DE IMAGENS DIGITAIS EM DIFERENTES ESPAÇOS DE CORES
Autor: Leonardo Crauss Daronco
Orientador: Prof. Dr. Marcos Cordeiro d'Ornellas
Local e data da defesa: Santa Maria, 05 de março de 2007.

Este trabalho apresenta o processo de pesquisa e desenvolvimento de uma ferramenta computacional com o objetivo de disponibilizar métodos de conversão entre espaços de cores de imagens digitais e oferecer ao usuário uma interface tridimensional para visualização da distribuição de cores dessas imagens. A pesquisa teve como foco principal o estudo das características de alguns dos espaços de cores mais utilizados, principalmente de suas equações de conversão. O desenvolvimento da ferramenta foi feito modularmente, separando o processo de conversão do processo de exibição tridimensional, ambos utilizando a linguagem de programação `Java` e o último utilizando a biblioteca gráfica `OpenGL` (`JOGL`) para criação do ambiente tridimensional interativo. Além disso, a ferramenta desenvolvida faz parte de um projeto em desenvolvimento no grupo de processamento de informação multimídia PIGS (UFSM), o que resultou em sua inclusão no sistema de processamento e análise de imagens `Arthemis`.

Palavras-chave: espaços de cores; interface tridimensional; análise de cores.

ABSTRACT

Trabalho de Graduação
Graduation in Computer Science
Universidade Federal de Santa Maria

3D INTERFACE FOR REPRESENTATION OF THE COLOR DISTRIBUTION IN DIGITAL IMAGES USING DIFFERENT COLOR SPACES
Author: Leonardo Crauss Daronco
Advisor: Prof. Dr. Marcos Cordeiro d'Ornellas

This paper describes the research and development of a computational tool to provide conversion methods between color spaces in digital images and offer to the user a 3D interface for image color distribution visualization. The research had its focus on the study about some of the most used color spaces and their features, specially about their conversion equations. The tool was developed modularly, detaching the conversion process from the 3D exhibition process, both using the `Java` programming language and the latter using the `OpenGL` graphics library (`JOGL`) to create the 3D interactive environment. Furthermore, this tool is a part of a project in development by the multimedia information processing group PIGS (from UFSM), which resulted in its inclusion in the image processing and analysis system `Arthemis`.

Keywords: color spaces; 3D interface; color analysis.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
BMP	Formato de imagens: Bitmap
CIE	Commission International de L'Eclairage
DICOM	Formato de imagens: Digital Imaging and Communications in Medicine
GTK	GIMP Toolkit para criação de interfaces gráficas
ICC	International Color Consortium
ImageIO	Java Image Input/Output API
ITU-R	International Telecommunication Union Radiocommunication Sector
JAI	Java Advanced Imaging
JOGL	Java OpenGL
JPEG	Formato de imagens: Joint Photographic Experts Group
JSR	Java Specification Requests
JVM	Java Virtual Machine
MPEG	Padrão para codificação de áudio e vídeo: Moving Picture Experts Group
PIGS	Grupo de processamento de informação multimídia da Universidade Federal de Santa Maria
SGI	Silicon Graphics, Inc.
TIFF	Formato de imagens: Tagged Image File Format
UML	Unified Modeling Language

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos têm-se notado uma crescente utilização das imagens digitais nas mais diversas áreas do conhecimento, onde os espaços de cores existem justamente para tornar possível a exibição de imagens reais no meio digital. Unindo conceitos de espaços de cores com visualização tridimensional, é possível criar maneiras alternativas para análise de imagens (análise de suas cores). Desta união foi definido o propósito principal deste trabalho: desenvolver uma ferramenta para disponibilizar métodos de conversão entre alguns dos diversos espaços de cores disponíveis e implementar uma interface tridimensional para exibição da distribuição de cores das imagens nestes espaços de cores.

A distribuição de cores pode ser representada de diversas maneiras em um gráfico 3D. Uma delas (a utilizada neste trabalho) é representar cada cor existente na imagem por apenas um objeto no gráfico (um ponto, por exemplo), posicionado no ambiente de acordo com os valores de suas coordenadas, o que formará a chamada nuvem de pontos do gráfico. Outra maneira seria representar as cores através de um histograma, onde cada cor é representada por um objeto de tamanho correspondente à freqüência desta cor na imagem, e não por um simples objeto de tamanho fixo como no caso anterior.

Visualizar a imagem em duas dimensões, como costumamos ver, é simplesmente uma das maneiras com que as informações desta imagem podem ser interpretadas. Em muitos casos, a existência de maneiras de visualização alternativas pode facilitar os processos de análise e processamento destas imagens. Exemplificando, pode ser citado o processo de agrupamento (um processo para auxiliar a tarefa de segmentação), um processo que, sendo realizado no espaço de cores da imagem através de uma representação em três dimensões deste, pode resultar na segmentação da imagem original. Portanto a visualização da imagem sob diferentes aspectos, incluindo a visualização da distribuição de cores em um ambiente 3D, pode ser bastante útil para a análise, pré-processamento e processa-

mento das imagens.

Para melhorar a visualização tridimensional, também podem ser incluídos alguns artifícios no ambiente, como as projeções. A projeção da nuvem de pontos em um dos eixos de coordenadas pode facilitar a análise da distribuição de cores na imagem e também a realização de operações sobre esta nuvem de pontos (operações que futuramente podem ser implementadas, como a seleção de regiões no gráfico). Exemplificando, podemos ter uma imagem cuja projeção da nuvem de pontos sobre o plano gerado pelo 1º e 2º eixos gera uma linha sobre este plano. Uma interpretação que pode ser feita deste caso é que a variação das cores está centralizada na coordenada representada pelo 3º eixo. E com a exibição de grades nos planos, é possível saber com mais precisão quais os valores das coordenadas dos pontos que formam a nuvem de pontos.

A visualização tridimensional das cores da imagem está diretamente relacionada ao espaço de cores que será utilizado, uma vez que este definirá como serão distribuídos os objetos no gráfico.

Um modelo de cor pode ser definido como um sistema de coordenadas tridimensionais e um subespaço dentro deste sistema, onde cada cor é representada por um único ponto contido dentro deste subespaço (GONZALEZ; WOODS, 2002). Modelos de cores são utilizados para facilitar e tornar possível a representação de cores no meio digital de várias formas (através da existência de vários modelos) de acordo com as necessidades dos dispositivos e usuários.

Atualmente existem diversos modelos de cores, cada um possuindo seu próprio sistema de coordenadas, ou seja, sua forma própria de representação das cores. Esses modelos podem ser divididos em dois grandes grupos, de acordo com sua aplicabilidade. O primeiro grupo é composto por modelos que foram definidos para serem utilizados principalmente em dispositivos de *hardware*, enquanto o segundo grupo é formado pelos modelos criados para uso em *softwares* que visam a manipulação de cores.

No caso onde a aplicação é voltada para dispositivos de *hardware*, os modelos utilizados são categorizados como dependentes de dispositivo. Hoje em dia, temos, por exemplo, o modelo RGB (*red*, *green*, *blue*) que é amplamente utilizado em televisores e monitores de computadores (tubos de raios catódicos, telas de cristal líquido e telas de plasma) e o modelo CMYK (*cyan*, *magenta*, *yellow*, *key*) muito utilizado em impressoras. A definição do modelo ou a escolha de um modelo já criado é feita de acordo com as

propriedades técnicas dos dispositivos nos quais eles serão utilizados.

No segundo grupo estão os modelos que auxiliam nas tarefas de análise e processamento de imagens, normalmente por serem relacionados à percepção humana das cores (RUSS, 1999). Entre estes podem ser citados os modelos definidos pela organização francesa CIE na primeira metade do século 20. O CIE parte do princípio de que o olho humano possui 3 tipos diferentes de estruturas receptoras de cores (chamadas `cones`) e estas estruturas respondem de uma maneira diferente à luz visível de acordo com o comprimento de onda desta luz. Apesar de alguns dos outros espaços existentes utilizarem este mesmo princípio, os modelos definidos pelo CIE também buscam representar todas as cores visíveis ao olho humano, o que não acontece com a maioria dos outros modelos. O espaço de cores CIE XYZ (também conhecido como *CIE 1931 color space*) é um dos espaços definidos seguindo estes princípios e hoje serve de base para a definição de diversos outros espaços de cores. Diversas definições sobre os espaços de cores do CIE e artigos relacionados podem ser encontrados em (HOFFMAN, 2005a).

Esta variedade de espaços existentes é um ponto importante no trabalho, pois disponibiliza diversas formas de visualizar a imagem no espaço 3D. Na literatura é possível encontrar diversas fórmulas matemáticas (normalmente tendo como base o modelo RGB) que já foram definidas para realizar as conversões entre alguns espaços de cores. Estas fórmulas foram utilizadas pela ferramenta desenvolvida por este trabalho e serão apresentadas nas próximas seções.

Além da pesquisa sobre espaços de cores e interfaces tridimensionais, uma etapa fundamental do trabalho é a etapa de desenvolvimento da ferramenta. Nesta etapa, o primeiro passo realizado foi a modelagem da ferramenta, onde ela foi dividida em dois módulos principais: o módulo das conversões e o módulo da interface.

O módulo da interface é formado pelos componentes do ambiente tridimensional e pelas janelas que exibem as opções de configuração da ferramenta ao usuário e que permitem que ele interaja com o sistema. O ambiente tridimensional possibilita que o usuário verifique a distribuição espacial das cores de uma imagem de uma maneira fácil e interativa, podendo configurar parâmetros, modificar cores, realizar rotações, entre outras diversas tarefas existentes com o objetivo de facilitar a visualização do gráfico.

Já o módulo de conversões é formado por todos aqueles componentes responsáveis pelo suporte aos espaços de cores, onde estão implementadas as fórmulas de conversão

entre esses espaços. O desenvolvimento deste módulo seguiu premissas de que a inclusão de suporte a novos modelos de cores deveria ser fácil e rápida, portanto os conversores são considerados componentes e podem ser adicionados e removidos da ferramenta com poucas (na maioria dos casos, nenhuma) alterações em outros componentes.

Outro aspecto importante do desenvolvimento é a inclusão da ferramenta desenvolvida no sistema de processamento e análise de imagens do grupo PIGS (UFSM, Santa Maria) chamado `Arthemis`[1]. Este sistema é desenvolvido utilizando a linguagem Java e bibliotecas disponíveis para a mesma, como a `JAI` e a `ImageIO`. Com a ferramenta fazendo parte desse sistema, o usuário tem a vantagem de não utilizá-la isoladamente, podendo usufruir de todas as funcionalidades do `Arthemis` para melhorar a visualização das imagens com a ferramenta. A integração também auxilia o desenvolvimento da ferramenta no momento em que ela pode fazer uso de diversas operações já implementadas no `Arthemis` (como funções para carregar e salvar imagens, por exemplo) e também contribui para tornar esse sistema mais completo, já que agora possui uma ferramenta para visualização tridimensional de imagens.

Como o desenvolvimento foi feito de forma modular, a ferramenta não é totalmente dependente do `Arthemis`. Temos o módulo das conversões, que é independente (podendo ser utilizado por qualquer aplicação, como uma biblioteca), enquanto o módulo da interface possui uma parte independente (os componentes de criação do ambiente 3D) e uma dependente do sistema (as janelas para integração da ferramenta no `Arthemis`).

Como objetivos específicos do desenvolvimento, podemos citar:

- Implementar métodos de conversão para o maior número possível de modelos de cores: são suportados dez diferentes espaços de cores, alguns com diversas opções de configuração;

- Suportar o maior número de formatos de imagens, dando prioridade aos mais utilizados como TIFF, BMP, DICOM, JPEG, entre outros: são suportados todos os formatos que o sistema `Arthemis` dá suporte;

- A interface tridimensional deve, além de permitir que o usuário visualize a distribuição das cores, ser configurável (opções de cores e exibição de projeções, por

[1]Mais informações podem ser encontradas no *website* do grupo em: http://w3.ufsm.br/pigs/

exemplo), interativa (ser possível realizar rotações e translações, por exemplo) e de fácil uso (comandos simples para realizar as tarefas citadas);

- Projetar o sistema de modo que exista uma maneira fácil e padronizada para inclusão de novos conversores para das suporte à novos espaços de cores;
- Manter um bom desempenho nas conversões e na interação com a interface 3D.

As próximas seções deste trabalho apresentarão a revisão da literatura, ressaltando conceitos e propriedades dos espaços de cores e interfaces 3D, o processo de desenvolvimento da ferramenta e, por fim, uma avaliação do trabalho desenvolvido e algumas sugestões de trabalhos futuros.

2 MODELOS DE CORES E SUA REPRESENTAÇÃO TRIDIMENSIONAL

Este capítulo tem como objetivo apresentar informações sobre conceitos e definições relacionadas à modelos de cores e à representação de cores em imagens digitais, citando aqueles modelos que serão utilizados por este trabalho e apresentando suas propriedades.

Após as considerações relacionadas aos modelos de cores, serão apresentadas as alternativas que foram consideradas para implementação da interface tridimensional, ressaltando seus pontos positivos e negativos. Por fim, serão discutidas algumas aplicações semelhantes a este trabalho que já foram desenvolvidas.

2.1 Representação de cores em imagens digitais

A utilização e representação de cores em imagens digitais é amplamente discutida em diversas referências, onde podem ser encontradas informações sobre o sistema humano de percepção de cores e o uso de conceitos relacionados para a exibição de cores em imagens digitais (ver (GONZALEZ; WOODS, 2002) e (PRATT, 2001)). Discutir todo este processo foge do escopo deste trabalho, que apresenta apenas algumas definições necessárias para conceituação do uso de modelo de cores.

Segundo (GONZALEZ; WOODS, 2002), o uso de cores no processamento de imagens ocorre em função de dois fatores: (1) cores são poderosas para identificar e distinguir objetos, e (2) o sistema visual humano pode distinguir muito mais facilmente entre as muitas cores existentes do que entre os poucos níveis de cinza existentes em imagens monocromáticas. Este segundo fator relaciona-se com o objetivo da interface 3D deste trabalho, que utiliza as cores para facilitar a análise humana das imagens.

A forma de representação de cores em imagens digitais parte das propriedades ex-

perimentais relacionadas ao sistema visual humano. Sabe-se que o olho humano possui milhões de estruturas chamadas `cones`, que são responsáveis pela identificação das cores. Essas estruturas podem ser divididas em três grandes grupos, onde cada grupo é responsável por identificar uma das três cores consideradas as cores primárias da luz: o vermelho, o verde e o azul.

Observando-se o espectro eletromagnético, temos que a luz visível ao olho humano varia do violeta (380 nm) ao vermelho (780 nm). Em 1931, a CIE definiu o comprimento de onda padrão para as três cores primárias: azul = 435.8 nm, verde = 546.1 nm e vermelho = 700 nm. Estes seriam os comprimentos de onda detectados por cada um dos grupos de cones do olho humano que foram citados anteriormente.

A CIE também definiu um diagrama de cromaticidade (figura 2.1), que mostra a composição das cores visíveis ao olho humano definindo a pureza e o comprimento de onda dominante de cada cor. O diagrama possui duas dimensões e é formado pelas coordenadas `x` e `y`, que definem as cores que podem ser representadas (RUSS, 1999). Existe também uma terceira coordenada `Y`, chamada de luminância (do inglês, *luminance*) que não é visível no diagrama.

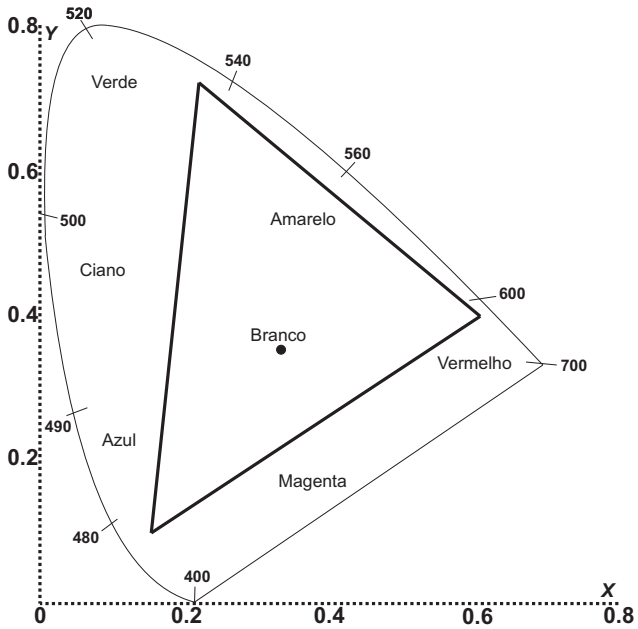

Figura 2.1: Diagrama de cromaticidade CIE exibindo o gamut do modelo RGB.

Criada com base em uma imagem obtida de (RUSS, 1999)

Para entender o conceito de luminância é necessário primeiro entender o conceito de brilho. De acordo com o CIE, o brilho de um objeto pode ser definido como o atributo da

sensação visual que corresponde à quantidade de luz emitida por este objeto (POYNTON, 2006), o que pode ser interpretado como a quantidade de luz que é refletida de um objeto. Como este conceito é difícil de ser representado matematicamente, foi definido o conceito de luminância, que corresponde à quantidade de luz irradiada de acordo com os padrões de percepção de um observador padrão (o *Standard Observer*, definido pela CIE). Uma definição mais aprofundada de luminância pode ser encontrada em (POYNTON, 2006).

O diagrama do CIE pode ser utilizado, entre outros propósitos, para identificar as variações de cores possíveis entre duas cores definidas e para a visualização do *gamut* dos espaços de cores. Para visualizar todas as variações entre duas cores, basta identificar os dois pontos onde essas cores estão no diagrama e traçar uma reta ligando esses dois pontos. As variações possíveis são todas as cores que se encontram sob a reta criada.

Gamut refere-se ao espaço que contém todas as cores que podem ser representadas por determinado espaço de cores, ou seja, o conjunto de todas as cores que podem ser geradas através da combinação das coordenadas utilizadas pelo espaço de cores em questão. Para visualização do *gamut* de um modelo, é normalmente utilizado o diagrama de cromaticidade CIE, onde basta que sejam traçadas linhas conectando as cores primárias do modelo. O *gamut* deste modelo será representado pela forma geométrica criada pelas linhas traçadas e as cores que podem ser representadas por ele serão todas aquelas localizadas na parte interna da forma geométrica.

Na figura 2.1, por exemplo, pode ser visualizado o *gamut* do modelo RGB (triângulo interno da figura) através do diagrama de cromaticidade da CIE, onde o conjunto das cores sob a área interna do triângulo representa o conjunto de todas as cores que podem ser representadas pelo RGB. Através da figura percebe-se que não são todas as cores visíveis que podem ser representadas pelo modelo RGB, apesar dele fazer o uso da combinação entre as cores vermelho, verde e azul, assim como está definido o sistema visual humano.

2.2 Modelos de cores

Como já citado na seção 1 deste trabalho, modelos de cores são utilizados para facilitar a especificação das cores de uma maneira padronizada e para a representação destas cores no meio digital. Antes da apresentação dos modelos utilizados neste trabalho (seção 2.3), é necessário definir alguns conceitos importantes, que serão discutidos no restante desta seção.

Os modelos de cores podem ser orientados à *hardware* ou não. Modelos orientados à *hardware* são assim chamados por terem sido criados devido às necessidades e propriedades dos dispositivos nos quais são utilizados. Neste grupo podem ser citados os dois modelos mais conhecidos: RGB, utilizado em dispositivos como monitores, e CMYK, utilizado em impressoras. Ambos os modelos já foram mencionados neste documento e mais detalhes serão vistos na seção 2.3. O outro grupo, onde os modelos não são orientados à *hardware*, é composto por modelos desenvolvidos para facilitar a análise de imagens, tanto análises humanas quanto análises computadorizadas. Alguns desses modelos foram criados com base na forma com que o sistema visual humano diferencia as cores, utilizando conceitos como a existência dos cones no olho humano e os variados comprimentos de onda que são percebidos. Neste segundo grupo podem ser citados modelos como o modelo HSI e o espaço CIE L*a*b*, que também serão apresentados em mais detalhes na seção 2.3.

Outro conceito importante é quanto ao espaço ser absoluto ou não. Um espaço de cores absoluto é um espaço que não é ambíguo, ou seja, que possui uma única maneira de representar cada uma das cores que fazem parte do seu *gamut*. Estes espaços possuem diversas especificações de como as cores devem ser visualizadas, que garantem que ele será absoluto e o tornam independente de fatores externos.

Um espaço não absoluto é exatamente o contrário: não possui uma única forma de representar suas cores e é dependente de fatores externos. Novamente, temos como exemplo o modelo RGB. Se visualizarmos uma mesma imagem em dois monitores, as cores que são percebidas podem ser diferentes devido às diferenças de configuração dos monitores (pois o modelo não especifica como eles devem estar configurados), apesar de internamente as imagens possuírem os mesmos valores de suas coordenadas.

Conversões entre espaços não absolutos e entre um espaço não absoluto e um absoluto podem ser realizadas, porém normalmente não possuem muita precisão (as cores podem sofrer alterações após a conversão). Isso torna a conversão inválida e praticamente sem sentido em casos onde são necessários resultados muito precisos. Porém, para análise visual, essas conversões podem ser úteis, por mais que não sejam absolutamente precisas. Espaços não absolutos podem ser transformados em absolutos adicionando-se a eles definições mais específicas, o que pode ser feito com o uso de `ICC profiles`, por exemplo.

`ICC Profiles` são perfis criados pelo ICC para definir mais precisamente as propriedades de um espaço de cores. Isso é feito através de um mapeamento entre as cores do espaço em questão e um espaço chamado *profile connection space* (PCS), que corresponde ao espaço CIE L*a*b* ou ao CIE XYZ. Além de tornarem um espaço não absoluto em absoluto, eles também são utilizados para conversões entre espaços e para definição das cores em dispositivos que capturam ou exibem cores (onde são consideradas três categorias de dispositivos: *input devices*, *display devices* e *output devices*). Como cada dispositivo possui seu próprio perfil, muitos fabricantes já o disponibilizam junto com seus dispositivos, enquanto outros possibilitam que o usuário crie este perfil utilizando ferramentas específicas para tal.

Uma aplicação prática dos perfis acontece na impressão de documentos coloridos, onde o usuário visualiza a imagem em um monitor, por exemplo, e então cria as cópias impressas. Nestes casos são utilizados os perfis dos dois dispositivos (monitor e impressora) para realização das conversões de cores necessárias. Assim, garante-se que as cores vistas no monitor são as mesmas que serão exibidas nos papéis impressos, processo que é feito por diversos programas de edição de imagens atualmente. Mais informações sobre eles podem ser encontradas em (ICC, 2007).

Além dos conceitos já citados também é importante explicar a diferença entre modelos de cores e espaços de cores. Um modelo de cores é um modelo matemático que especifica como as cores podem ser representadas utilizando conjuntos de números (tuplas), normalmente com 3 ou 4 valores. No momento em que temos uma maneira de conectar este modelo a um espaço de cores absoluto, através de uma função de mapeamento que irá delimitar o *gamut* desse modelo de cores no espaço utilizado, podemos fazer deste modelo um espaço de cores. Ou seja, quando o modelo possui especificações mais precisas de como os seus valores devem ser interpretados (como, por exemplo, com o uso de `ICC profiles`), ele se torna um espaço de cores.

Temos, por exemplo, o modelo de cores RGB e os espaços de cores sRGB e Adobe RGB, ambos criados com base no modelo RGB. Estes espaços possuem diferenças relacionadas às suas definições, como as coordenadas de cromaticidade definidas para os valores de `R`, `G` e `B` e o ponto de luz branca utilizado, por exemplo.

A figura 2.2 apresenta um exemplo de uma mesma imagem sendo exibida em quatro diferentes espaços de cores. A primeira imagem, representada em RGB, mostra as

cores originais da imagem, as cores assim como elas são percebidas pelo sistema visual humano, enquanto as outras representam esta imagem convertida para os espaços CMY, YIQ e HSI, fazendo o uso de pseudo-cores (ou cores falsas).

Figura 2.2: Uma mesma imagem em quatro modelos diferentes: a) RGB b) CMY c) YIQ d) HSI.

Nos exemplos da figura 2.2, a conversão da imagem original para três espaços de cores diferentes originou três imagens com coordenadas diferentes. Os valores das coordenadas dessas imagens são interpretados e exibidos como se estivessem no modelo RGB, o que possibilita a visualização das diferenças existentes entre estes valores, e, portanto, entre os espaços de cores utilizados.

Visualizando essas diferenças é possível perceber que diferentes espaços de cores destacam diferentes partes da imagem, diferentes objetos. No espaço HSI, por exemplo, nota-se que a plumagem existente no chapéu e o cabelo da modelo estão em destaque por apresentarem um grande contraste em relação ao resto da imagem. Isso mostra que

existe uma grande variação de cores nesta imagem, maior do que a variação que existe na imagem exibida no modelo RGB. Esta variação que pode ser utilizada como um artifício para facilitar tarefas de processamento de imagens, como a segmentação.

2.3 Modelos utilizados

Nesta seção serão apresentados os modelos de cores utilizados neste trabalho até o presente momento. Serão discutidas suas propriedades, uso e equações de conversão. As equações de conversão estarão sempre relacionadas à conversão do modelo RGB para o modelo em questão, enquanto que as equações para realizar o processo contrário não serão apresentadas. As últimas citadas não são necessárias para o trabalho pois assumimos que todas as imagens carregadas na aplicação estão armazenadas no modelo RGB, por ele ser o modelo que normalmente é adotado para armazenamento e exibição de imagens.

2.3.1 RGB

O modelo RGB é composto de três coordenadas, `R` (*red*), `G` (*green*) e `B` (*blue*), que especificam o brilho dos sinais vermelho, verde e azul (as cores primárias da luz), respectivamente, cada um representado por um comprimento de onda definido (RUSS, 1999). As cores são então geradas pela adição dos três componentes.

O RGB é um modelo muito utilizado atualmente, principalmente por dispositivos com capacidade de emissão de luz, como monitores, televisores, câmeras de vídeo, entre outros, tanto para captura quando para exibição de imagens. Estes dispositivos utilizam o modelo RGB pela facilidade de criação das cores que este possibilita. Monitores de raios catódicos, por exemplo, possuem emissores para os três canais `R`, `G` e `B`, que se combinam na tela formando a cor desejada. Em função da maioria dos dispositivos de visualização atuais fazerem uso do modelo RGB, ele também é muito utilizado para o armazenamento de imagens digitais, o que facilita a tarefa de exibição das imagens.

O RGB é um modelo não absoluto e orientado à *hardware*. A exibição de uma mesma imagem em dois dispositivos pode resultar em cores aparentemente diferentes. Porém, muitos espaços de cores absolutos foram definidos baseando-se no modelo RGB. Quando é necessária a conversão do modelo RGB para espaços absolutos como o CIE L*a*b*, por exemplo, é necessário que primeiro seja efetuada uma conversão do modelo RGB para algum espaço absoluto baseado neste modelo. Neste trabalho, o objetivo do uso de

diversos espaços é possibilitar uma variedade de escolha para o usuário, permitindo que ele escolha aquela que mais se adapta ao seu problema. Diversas informações sobre estes espaços de cor podem ser encontradas em (LINDBLOOM, 2007).

As equações utilizadas para a conversão do modelo RGB para os espaços absolutos citados acima (a lista dos espaços utilizados pelo trabalho encontra-se na seção 3.1.2) seguem a seguir, onde (r, g, b) são os valores RGB convertidos, (R, G, B) são os valores RGB originais e γ representa o *gamma* do espaço de cor escolhido. Elas podem ser encontradas em (LINDBLOOM, 2007).

Para o espaço sRGB:

$$r = \begin{cases} R/12.92 & \text{se } R \leq 0.04045 \\ ((R + 0.055)/1.055)^{2.4} & \text{se } R > 0.04045 \end{cases} \tag{2.1a}$$

$$g = \begin{cases} G/12.92 & \text{se } G \leq 0.04045 \\ ((G + 0.055)/1.055)^{2.4} & \text{se } G > 0.04045 \end{cases} \tag{2.1b}$$

$$b = \begin{cases} B/12.92 & \text{se } B \leq 0.04045 \\ ((B + 0.055)/1.055)^{2.4} & \text{se } B > 0.04045 \end{cases} \tag{2.1c}$$

Para os outros espaços:

$$r = R^{\gamma} \tag{2.2a}$$

$$g = G^{\gamma} \tag{2.2b}$$

$$b = B^{\gamma} \tag{2.2c}$$

Os valores RGB são normalmente representados utilizando-se 24 bits, 8 bits para cada componente. Isso faz com que o valor de cada componente possa variar no intervalo [0,255], valores que normalmente são normalizados para ficarem no intervalo [0,1] antes dos cálculos.

2.3.2 CMY e CMYK

O modelo CMY é um modelo subtrativo, onde os valores de suas coordenadas são subtraídos (e não adicionados como no caso do RGB) para formar as cores desejadas. Assim como o modelo RGB, ele é um modelo não absoluto e orientado a *hardware*. Seus componentes são as cores primárias dos pigmentos: `C` (ciano, do inglês *cyan*), `M`

(magenta, assim como no inglês) e `Y` (amarelo, do inglês *yellow*). Estas três cores são também chamadas de cores secundárias da luz, pois caso iluminarmos um objeto da cor magenta, por exemplo, ele absorverá a cor verde, emitindo apenas raios vermelhos e azuis que formarão a cor magenta. Portanto, o magenta é a cor complementar do verde, assim como ocorre entre o ciano e o vermelho e entre o amarelo e o azul.

A equação utilizada para conversão do modelo RGB para CMY é bastante simples (GONZALEZ; WOODS, 2002), (PRATT, 2001):

$$C = 1.0 - R \tag{2.3a}$$

$$M = 1.0 - G \tag{2.3b}$$

$$Y = 1.0 - B \tag{2.3c}$$

Os valores dos componentes do CMY variam conforme os valores RGB. Assumindo que os valores RGB estejam normalizados, os valores CMY também estarão no intervalo [0,1].

O modelo CMY é muito utilizado em impressoras, onde é interessante assumir que a ausência de cores forma a cor branca (o que acontece no CMY), normalmente a cor dos papéis utilizados para impressão. Porém, para formar o preto, é necessário combinar certa quantidade de cada um dos componentes, o que normalmente provoca o uso excessivo de cada uma das tintas.

Em função disso foi criado o modelo CMYK, que adiciona o quarto componente `K` ao modelo CMY. Este último componente é chamado de *key* e corresponde à cor preta (ou à tinta preta nas impressoras). O valor deste componente é normalmente o menor valor da tupla CMY, após a conversão utilizando as equações (2.3). Depois de encontrado o valor de `K`, ele é subtraído de cada um dos componentes CMY, gerando os valores do modelo CMYK.

Com isso é possível incluir a tinta preta nas impressões, o que resulta na redução do uso das tintas coloridas que normalmente são mais caras. E, nos casos de impressões em preto e branco, pode ser utilizada apenas a tinta preta, descartando o uso de tintas coloridas.

2.3.3 YIQ e YUV

Os modelos YIQ e YUV são utilizados para transmitir imagens de televisão analógica. O componente `Y` de ambos os modelos possui o mesmo valor e corresponde à luminância da imagem, o valor do brilho monocromático desta imagem de acordo com a sensibilidade do olho humano a cada um dos componentes RGB. `IQ` e `UV` estão associados à cromaticidade, ou seja, às cores da imagem. Esta separação permite que o mesmo sinal possa ser transmitido para televisores coloridos e monocromáticos, onde, nos primeiros, os três canais serão utilizados e, nos outros, apenas o canal `Y` pode formar a imagem desejada.

As equações utilizadas para conversão de RGB para YIQ e para YUV são (RUSS, 1999), (COLANTONI, 2004), (PRATT, 2001), (SHAPIRO; STOCKMAN, 2001):

$$Y = 0.29889531 * R + 0.58662247 * G + 0.11448223 * B \tag{2.4a}$$

$$I = 0.59597799 * R - 0.27417610 * G - 0.32180189 * B \tag{2.4b}$$

$$Q = 0.21147017 * R - 0.52261711 * G + 0.31114694 * B \tag{2.4c}$$

$$Y = 0.30 * R + 0.59 * G + 0.11 * B \tag{2.5a}$$

$$U = 0.493 * (B - Y) \tag{2.5b}$$

$$V = 0.877 * (R - Y) \tag{2.5c}$$

Os valores de `Y` variam no intervalo [0,1] enquanto os outros componentes podem ser positivos ou negativos.

2.3.4 YCbCr

O YCbCr, assim como os modelos YIQ e YUV citados anteriormente, é utilizado em sistemas de vídeo, sendo muitas vezes confundido com o próprio YUV (WIKIPEDIA, 2006). O componente `Y` é chamado *luma*, que corresponde ao brilho da imagem e é normalmente utilizado com sinais de vídeo. Os componentes `Cb` e `Cr` correspondem aos componentes cromáticos do azul e do vermelho, respectivamente.

Sendo utilizado em sistemas de vídeo, a conversão para o espaço YCbCr é dependente do padrão de codificação de vídeo que será utilizado. No trabalho é utilizado o padrão `ITU-R BT.601`, um antigo padrão que já foi reutilizado em alguns formatos de vídeo

novos como o MPEG, através do qual são definidas as seguintes equações de conversão (PRATT, 2001), (COLANTONI, 2004):

$$Y = 0.29900 * R + 0.58700 * G + 0.11400 * B \tag{2.6a}$$

$$Cb = -0.16874 * R - 0.33126 * G + 0.5 * B \tag{2.6b}$$

$$Cr = 0.5 * R - 0.41869 * G - 0.08131 * B \tag{2.6c}$$

2.3.5 HSI

O modelo HSI, ao contrário dos modelos citados até o momento, que eram orientados a *hardware*, foi definido com base na maneira com que o homem descreve as cores. O homem não descreve a cor de um objeto através da quantidade de cada cor primária que ela possui (como no modelo RGB), mas sim através de seu matiz, saturação e brilho (GONZALEZ; WOODS, 2002).

`H` (matiz, do inglês *hue*), `S` (saturação, do inglês *saturation*) e `I` (brilho, do inglês *brightness* ou *intensity*) são as três coordenadas que compõe o modelo HSI. Neste caso, o brilho também consiste na representação monocromática da imagem, semelhante aos casos do `Y` nos modelos YIQ e YUV, por exemplo. Os outros dois componentes são responsáveis pela representação das cores: matiz corresponde à cor pura (como, por exemplo, vermelho, azul e verde) e saturação corresponde à quantidade de luz branca existente na cor. Quanto maior a saturação, menor a quantidade de luz branca e, portanto, maior a pureza da cor. A cor rosa, por exemplo, seria a cor vermelha com pouca saturação.

Este modelo é normalmente utilizado para processamento e análise de imagens, por possuir a característica importante de separar a informação das cores nos componentes matiz e saturação enquanto o outro componente representa apenas a intensidade. Portanto, se necessário, o processamento pode ser feito simplesmente nos canais que representam as cores da imagem ou simplesmente no canal que representa a intensidade, o que muitas vezes facilita este processamento.

A representação do modelo HSI normalmente é feita através de uma forma geométrica composta por dois cones (ou hexágonos) com suas bases unidas. A linha reta vertical que liga as pontas dos dois cones representa o eixo da intensidade, tendo a cor branca no pico superior e a cor preta no pico inferior. O matiz é representada pelo ângulo formado em relação à base dos cones (a base forma um círculo). Normalmente a localização inicial

0º está sobre a cor vermelha, representando o valor zero para o matiz. Os valores vão aumentando conforme aumenta-se o ângulo, em sentido anti-horário. O último componente, a saturação, é a distância em que a cor está do centro da figura (local onde passa a linha da intensidade). Quando mais longe estiver do centro, maior a saturação. Uma ilustração desta representação pode ser vista na figura 2.3.

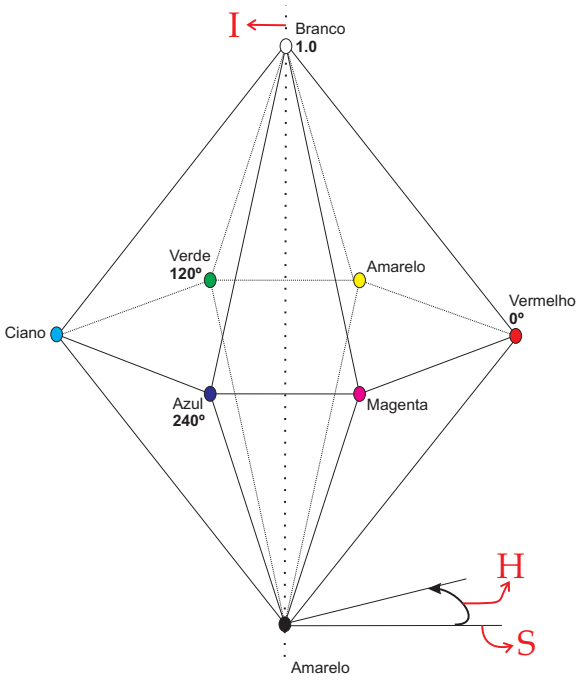

Figura 2.3: Representação geométrica do modelo HSI utilizando dois hexágonos.

As equações usadas para conversão do modelo RGB para HSI seguem a seguir (GONZALEZ; WOODS, 2002), (RUSS, 1999), (SHAPIRO; STOCKMAN, 2001):

$$S = 1 - \frac{3}{(R+G+B)}\left[min(R,G,B)\right] \tag{2.7a}$$

$$I = \frac{1}{3}(R+G+B) \tag{2.7b}$$

$$H = \begin{cases} \theta & \text{se } B \leq G \\ 360 - \theta & \text{se } B > G \end{cases} \tag{2.7c}$$

onde:

$$\theta = \cos^{-1}\left\{\frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\left[(R-G)^2+(R-B)(G-B)\right]^{\frac{1}{2}}}\right\}$$

2.3.6 CIE XYZ

Esta seção apresenta o primeiro dos espaços de cores definidos pela CIE que são abordados neste trabalho. Estes espaços têm a característica comum de serem baseados na percepção humana das cores, nos valores dos *tristimulus*, e procuram representar todas as variações de cores que podem ser vistas pelo homem. Os chamados *tristimulus values* são representados pelos componentes `X`, `Y` e `Z`, que são comentados na seqüência desta seção.

Algumas equações e conceitos apresentados aqui foram retirados de (LINDBLOOM, 2007), que apresenta uma grande variedade de informações envolvendo os espaços de cores CIE utilizados, equações de conversão entre estes espaços, processos intermediários que são necessários para as conversões, entre outros.

O espaço CIE XYZ é composto pelos três componentes primários `X`, `Y` e `Z` que foram definidos a partir de combinações dos valores vermelho, azul e verde. Essas combinações foram realizadas com o objetivo de definir um espaço com algumas características importantes, como: ser relacionado à percepção humana das cores, ser capaz de representar qualquer cor visível, ter os valores de seus componentes sempre positivos e separar a informação das cores da informação que representa a luminância (representada apenas pelo `Y`).

A conversão de RGB para o espaço CIE XYZ envolve algumas operações a mais do que as simples conversões citadas anteriormente. Basicamente, a conversão pode ser feita com o uso da equação (2.8), porém o cálculo da matriz $[M_1]$ citada requer alguns cálculos que devem ser feitos de acordo com parâmetros necessários no momento da conversão.

$$\begin{bmatrix} X & Y & Z \end{bmatrix} = \begin{bmatrix} r & g & b \end{bmatrix} \begin{bmatrix} M_1 \end{bmatrix}_{3x3} \tag{2.8}$$

Os valores (r, g, b) da equação (2.8) são os valores RGB convertidos para um espaço RGB absoluto, conforme descrito pelas equações (2.1) e (2.2), o que consiste no primeiro passo que deve ser executado.

O segundo passo consiste no cálculo da matriz $[M_1]$. Este cálculo é formado pelas equações (2.9) seguidos de uma adaptação cromática, caso necessário. Nas equações, (x_r, y_r), (x_g, y_g) e (x_b, y_b) são as coordenadas de cromaticidade do espaço RGB e (X_w, Y_w, Z_w) os valores XYZ da *white reference* utilizados.

$$\Big[M_1\Big]_{3x3} = \begin{bmatrix} S_rX_r & S_rY_r & S_rZ_r \\ S_gX_g & S_gY_g & S_gZ_g \\ S_bX_b & S_bY_b & S_bZ_b \end{bmatrix} \tag{2.9}$$

onde:

$$X_r = x_r/y_r$$
$$Y_r = 1$$
$$Z_r = (1 - x_r - y_r)/y_r$$

$$X_g = x_g/y_g$$
$$Y_g = 1$$
$$Z_g = (1 - x_g - y_g)/y_g$$

$$X_b = x_b/y_b$$
$$Y_b = 1$$
$$Z_b = (1 - x_b - y_b)/y_b$$

$$\Big[S_r \quad S_g \quad S_b\Big] = \Big[X_w \quad Y_w \quad Z_w\Big] \begin{bmatrix} X_r & Y_r & Z_r \\ X_g & Y_g & Z_g \\ X_b & Y_b & Z_b \end{bmatrix}^{-1}$$

O processo de adaptação cromática consiste em adaptar as cores para a *white reference* desejada. *White reference*, que também pode ser chamado de *white point* ou *illuminant* (entre outros sinônimos), consiste no ponto de luz branca que é utilizado para representação do espaço. Em um ambiente físico, as cores terão uma aparência diferente conforme a fonte de luz utilizada. Portanto, de acordo com o *white point* utilizado, as cores do espaço CIE XYZ também possuirão valores diferentes. A adaptação cromática serve justamente para realizar esta conversão.

A tabela 2.1 contém uma lista de *white points* que são normalmente utilizados e apresenta suas respectivas coordenadas de cromaticidade `x` e `y`, tanto no campo de visão de 2º (`CIE 1931`) quando no 10º (`CIE 1964`).

Os espaços de cores absolutos baseados no RGB possuem todos uma *white reference* padrão. Em função disto, a matriz $[M_1]$ gerada na equação (2.9) deve ser utilizada para

Tabela 2.1: Tabela de cromaticidade de alguns *white points* bastante utilizados.

	CIE 1931		**CIE 1964**	
Nome	*x*	*y*	*x*	*y*
E	1/3	1/3	1/3	1/3
A	0.44757	0.40745	0.45117	0.40594
B	0.34842	0.35161	0.3498	0.3527
C	0.31006	0.31616	0.31039	0.31905
D50	0.34567	0.35850	0.34773	0.35952
D55	0.33242	0.34743	0.33411	0.34877
D65	0.31271	0.32902	0.31382	0.33100
D75	0.29902	0.31485	0.29968	0.31740

gerar os valores XYZ de acordo com a *white reference* do espaço RGB utilizado. Caso a *white reference* não for a desejada, o processo de adaptação cromática deve ser efetuado seguindo a equação (2.13).

Na equação (2.13), (X_d, Y_d, Z_d) são as coordenadas resultantes da multiplicação da matriz $[M_2]$ por (X_s, Y_s, Z_s), as coordenadas XYZ originais. (X_{ws}, Y_{ws}, Z_{ws}) são as coordenadas da *white reference* original e (X_{wd}, Y_{wd}, Z_{wd}) da *white reference* destino, a desejada.

$$\begin{bmatrix} X_d & Y_d & Z_d \end{bmatrix} = \begin{bmatrix} X_s & Y_s & Z_s \end{bmatrix} \begin{bmatrix} M_2 \end{bmatrix}_{3x3} \tag{2.13}$$

onde:

$$\begin{bmatrix} M_2 \end{bmatrix}_{3x3} = \begin{bmatrix} M_A \end{bmatrix}_{3x3} \begin{bmatrix} \rho_d/\rho_s & 0 & 0 \\ 0 & \gamma_d/\gamma_s & 0 \\ 0 & 0 & \beta_d/\beta_s \end{bmatrix} \begin{bmatrix} M_A \end{bmatrix}_{3x3}^{-1}$$

$$\begin{bmatrix} \rho_s & \gamma_s & \beta_s \end{bmatrix} = \begin{bmatrix} X_{ws} & Y_{ws} & Z_{ws} \end{bmatrix} \begin{bmatrix} M_A \end{bmatrix}$$
$$\begin{bmatrix} \rho_d & \gamma_d & \beta_d \end{bmatrix} = \begin{bmatrix} X_{wd} & Y_{wd} & Z_{wd} \end{bmatrix} \begin{bmatrix} M_A \end{bmatrix}$$

A matriz $[M_A]$ é uma matriz de valores constantes que irá depender do método de adaptação cromática a ser utilizado. Em (LINDBLOOM, 2007) são apresentados três

Tabela 2.2: Matrizes dos métodos XYZ Scaling, Bradford e Von Kries.

XYZ Scaling			**Bradford**			**Von Kries**		
1.0	0.0	0.0	0.89510	-0.75020	0.03890	0.40024	-0.22630	0.00000
0.0	1.0	0.0	0.26640	1.71350	-0.06850	0.70760	1.16532	0.00000
0.0	0.0	1.0	-0.16140	0.03670	1.02960	-0.08081	0.04570	0.91822

métodos: XYZ Scaling, Bradford e Von Kries, e os valores da matriz de cada um dos métodos podem ser vistos na tabela 2.2.

Portanto, para realizar o processo de conversão, são necessárias três etapas:

- Conversão do modelo RGB para o espaço de cores RGB absoluto desejado;
- Encontrar a matriz de conversão $[M_1]$ e realizar a conversão;
- Aplicar a adaptação cromática, caso necessário.

Este processo é utilizado para conversão do modelo RGB para todos os modelos CIE citados nas próximas seções, pois as equações para conversão utilizadas por estes espaços partem sempre do CIE XYZ, não mais do modelo RGB. Portanto, a conversão para o espaço CIE XYZ será necessária para as conversões para os espaços CIE xyY, CIE L*a*b* e CIE L*u*v*.

2.3.7 CIE xyY

O espaço CIE xyY é composto pelos três componentes utilizados no diagrama de cromaticidade CIE exibido na figura 2.1. Note que o componente `Y` não aparece no diagrama, como já citado. Isso acontece por ele não ser necessário para o diagrama, já que ele contém informações da luminância e não da cromaticidade, e também para que o diagrama possa ser representado em duas dimensões.

Os valores `x`, `y` e `z` são chamados de valores de cromaticidade e dependem somente da matiz ou do comprimento de onda dominante e da saturação, portanto são independentes da luminância (HOFFMAN, 2005b). Eles são calculados através das equações (2.16).

$$x = \frac{X}{X + Y + Z} \tag{2.16a}$$

$$y = \frac{Y}{X + Y + Z} \tag{2.16b}$$

$$z = \frac{Z}{X + Y + Z} \tag{2.16c}$$

Através das equações acima observa-se que $x + y + z = 1$, portanto z pode ser calculado a partir de x e y. Utilizando-se disso, o espaço CIE xyY utiliza apenas os valores de x e y e inclui o componente `Y`, que determina a luminância.

Portanto, para fazer a conversão do espaço CIE XYZ para CIE xyY, utiliza-se as equações (2.16a) e (2.16b) e considera-se que o componente `Y` é o mesmo componente `Y` do espaço CIE XYZ.

2.3.8 CIE L*a*b*

O espaço CIE L*a*b* foi definido para prover uma medida computacionalmente simples das cores, de acordo com o sistema de cores de Munsell (PRATT, 2001). Ele é um espaço independente de dispositivo que foi diretamente baseado no CIE XYZ. No CIE L*a*b*, as cores que são aparentemente iguais são codificadas da mesma maneira (é colorimétrica) e as diferenças entre os diferentes matizes são percebidas uniformemente (é perceptualmente uniforme) (GONZALEZ; WOODS, 2002). Apesar de não ser possível exibir diretamente as cores do modelo CIE L*a*b* nos dispositivos de visualização atuais (monitores, por exemplo), ele é um espaço muito utilizado internamente por sistemas computacionais que trabalham com o tratamento de cores, devido à sua precisão de representação.

Assim como nos espaços CIE XYZ e CIE xyY, o CIE L*a*b* separa a informação de cromaticidade nos componentes `a*` e `b*`, enquanto `L*` refere-se à *lightness*. O conceito de *lightness* é semelhante ao conceito de luminância, porém eles diferem. Isso ocorre pois a visão humana não responde linearmente às variações no brilho, ou seja, um objeto com apenas 50% da luminância de outro objeto, não será visto como um objeto 50% mais escuro. Portanto, *lightness* pode ser definido como a percepção visual que temos da luminância (POYNTON, 2006), conceito que também é utilizado em outros espaços como o CIE L*u*v*, por exemplo.

A conversão para o espaço CIE L*a*b* parte do espaço CIE XYZ, utilizando as equa-

ções (2.17) (GONZALEZ; WOODS, 2002), (PRATT, 2001), (COLANTONI, 2004), onde (X_w, Y_w, Z_w) são as coordenadas XYZ da *white reference* utilizada.

$$L^* = \begin{cases} 116 \times (\frac{Y}{Y_w})^{\frac{1}{3}} - 16 & \text{se } \frac{Y}{Y_w} > 0.008856 \\ 903.3 \times (\frac{Y}{Y_w}) & \text{se } \frac{Y}{Y_w} \leq 0.008856 \end{cases} \tag{2.17a}$$

$$a^* = 500 \times \left[f\left(\frac{X}{X_w}\right) - f\left(\frac{Y}{Y_w}\right) \right] \tag{2.17b}$$

$$b^* = 200 \times \left[f\left(\frac{X}{X_w}\right) - f\left(\frac{Z}{Z_w}\right) \right] \tag{2.17c}$$

$$\tag{2.17d}$$

onde:

$$f(a) = \begin{cases} a^{\frac{1}{3}} & \text{se } a > 0.008856 \\ 7,787 \times a + 16/116 & \text{se } a \leq 0.008856 \end{cases}$$

2.3.9 CIE L*u*v*

O espaço CIE L*u*v* foi desenvolvido a partir dos espaços CIE L*a*b* e CIE U*V*W* e tornou-se um padrão em 1976 (PRATT, 2001). Assim como no CIE L*a*b*, ele separa a informação de cromaticidade nos componentes `u⋆` e `v⋆`, enquanto `L⋆` corresponde à *lightness*.

A conversão para o espaço CIE L*u*v* também parte do espaço CIE XYZ, utilizando as equações (2.18) (PRATT, 2001), (COLANTONI, 2004), onde (X_w, Y_w, Z_w) são as coordenadas XYZ da *white reference* utilizada.

$$L^* = \begin{cases} 25 \times (100 \times \frac{Y}{Y_w})^{\frac{1}{3}} - 16 & \text{se } \frac{Y}{Y_w} \geq 0.008856 \\ 903.3 \times (\frac{Y}{Y_w}) & \text{se } \frac{Y}{Y_w} < 0.008856 \end{cases} \tag{2.18a}$$

$$u^* = 13 \times L^* \times [U(X,Y,Z) - U(X_w,Y_w,Z_w)] \tag{2.18b}$$

$$v^* = 13 \times L^* \times [V(X,Y,Z) - V(X_w,Y_w,Z_w)] \tag{2.18c}$$

onde:

$$U(X,Y,Z) = \frac{4X}{X + 15Y + 3Z}$$

$$V(X,Y,Z) = \frac{9Y}{X + 15Y + 3Z}$$

2.4 Bibliotecas gráficas e a representação tridimensional

A criação de ambientes tridimensionais em aplicações é facilitada através do uso de bibliotecas gráficas que disponibilizam diversas rotinas para tal finalidade. Neste trabalho foram consideradas as bibliotecas OpenGL e Java3D, por possuírem alguns pontos em comum que justificam seu uso neste trabalho.

Elas possibilitam a criação de ambientes 3D, tanto simples quanto complexos, possuem rotinas para interação entre o usuário e o ambiente, são portáveis e suportadas por diversos sistemas e podem ser utilizadas com a linguagem de programação Java.

Obviamente, essas duas bibliotecas possuem características diferentes, que serão citadas nas seções seguintes.

2.4.1 Java3D

Java3D é uma biblioteca para a exibição de gráficos tridimensionais desenvolvida para a linguagem Java, garantindo com isso diversas vantagens que a linguagem possui, como portabilidade, robustez, segurança e integração com as outras bibliotecas existentes. Ela possui duas distribuições, uma construída sobre a biblioteca OpenGL e outra sobre a biblioteca DirectX, onde a última funciona apenas em plataformas Windows, devido às restrições do DirectX.

Durante o seu desenvolvimento, um dos principais objetivos era manter o nível de desempenho elevado, o que resultou na implementação de diversas funcionalidades que possibilitam elevar o desempenho das aplicações. Algumas dessas funcionalidades são a capacidade de informar à biblioteca se atributos dos objetos da cena são utilizados frequentemente ou não, informar os limites espaciais de cada componente (portanto eles não influenciam na renderização de componentes que estão fora de seus limites) e compilar toda a estrutura que contém a cena para gerar uma nova estrutura em um formato mais eficiente para a renderização. Em função disso, é possível a construção de aplicações complexas sem grandes perdas de desempenho em comparação às bibliotecas sobre a qual ela foi desenvolvida.

Sua API é bastante simples se comparada às das outras bibliotecas, o que a torna mais fácil de ser aprendida e utilizada, acelerando o processo de desenvolvimento. Vale citar que esta simplicidade não evita que possam ser criados gráficos tão robustos quanto os que poderiam ser criados com bibliotecas consideradas mais complexas, como o caso do OpenGL. As formas geométricas e os outros componentes da cena (como as transformações) são dispostos em grafos (grafo chamado *scenegraph*) através das funções disponibilizadas pela API. Este grafo é então compilado (em *run-time*), o que irá otimizá-lo para execução na máquina que está sendo utilizada.

Em função das otimizações de desempenho possíveis, a biblioteca não possui a capacidade de realizar renderizações realistas como ocorre com o uso de técnicas como *ray-tracing*. Como o módulo tridimensional deste trabalho tem o desempenho como um aspecto mais importante que o realismo da cena, Java3D foi uma das opções consideradas.

2.4.2 OpenGL

OpenGL (*Open Graphics Library*) é uma especificação padrão de uma API multiplataforma para produção de gráficos 2D e 3D. Esta especificação de métodos (rotinas, funções) é utilizada na prática através do uso de bibliotecas que implementam estes métodos, portanto o OpenGL será considerado como uma biblioteca na continuação deste documento. Existem variações entre as implementações existentes, porém todas devem estar de acordo com as definições que o OpenGL assume.

Sendo uma das bibliotecas gráficas mais utilizadas atualmente (OPENGL, 2007), ela divide com o DirectX o domínio sobre as aplicações gráficas desenvolvidas, especialmente nos jogos eletrônicos atuais que têm parte de seu sucesso relacionado à qualidade de seus gráficos. Apesar de o DirectX ser muito utilizado e também possuir diversas características ótimas, ele não é considerado neste trabalho por ser proprietário e não portável (só é suportado em plataformas Windows), características opostas às do OpenGL.

Podem ser citadas diversas características importantes que tornaram esta biblioteca tão utilizada, como um ótimo desempenho, portabilidade, boa documentação, escalabilidade, confiabilidade, entre outras. Em função destas características (e pode ser adicionada a facilidade de aprendizado, quando comparada ao DirectX), ela passou a ser muito utilizada no meio acadêmico, principalmente, como instrumento didático e como parte do desenvolvimento de diversos trabalhos.

Atualmente são encontradas implementações da biblioteca OpenGL para diversos sistemas operacionais (entre eles podem ser citados UNIX, Windows 95/98, Windows 2000, Windows NT, Linux, OPENStep e Mac OS) e ela pode ser utilizada com diversas linguagens de programação (C, C++, Java, Fortran, entre tantas), o que afirma a sua portabilidade. O OpenGL também é suportado por uma grande quantidade de placas gráficas, que implementam, em seus circuitos, as rotinas da biblioteca (não necessariamente todas elas), o que aumenta consideravelmente a eficiência das aplicações que a utilizam.

Outra característica importante desta biblioteca é ter sido modelada para suportar a inclusão de extensões, que possibilitam adicionar à ela novas funcionalidades que não estão especificadas na API oficial. Isso possibilita que, por exemplo, vendedores de placas gráficas incorporem extensões em seus produtos para dar suporte à novas funcionalidades que ainda não foram incluídas no núcleo do OpenGL mas que existem em suas placas gráficas. Ocasionalmente são realizadas revisões formais da biblioteca e algumas extensões que tenham se tornado amplamente conhecidas podem ser incluídas no núcleo.

Sua API é bastante extensa é pode ser considerada difícil de aprender e utilizar. Ela permite a implementação de rotinas bastante complexas e disponibiliza diversas opções de otimização, o que pode ser utilizado para melhorar o desempenho e o gerenciamento de recursos. Apesar desta complexidade, muita informação pode ser encontrada na documentação da API e na Internet, o que ameniza o problema citado.

A figura 2.4 mostra iamgens da aplicação `3D AFM Surf ActiveX control`, que é desenvolvida com o uso de OpenGL que pode ser encontrada em (MACROSYSTEM, 2007).

2.4.3 JOGL (Java OpenGL)

O JOGL é um *binding* da linguagem Java para a biblioteca OpenGL, ou seja, ele permite que funções da biblioteca OpenGL sejam utilizadas por aplicações desenvolvidas em Java. Ele faz parte de um projeto desenvolvido pelo *Game Technology Group* da *Sun Microsystems*, empresa responsável pela linguagem Java, além de ter o suporte da SGI, criadora do OpenGL.

JOGL possibilita acesso integral às rotinas especificadas pela API do OpenGL (atualmente OpenGL 2.0) e às diversas extensões disponibilizadas pelos fabricantes de *hardware*, além de ter integração com os *frameworks* AWT e Swing, utilizados em Java para

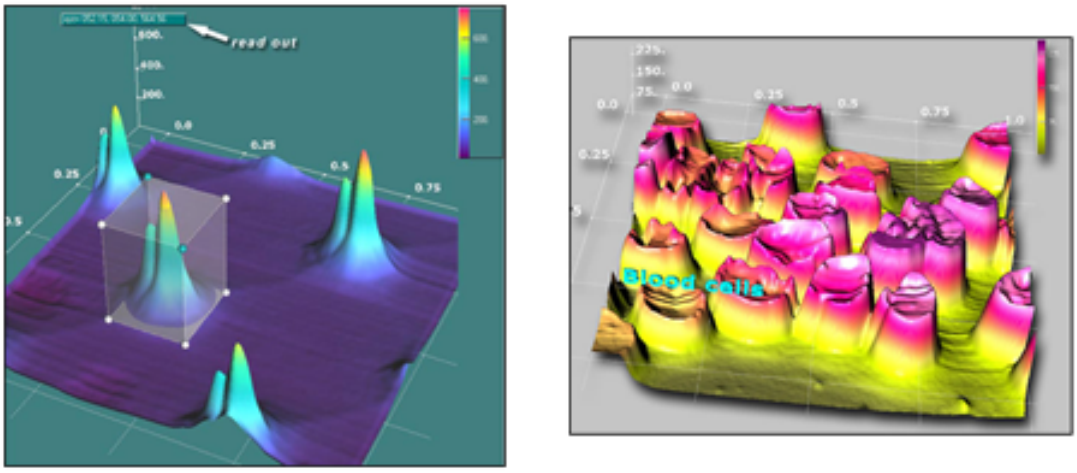

Figura 2.4: Exemplo de uma aplicação desenvolvida utilizando OpenGL: 3D AFM Surf ActiveX control.

a criação de interfaces gráficas. Seu uso é semelhante aos usos "clássicos" da biblioteca OpenGL (como no caso de utilizá-la com a linguagem C), com algumas modificações necessárias para a adaptação ao Java e outras para usufruir de benefícios da linguagem, como a própria orientação a objetos que não é possível na linguagem C.

Desde o ano 2003 estava em processo de aprovação uma JSR (JSR 231) na *Java Community Process* para padronização dos *bindings* OpenGL para a linguagem Java, adotando como base o projeto JOGL (PETERSEN; RUSSELL, 2004). Nos últimos meses esta requisição foi aprovada e está atualmente em processo. Ter sido a base para este processo mostra que o JOGL é uma das melhores escolhas no momento, além de estar em constante atualização e dentro dos padrões que estão sendo definidos.

Diversas aplicações de demonstração, tutoriais e a documentação do JOGL podem ser encontradas no *website* do projeto responsável pelo seu desenvolvimento, em (JOGL, 2006).

2.5 Trabalhos relacionados

2.5.1 Couleur ColorSpace Application

O `Couleur ColorSpace` é uma ferramenta desenvolvida pelo professor francês Philippe Colantoni, disponível em (COULEUR, 2006).

O propósito desta ferramenta é bastante semelhante ao propósito deste trabalho. Ela disponibiliza métodos de conversão entre espaços de cores e também uma forma de vi-

sualizar a distribuição das cores em uma interface tridimensional. Ele possui suporte aos espaços de cores RGB, CIE XYZ, CIE xyY, I1I2I3, UVW, LSLM, CIE L*a*b*, CIE L*u*v*, LHC, HSV, HSV Polar, CMY, HSI, HSI Polar, LHS, YUV, YIQ e YCbCr, e os modos de visualização das imagens em 2D e 3D.

A ferramenta utiliza a biblioteca GTK para construção da interface gráfica e é disponibilizada para os sistemas operacionais Windows e Unix. Ela possui um módulo para visualização das imagens 2D, chamado de *Image Viewer*, que utiliza a biblioteca ImageMagick para leitura das imagens, suportando assim uma enorme quantidade de formatos de imagens. Através do *Image Viewer*, o usuário pode acessar a janela de conversões, onde é escolhido o espaço de cores a ser utilizado e onde são configurados os parâmetros da conversão. A partir desta, o usuário pode exibir a imagem em 2D ou 3D, de acordo com as escolhas feitas (figura 2.5).

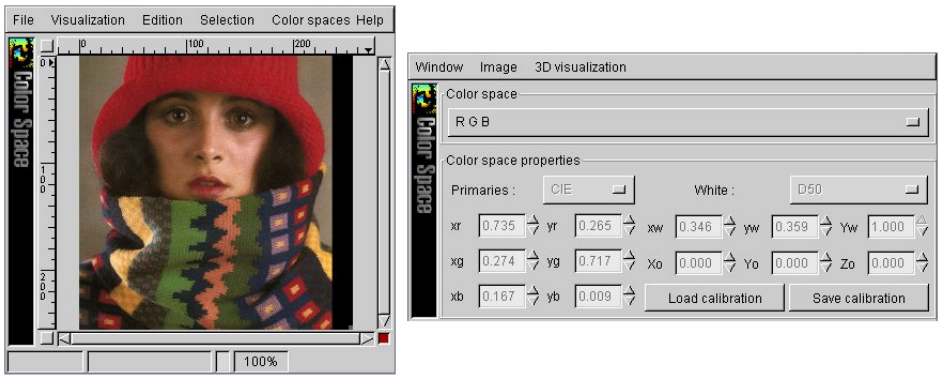

Figura 2.5: Couleur ColorSpace: *Image Viewer* à esquerda e janela para seleção e configuração do espaço de cores à direita.

Imagens obtidas de (COULEUR, 2006)

Esta ferramenta possui diversos pontos fortes. Ela é bastante rápida, suporta diversos formatos de imagens, a interface 3D possibilita várias formas de interação e podem ser feitas configurações bastante detalhadas nas propriedades dos espaços de cores. Em relação às configurações, além de possibilitar a seleção do *illuminant* e do espaço RGB (chamado de *primaries*), ela permite que o usuário visualize os valores de cromaticidade desses e altere-os, resultando em mudanças no gráfico exibido.

Além disso, ela possui uma opção interessante que consiste na exibição de espaços de cores híbridos. Espaços de cores híbridos são formados por coordenadas de diversos espaços de cores diferentes (por exemplo, o `R` do RGB, o `H` do HSI e o `Y` do CIE xyY), e

sua representação em duas ou três dimensões pode ter um resultado bastante interessante e importante para a análise da imagem.

A interface 3D (figura 2.6), além de ter um bom desempenho, como já citado, possui várias opções de configuração e interação. O usuário pode realizar rotações, translações, modificar as cores dos componentes e do fundo, optar pela exibição dos eixos, exibição das projeções nos três eixos, exibição da área do cubo RGB, entre outros.

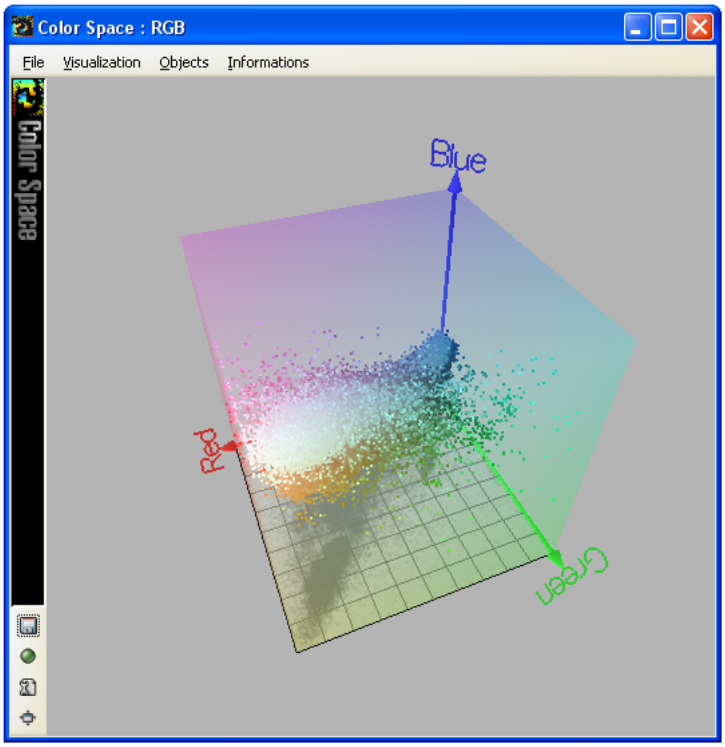

Figura 2.6: Couleur ColorSpace: Exemplo da interface 3D.

Por estes vários fatores, esta é considerada a melhor aplicação semelhante a este trabalho entre as que foram encontradas.

2.5.2 3D Color Inspector Plug-in para ImageJ

O `3D Color Inspector` é um *plug-in* para a aplicação `ImageJ` que foi desenvolvido por Kai Uwe Barthel e está disponível em (BARTHEL, 2006). O `ImageJ` é um aplicativo para processamento de imagens desenvolvido na linguagem Java que possibilita tarefas comuns de processamento de imagens além de possuir algumas outras características importantes, como: suporte a macros e *plug-ins*, ser *open source*, melhor desempenho para processamento de imagens em relação à outras aplicações desenvolvidas em Java (IMAGEJ, 2006), pode ser considerado um *toolkit* para desenvolvimento de aplicações,

entre outras.

Este *plug-in* exibe a imagem em 2D e, ao lado, uma interface 3D exibindo a distribuição de cores desta imagem (figura 2.7). Ele suporta os espaços de cores RGB, YUV, YCbCr, HSB, HSV, HSL, HMMD, CIE xyY, CIE XYZ, Lab e Luv e também disponibiliza algumas opções de ajuste na imagem, como correção do brilho, contraste e saturação. Além disso são disponibilizados 5 modos de visualização das cores na interface 3D: todas as cores, de acordo com a freqüência com que elas aparecem, histograma, redução através do *median cut* (algoritmo para redução do número de cores da imagem) e redução através do *wu quant* (outro algoritmo para redução das cores de uma imagem, que foi criado por Xiaolin Wu e pode ser encontrado em (WU, 1992)). Outra funcionalidade importante é a possibilidade de segmentação da imagem, onde deve ser escolhida uma opção de coloração (como preto e branco, cores originais e preto, cores originais e branco, entre outras possíveis) e o valor da profundidade das cores, que fará a divisão das cores de acordo com as colorações escolhidas.

A interface 3D possui boas opções de rotação, translação e mudança na escala, porém limitadas. Seu desempenho também é limitado, apesar de não ser comprometedor para visualização e interação com o gráfico. Como pontos positivos, temos os cinco modos de exibição das cores, a segmentação, as possibilidades de ajuste de contraste, brilho e saturação e a integração com o `ImageJ`, que possibilita que sejam executadas diversas outras tarefas sobre a imagem.

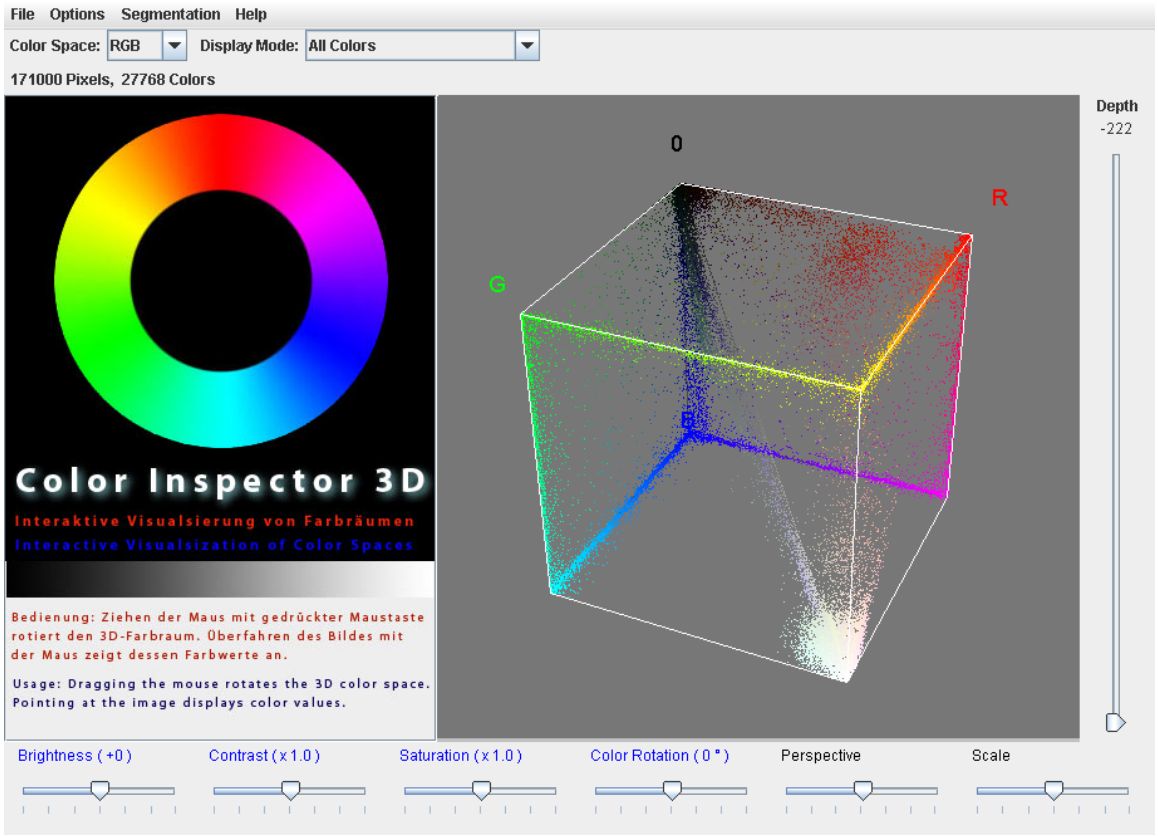

Figura 2.7: Imagem do plug-in Color Inspector.

3 APRESENTAÇÃO E DESENVOLVIMENTO DA FERRAMENTA

Este capítulo apresenta o processo de desenvolvimento da ferramenta, detalhando a modelagem realizada e as técnicas e ferramentas utilizadas para a implementação.

3.1 Métodos de conversão

Como já citado anteriormente, os métodos de conversão fazem parte de um módulo do sistema independente dos outros módulos, portanto ele não possui dependências em relação à interface. Nas subseções seguintes é explicado como este módulo foi modelado, apresentando as classes básicas e os métodos de conversão que foram desenvolvidos.

3.1.1 Modelagem

A idéia de fazer o módulo de conversões independente dos outros módulos teve como objetivo isolá-lo e tornar seu uso possível em outras aplicações ou por outros módulos da aplicação na qual a ferramenta está incluída, podendo ser utilizado como uma biblioteca. A simples construção de classes que não façam referências a classes de outros módulos já garante que o módulo ao qual estas classes pertencem será independente dos outros módulos. Porém, isso não agrupa os componentes e não facilita o uso dos mesmos.

Para resolver esta questão, foi criada a classe `Converters`, que é responsável por identificar os conversores existentes e disponibilizar uma forma fácil e padronizada de acesso a eles. A identificação dos conversores existentes é feita com o uso da API *reflection* disponibilizada pela linguagem Java. Esta API permite acesso às classes, interfaces e objetos que estão na máquina virtual do Java (JVM) durante a execução da ferramenta (em *run-time*), o que possibilita diversas ações como, por exemplo, criar objetos de clas-

ses cujos nomes não são conhecidos antes da execução, determinar a classe de um objeto já existente e obter uma lista de propriedades de determinada classe.

Com isso é possível acessar todas as classes de um pacote (um pacote padrão onde os conversores devem estar) e verificar quais delas são realmente conversores e quais não são. Depois de identificados, os conversores são armazenados em uma tabela *hash*, facilitando os acessos que serão feitos a eles posteriormente.

A identificação de quais classes realmente são conversores é feita com uso da interface `ConverterInterface`. Todas as classes que implementarem esta interface devem implementar os métodos nela declarados, portanto serão conversores. Este processo de identificar os conversores existentes é realizado na inicialização da classe `Converters`, processo que deve ocorrer apenas uma vez durante a execução e que pode ser realizado na inicialização da aplicação (na inicialização do `Arthemis`, por exemplo), apesar de não ser um processo que comprometa o desempenho da ferramenta.

Para acesso aos conversores, a classe `Converters` disponibiliza funções para obter os nomes daqueles conversores que estiverem disponíveis e para obter as instâncias destes. Este processo é padrão para todos eles, independente da complexidade da conversão e dos parâmetros de configuração necessários.

A figura 3.1 mostra um diagrama UML simplificado exibindo a relação entre as classes que implementam os conversores, a interface que elas implementam e a classe que agrupa e disponibiliza acesso a elas. O diagrama também mostra os métodos da interface `ConverterInterface` que possuem a função de converter os dados, obter as opções de configuração do conversor e obter o descritor do conversor, respectivamente. Também pode ser vistos os métodos da classe `Converters`, para obter nomes e instâncias dos conversores.

Um conversor pode dar suporte a diversas conversões e não somente uma, ou seja, ele pode dar suporte a múltiplos espaços de cores caso necessário. Para descrever quais os espaços suportados por cada conversor, é usada a classe `ConverterDescriptor`, que pode ser vista como uma coleção de objetos da classe `ConverterUnit`. Um `ConverterUnit` descreve um espaço suportado por um conversor, possuindo um nome e algumas propriedades que serão utilizadas para configuração da interface 3D quando este espaço estiver sendo utilizado. Através deste descritor é possível encontrar qual o conversor que suporta o espaço desejado e então utilizar este conversor para os cálculos.

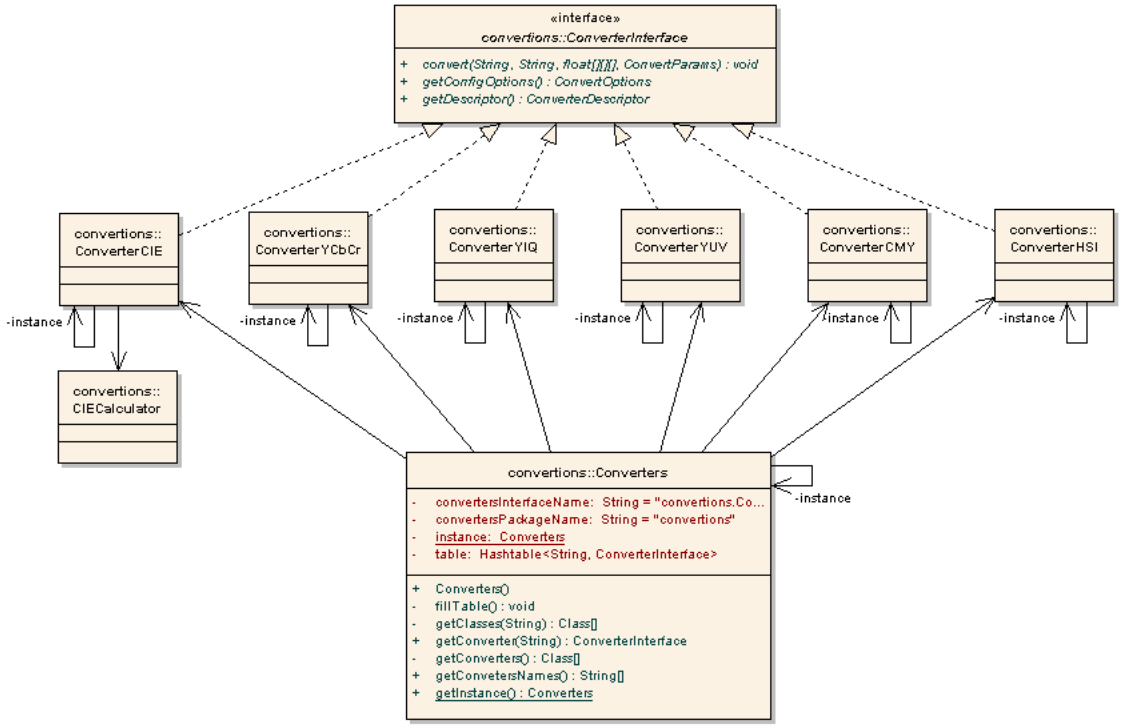

Figura 3.1: Diagrama UML mostrando os conversores desenvolvidos, a interface implementada por eles e a classe que os integra.

Para armazenamento em memória dos dados da imagem (a cor dos *pixels*) e conversão desses dados, é utilizada uma matriz de três dimensões de dados do tipo `float`. Como já citado, as conversões são sempre do espaço RGB para outro espaço, portanto os dados de entrada sempre deverão estar no modelo RGB e normalizados, ou seja, todos os valores devem estar no intervalo [0,1]. Armazenar os dados desta maneira torna bastante simples a manipulação e ajuda muito no desempenho, que poderia ficar comprometido caso fossem utilizadas estruturas mais complexas como, por exemplo, a classe `Raster` disponibilizada pelo Java (que foi considerada no início do projeto).

De acordo com a maneira como os conversores foram modelados, existe a possibilidade de permitir que o usuário carregue imagens armazenadas em um modelo diferente do modelo RGB. Para isso basta que sejam implementadas as equações para conversão do modelo desejado para o modelo RGB e que sejam realizadas algumas mudanças na interface da ferramenta, mas não há necessidade de mudanças na estrutura dos conversores.

3.1.2 Métodos desenvolvidos

Foram desenvolvidos métodos de conversão para dar suporte aos seguintes espaços de cores: CIE XYZ, CIE xyY, CIE L*a*b*, CIE L*u*v*, CMY, HSI, YCbCr, YIQ e YUV, sem contar o modelo RGB que não necessita um conversor já que assumimos que as imagens carregadas na aplicação já estão neste espaço.

A figura 3.2 mostra a relação entre os espaços de cores suportados pela aplicação, onde se pode perceber que todos eles derivam do espaço RGB, direta ou indiretamente. Os espaços do CIE (com exceção do CIE XYZ) são aqueles que herdam indiretamente do RGB. Eles necessitam a conversão para o espaço CIE XYZ para que então a conversão para eles seja feita. Por este motivo que todas as conversões foram implementadas partindo do espaço RGB.

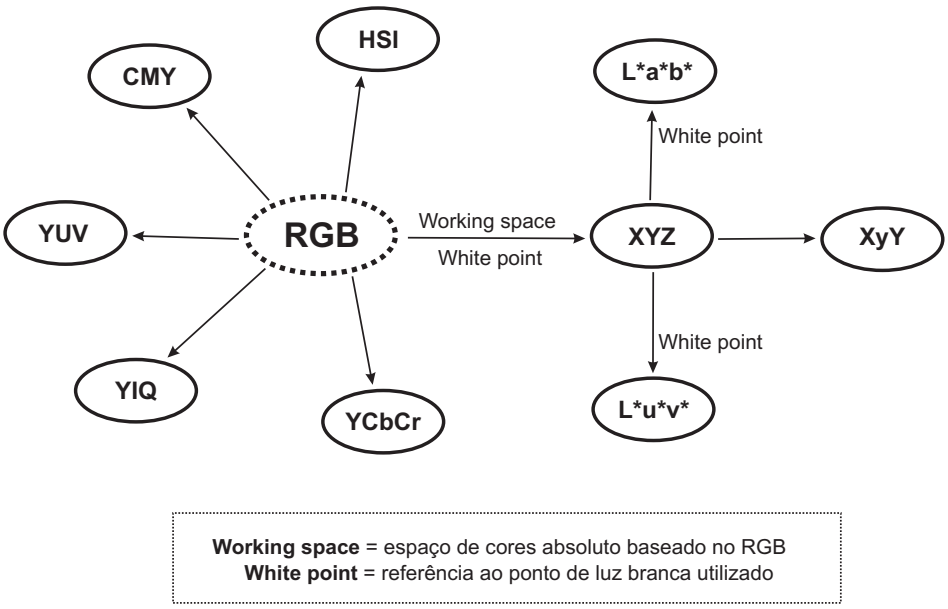

Figura 3.2: Diagrama mostrando a relação entre os espaços de cores implementados.

As equações matemáticas utilizadas para as conversões são as mesmas que foram apresentadas na seção 2.3, algumas com pequenas alterações para otimização de processamento ou facilidade de implementação (alterações que não alteram o resultado da equação).

Quanto à conversão para o espaço CIE XYZ, são necessários como parâmetro o chamado *RGB working space*, ou seja, o espaço de cores absoluto baseado no RGB, e o *white point* que será utilizado. Para conversão do CIE XYZ para CIE L*u*v* e para CIE L*a*b*, também é necessário que seja determinado o *white point* utilizado. Além

disso, como citado na seção 2.3.6, outros dois parâmetros podem ser utilizados nessas conversões. São eles o método de adaptação cromática a ser utilizado e o campo de visão para obter a cromaticidade dos *white points* utilizados. A tabela 3.1 apresenta a lista de parâmetros disponíveis para as conversões para os espaços CIE.

Tabela 3.1: Parâmetros disponíveis para a conversão para os espaços CIE.

Parâmetro	**Valores possíveis**
RGB *working space*	Adobe RGB, Apple RGB, Best RGB, Beta RGB, Bruce RGB, CIE RGB, ColorMatch RGB, DonRGB4, ECI RGB, Ekta Space PS5, NTSC RGB, PAL/SECAM RGB, ProPhoto RGB, SMPTEC-C, sRGB, Wide Gamut RGB
White point	A, B, C, D50, D55, D65, D75, E
Campo de visão	2°, 10°
Método de adaptação cromática	XYZ Scaling, Bradford, Von Kries

3.2 Interface e módulo gráfico

O segundo módulo desenvolvido foi o módulo da interface. Ele é composto pelo componente de criação do ambiente 3D (chamado `Viewer`), pelas janelas para configuração dos conversores e exibição do `Viewer` e por outros componentes auxiliares.

A figura 3.3 exibe um diagrama que ilustra o relacionamento entre os componentes da interface e entre a interface e o módulo de conversões. O diagrama é dividido em dois núcleos: o núcleo dependente do `Arthemis` e o núcleo independente. O componente `Viewer`, apesar de fazer parte do módulo da interface, é independente, pois não faz uso de nenhum componente existente no `Arthemis`, podendo ser utilizado por qualquer outro sistema. O mesmo ocorre com o módulo de conversões. Já os outros componentes da interface foram criados especificamente para o `Arthemis`, utilizando classes e componentes desse sistema, portanto só podem ser utilizados dentro do mesmo.

Nas subseções seguintes serão comentados em mais detalhes os componentes mais importantes da interface: o `Viewer`, as janelas de configuração (o `ViewerFrame` e o `MainFrame`) e o gerenciador da interface (`Manager`).

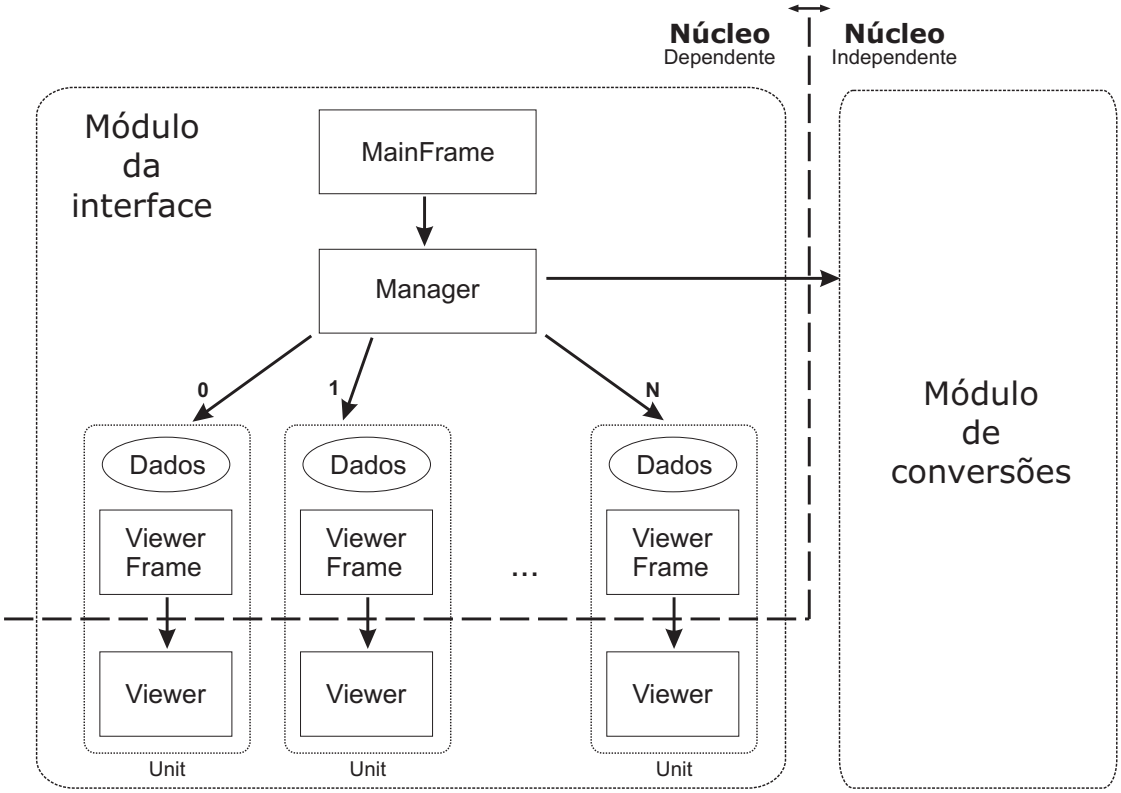

Figura 3.3: Diagrama de relacionamento entre os componentes da interface.

3.2.1 Componente 3D (Viewer) e o uso do JOGL

Como já citado, o `Viewer` é o componente responsável pela criação do ambiente tridimensional. Ele contém toda a implementação da exibição do gráfico 3D, do posicionamento dos objetos na tela conforme os dados da imagem, das opções de interação com o usuário e das possibilidades de personalização do ambiente. A figura 3.4 mostra algumas imagens do ambiente 3D utilizando imagens e espaços de cores diferentes para ilustrar algumas das opções de visualização possíveis.

Este é o componente que faz o uso do JOGL, comentado na seção 2.4.3. O JOGL possui integração com os *frameworks* AWT e Swing, que são utilizados por este trabalho e pelo `Arthemis` para a criação dos componentes da interface gráfica. Esta integração permite que este componente seja facilmente utilizado pelos outros componentes que compõe a interface.

Por ser um componente independente, ele não faz referência a nenhum componente do sistema `Arthemis`, sendo simplesmente utilizado por este sistema. Para criação do gráfico 3D ele necessita dos dados da imagem que deve ser exibida e também outros parâmetros opcionais de configuração do ambiente. É necessário que ele receba os dados

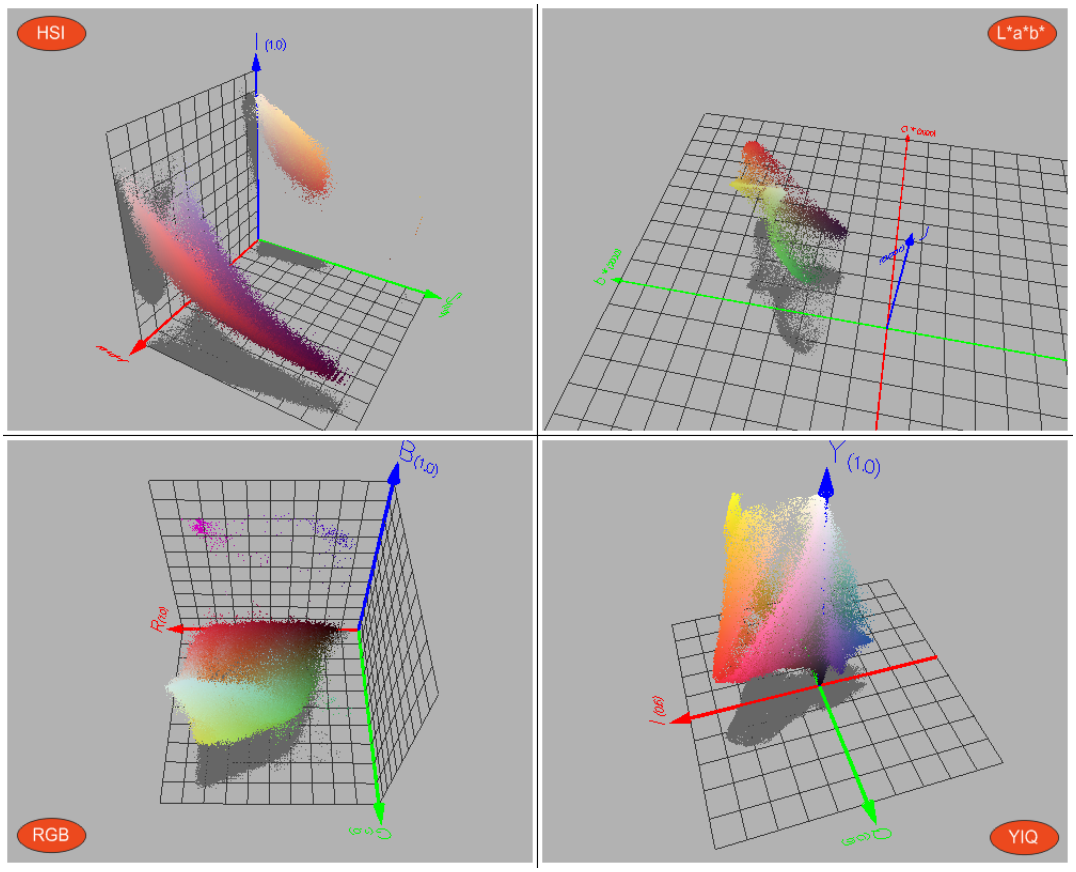

Figura 3.4: Imagens do ambiente 3D desenvolvido.

da imagem no espaço RGB, para definição das cores dos objetos, e no espaço que será utilizado para exibir o gráfico, para definição da posição dos objetos neste. Estes dados são obtidos através do gerenciador da interface antes que o componente seja exibido.

Como citado, o posicionamento dos objetos no gráfico (mais especificamente, dos pontos na nuvem de pontos) é realizado de acordo com os valores das coordenadas de cada *pixel* da imagem no espaço de cores escolhido pelo usuário. Um *pixel* com a cor preta, por exemplo, teria os valores (0, 0, 0) para as coordenadas no espaço RGB, portanto seria posicionado no ponto de encontro dos três eixos no gráfico, onde todos eles possuem valor zero. O mesmo ponto seria representado pelos valores (1, 1, 1) no espaço CMY, e seria posicionado no local onde as três coordenadas possuem valor um (no vértice oposto ao vértice de origem do cubo gerado pelos três eixos de coordenadas). Já a coloração dos pontos no gráfico é feita através dos valores RGB de cada *pixel*, para representar a cor que este *pixel* representa na imagem real.

No modo de visualização implementado, cada *pixel* da imagem é representado por um único ponto no gráfico. Como já citado na introdução deste trabalho, um outro modo

de visualização seria a exibição do gráfico no formato de um histograma, onde cada cor possível de ser exibida no gráfico seria representada por um objeto com tamanho equivalente à quantidade de *pixels* da imagem que possuem esta cor. Exemplificando, uma imagem que possui 20 *pixels* com a cor branca e 10 *pixels* com a cor preta, apresentaria, no gráfico, um objeto na posição da cor branca com dimensões duas vezes maiores em relação ao objeto de cor preta.

Os parâmetros para configuração do ambiente tridimensional são vários, alguns deles são definidos pelo conversor do espaço de cores que está sendo utilizado e alguns definidos pelo usuário. Os primeiros consistem nos parâmetros básicos necessários para criação do gráfico (valores máximos e mínimos para as coordenadas, nome das coordenadas, escala do gráfico, entre outros), que são obtidos dos conversores e passados para o `Viewer` na sua criação. Os parâmetros definidos pelo usuário (número de grades, projeções e cores dos objetos, por exemplo) já possuem um valor padrão na inicialização do `Viewer` e podem ser modificados pelo usuário através da janela que exibe o componente.

A tabela 3.2 mostra os parâmetros de configuração do `Viewer` de acordo com os componentes existentes nele, indicando se esses parâmetros estão disponíveis ao usuário ou não.

Este componente também implementa as funções de interação entre o usuário e o ambiente 3D, que possibilitam que o usuário visualize o gráfico de diversos ângulos diferentes. O usuário pode realizar rotações e translações nos eixos `x`, `y` e `z` do gráfico, utilizando teclas pré-definidas do teclado ou o *mouse*, clicando e arrastando. Utilizando teclas de atalho no teclado, o usuário pode salvar o estado atual de translações e rotações e restaurar este estado mais adiante, o que facilita a visualização rápida do gráfico de diversos ângulos sem necessidade de reposicionar o gráfico sempre que isso for necessário. É possível armazenar e restaurar até 10 estados diferentes do gráfico.

Além da possibilidade de rotacionar e transladar o gráfico utilizando o teclado, com ele o usuário também pode realizar outras operações, como ligar ou desligar o autorotacionar e mover o gráfico de volta para a sua posição inicial. Os comandos possíveis são exibidos em uma janela de ajuda externa que pode ser vista na figura 3.5.

Tabela 3.2: Parâmetros para configuração do gráfico 3D.

Componente	Usuário	Ação
Escala	Sim	Modificar o tamanho da escala
Fundo	Sim	Modificar a cor do fundo
Rotação	Sim	Ligar/desligar auto-rotação
Eixos	Sim	Exibir/esconder eixos
Projeções	Sim	Exibir/esconder projeções em cada um dos três eixos e alterar a cor das projeções
Grades	Sim	Exibir/esconder grades em cada um dos três eixos, alterar a cor das grades e alterar o número de células em cada uma das grades
Polígonos	Sim	Alterar a maneira de exibição de polígonos: *Fill*, *Wireframe* ou *Points*
Bandas	Não	Alterar o nome e a cor de cada uma das bandas da imagem
Eixos	Não	Alterar valores máximo e mínimo dos eixos

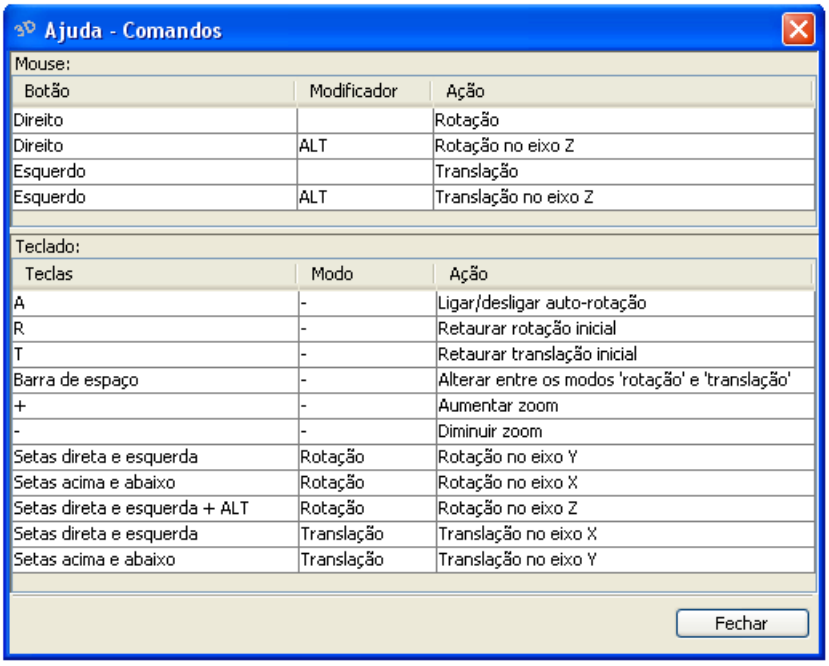

Figura 3.5: Janela de ajuda exibindo os comandos possíveis no gráfico 3D.

3.2.2 Janelas de configuração

As janelas de configuração são os componentes utilizados para configuração dos parâmetros e exibição do `Viewer`. Dois componentes foram criados, o `MainFrame` e o

`ViewerFrame`.

O primeiro corresponde à janela que possui as opções de configuração dos conversores e que possibilita a execução do componente 3D. Esta é a janela principal da ferramenta, aquela que será exibida quando o usuário executar a ferramenta através dos menus do `Arthemis`. Os parâmetros básicos que devem ser escolhidos são: a imagem que será exibida no gráfico 3D, o espaço de cores a ser utilizado e, dependendo do espaço escolhido, são necessárias as opções de configuração deste. A figura 3.6 ilustra este componente com alguns parâmetros de configuração selecionados.

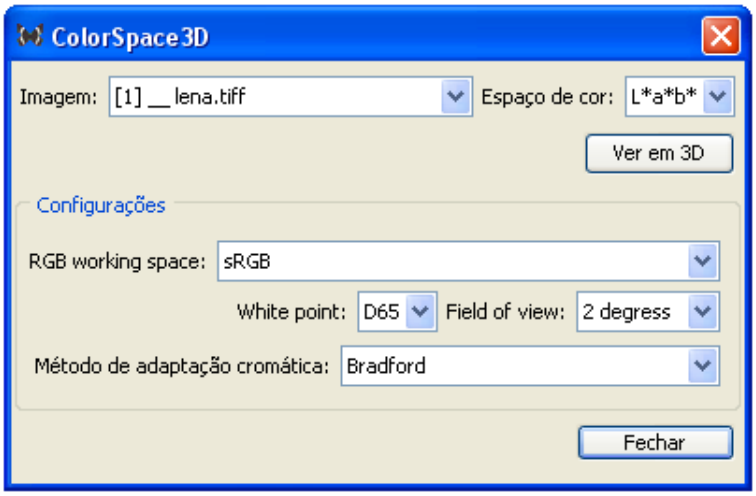

Figura 3.6: Janela de configurações da ferramenta.

Este componente possui dois aspectos importantes que devem ser citados. O primeiro é que ele controla a lista de imagens automaticamente através de componentes disponibilizados pelo `Arthemis`, ou seja, quando uma imagem é aberta, fechada ou modificada no sistema, ele reconfigura seus componentes automaticamente, não sendo necessário reiniciar a ferramenta. O outro item é a identificação em *run-time* dos conversores e criação dos componentes da interface de acordo com os conversores existentes. Isso significa que a lista de espaços é montada dinamicamente quando a ferramenta é executada, assim como as listas de opções de configuração dos espaços que foram encontrados. Isso possibilita a inclusão de novos conversores sem necessidade de alterações na interface mesmo que os valores possíveis para os parâmetros sejam diferentes, exceto nos casos em que novos tipos de parâmetros (e não apenas valores) forem necessários.

O segundo componente, o `ViewerFrame`, consiste na janela que exibe o gráfico 3D. Ele possui o gráfico 3D ocupando a maior parte de sua área e alguns menus de acesso às

opções de configuração deste gráfico. Estas opções de configuração são as citadas na tabela 3.2 que são disponíveis ao usuário e são exibidas em uma nova janela, onde o usuário as configura de acordo com suas necessidades e essas mudanças refletem imediatamente na exibição do gráfico.

3.2.3 Gerenciador da interface

O gerenciador da interface, como o próprio nome diz, é o responsável por gerenciar os componentes da interface, mais especificamente os componentes `ViewerFrame` criados, seus gráficos 3D e os dados que serão atribuídos a esses gráficos. Além disso, ele é o único componente da interface que faz o acesso ao módulo dos conversores.

O gerenciador possui uma instância única que é criada e utilizada pelo `MainFrame`, inicialmente para obtenção dos conversores e suas propriedades. Quando é necessária a criação de um novo gráfico 3D, o gerenciador é utilizado para, em um primeiro momento, fazer as conversões entre os espaços de cores e então criar, configurar e exibir o gráfico. Ele também faz a verificação do número de bandas da imagem escolhida pelo usuário. Se for uma imagem de apenas uma banda, ela é convertida para 3 bandas para ser aceita na ferramenta 3D. Apesar de não ter muita utilidade visualizar uma imagem monocromática no ambiente 3D, futuramente, com a implementação de novos modos de visualização do gráfico, o suporte à imagens de apenas uma banda pode ser necessário.

Para organização interna, são utilizados componentes chamados `Unit`, que unem os dados e componentes responsáveis pela exibição de um (apenas um) gráfico. Uma `Unit` é criada pelo gerenciador cada vez que é necessário exibir um novo gráfico, e conterá o componente `Viewer`, o `ViewerFrame` para exibição e os dados que serão atribuídos ao gráfico. Com o uso desta `Unit` fica mais simples a remoção de um gráfico e seus dados quando eles não forem mais necessários.

Este componente permite a criação de quantos gráficos forem necessários. Basta que o usuário selecione as opções no `MainFrame` e solicite a exibição de um novo gráfico que uma nova `Unit` será criada, configurada e o gráfico será exibido.

3.3 Integração com o Arthemis

A integração da ferramenta com o sistema `Arthemis` foi feita através do núcleo dependente citado na figura 3.3. Os componentes foram criados de acordo com os padrões

do sistema, utilizando algumas classes e métodos já existentes.

Além destes componentes do módulo da interface, alguns componentes adicionais foram necessários para a integração, como, por exemplo, as *actions*. Estas, são os componentes responsáveis pela criação dos menus de acesso no `Arthemis` e pela execução dos métodos, que no caso é a criação e exibição da janela de execução da ferramenta.

Outro aspecto importante é a possibilidade de alteração da linguagem utilizada no sistema, funcionalidade que é disponibilizado pelo `Arthemis`. Para utilizar esta característica, foi necessária a edição de alguns arquivos de configuração do `Arthemis` para incluir todas as constantes textuais utilizadas e fazer uso de classes existentes no sistema para acesso a essas constantes.

A figura 3.7 mostra a janela de configuração da ferramenta desenvolvida já integrada no `Arthemis`, que estava utilizando a língua inglesa como linguagem padrão.

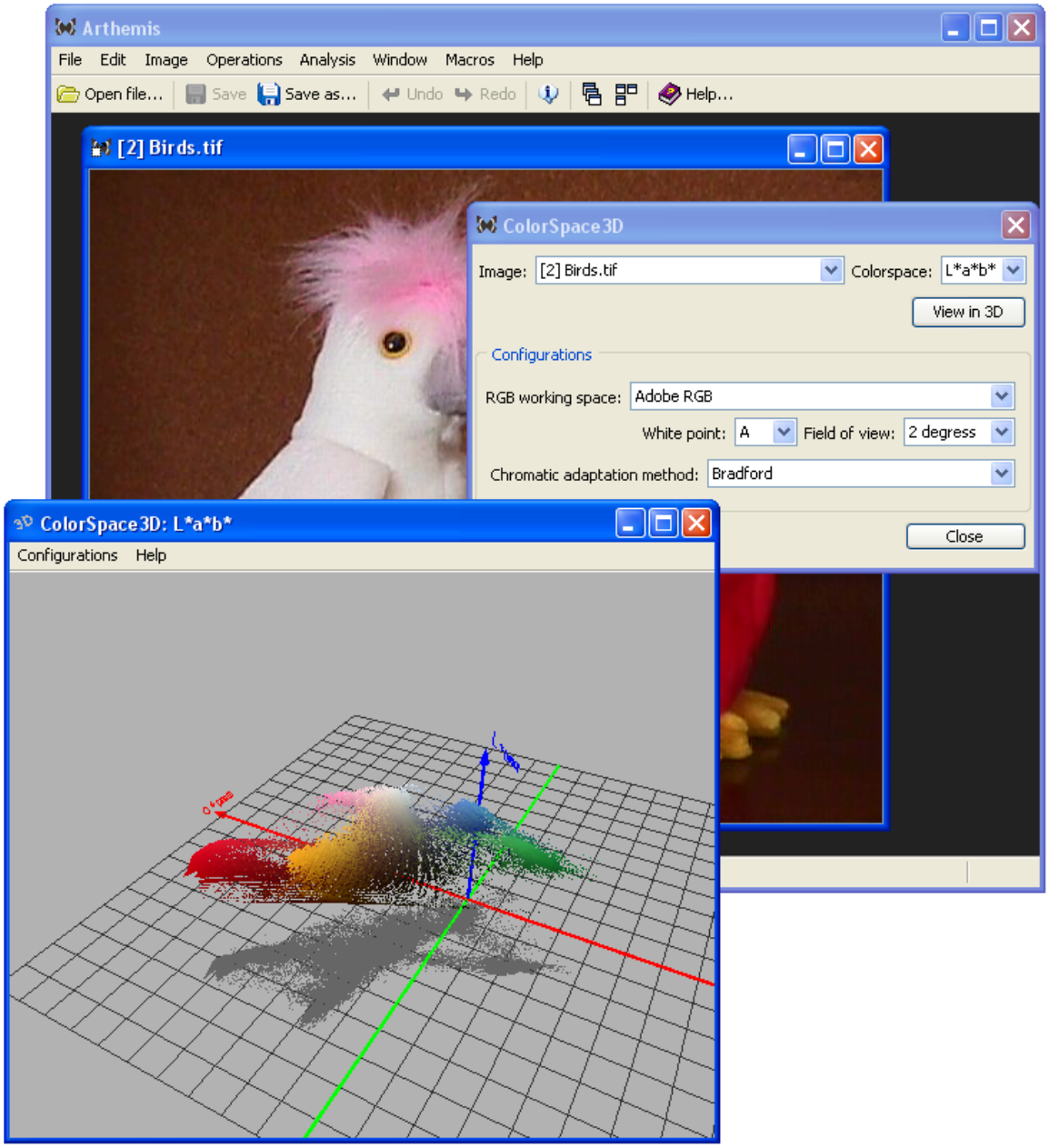

Figura 3.7: Imagem da ferramenta integrada no Arthemis exibindo suas janelas na língua inglesa.

4 AVALIAÇÃO

Este capítulo apresenta os pontos positivos e negativos da ferramenta desenvolvida com base nos objetivos propostos no início do trabalho. Inicialmente serão avaliados o módulo dos conversores e o módulo gráfico e, por fim, a ferramenta será comparada com as duas ferramentas semelhantes a ela já citadas na seção 2.5.

Para os testes de desempenho e uso de recursos, foi utilizado um computador com processador Pentium IV 2.0GHz, 512MB de memória RAM e placa de vídeo aceleradora modelo NVIDIA GeForce FX5200 de 64bits e 128MB de memória.

4.1 Conversores

O módulo dos conversores tinha como objetivos principais o desempenho, a facilidade de inclusão de novos conversores e o suporte ao maior número de espaços de cores possíveis.

O desempenho de uma conversão depende muito do tamanho da imagem que está sendo utilizada e também do computador no qual a ferramenta está sendo executada, obviamente. Além disso, o desempenho varia de acordo com o espaço de cores ao qual a imagem está sendo convertida. Apesar de que a maioria dos espaços implementados possui uma fórmula simples para conversão, alguns espaços como o L*a*b* e L*u*v* precisam de alguns passos intermediários que influenciam no desempenho geral. Entre os espaços suportados, aqueles que apresentaram a conversão mais demorada foram os espaços L*a*b* e o HSI, respectivamente.

O processo de conversão acontece somente uma vez a cada exibição de um novo gráfico, portanto não é um processo repetitivo que necessite ser instantâneo (como ocorre com alguns processos necessários para interação no gráfico 3D). Para uma imagem de

512 *pixels* de altura e de largura, o processo possui duração de cerca de 1 segundo para os casos mais demorados.

Para imagens maiores o processo torna-se mais lento, e pode-se dizer que o tempo de execução é linear ao número de *pixels* da imagem que está sendo convertida. Em testes com uma imagem de 960 *pixels* de largura e 800 *pixels* de altura, a conversão levou entre 2,5 e 3 segundos nos casos mais demorados, cerca de três vezes o tempo de conversão da imagem citada anteriormente. Este teste mostra a linearidade citada, uma vez que a primeira imagem possui 262.144 *pixels* e a segunda 768.000 *pixels*, cerca de três vezes mais do que a primeira.

Quanto ao suporte aos espaços de cores, a ferramenta suporta 10 diferentes espaços, utilizando 6 conversores que implementam 9 métodos de conversão, já que a conversão para o modelo RGB não é necessária, como já citado anteriormente. Atualmente, existe uma quantidade muito maior do que apenas 10 espaços de cores, porém este não é um número baixo de espaços suportados em comparação às outras ferramentas semelhantes, e, no momento, é suficiente para o trabalho.

A facilidade de inclusão de novos conversores vem para suprir a necessidade futura de novos espaços na ferramenta. Da forma como a ferramenta foi modelada, adicionar um novo conversor para suportar novos espaços de cores é bastante simples. Este conversor precisa, obviamente, ser implementado de acordo com os padrões propostos (que consiste em apenas implementar os métodos de uma interface) e então inserido no pacote padrão dos conversores. Ele será automaticamente detectado pelo módulo dos conversores e pela interface, sem necessidade de modificações nestes.

O único caso onde um novo conversor não poderá ser incluído na ferramenta apenas com os passos acima é quando ele necessitar de algum parâmetro de conversão diferente dos parâmetros utilizados pelos conversores já implementados (que são os parâmetros para os modelos do CIE, exibidos na tabela 3.1). Nestes casos, a interface necessitará de mudanças que incluam novos componentes visuais para exibir as opções de configuração desses novos parâmetros. Nos casos em que o novo conversor necessite apenas de novos valores para algum parâmetro já existente, a detecção é automática. Basta que este novo valor seja incluído no módulo dos conversores.

Com isso pode-se dizer que os objetivos principais foram alcançados de forma satisfatória. Além disso, o módulo dos conversores também tem a vantagem de ser isolado

da interface, portanto, com sua integração no `Arthemis`, pode ser utilizado para realizar conversões sempre que necessário, não apenas com o uso da interface da ferramenta desenvolvida.

4.2 Módulo gráfico

O módulo gráfico, ou módulo 3D, tinha como principais objetivos disponibilizar opções de interação com o usuário, manter um bom desempenho durante essas interações e possibilitar, ao usuário, a personalização dos componentes do gráfico. Além disso, também era necessário um bom desempenho na inicialização do gráfico, o que inclui o processo de montagem do mesmo e de criação da nuvem de pontos (etapa mais lenta da inicialização).

Para interação com o gráfico, o usuário tem a possibilidade de realizar rotações e translações, ambas possíveis nos três eixos de coordenadas. Estas operações podem ser realizadas com o uso do *mouse*, clicando e arrastando, ou através de algumas teclas pré-definidas do teclado. Estas são as operações básicas para que o usuário possa visualizar o gráfico da melhor maneira possível e elas podem ser facilmente utilizadas por este usuário.

Algumas outras operações de interação com o gráfico poderiam ser importantes para o usuário, como, por exemplo, modificar o ponto central das rotações. Este ponto central está sempre posicionado no centro do cubo formado pelos eixos onde todas as coordenadas são positivas (se todas possuem valor máximo igual a 1, o ponto central terá coordenadas 0,5 para os três eixos, por exemplo). Em casos onde as coordenadas assumem valores negativos, pode ser difícil posicionar o gráfico de uma maneira na qual a nuvem de pontos possa ser visualizada da maneira desejada.

Para manter um bom desempenho durante essas interações, diversas otimizações foram feitas no código que renderiza os componentes do gráfico na tela. Este código esta implementado em uma função que é executada temporariamente pelo OpenGL, em intervalos de tempo muito pequenos. Portanto, o tempo de execução desta função é muito importante e relevante para o desempenho geral da ferramenta.

A primeira otimização foi controlar as chamadas à função citada para que ela só seja executada quando realmente for necessário, ou seja, quando alguma modificação foi feita no gráfico. Esta é a otimização mais básica e também a que apresentou o melhor resultado. Outra otimização utilizada foi a criação de listas do OpenGL para a renderização

dos eixos, grades, projeções e a própria nuvem de pontos. Estas listas são criadas na inicialização do gráfico e agrupam diversos comandos que são responsáveis por renderizar cada um dos componentes citados. Todos os cálculos necessários para a renderização são realizados apenas uma vez, durante a criação das listas. Estas listas serão utilizadas para exibir os componentes na tela no momento da renderização do gráfico e, como a maioria dos cálculos necessários já foram realizados, o processo de renderização será acelerado.

Além das duas otimizações citadas, também foram feitas diversas outras otimizações menores, todas procurando acelerar o processo de interação com o usuário.

O desempenho final não pode ser considerado como ótimo, mas é satisfatório. Assim como nas conversões, aqui o tamanho das imagens também influencia muito no desempenho das interações. Para imagens com 512 *pixels* de largura e altura, as rotações e translações são praticamente instantâneas. Isso não acontece para imagens maiores ou quando são exibidos diversos outros itens que foram selecionados pelo usuário (projeções e grades). Ainda utilizando imagens com as dimensões citadas, mesmo que sejam exibidas grades e projeções em todos os eixos, o desempenho continua bom, porém é fácil perceber a diferença em relação ao mesmo gráfico quando um número reduzido de opções estão selecionadas.

O outro objetivo citado era o de disponibilizar diversas opções de configuração do gráfico para o usuário. Este usuário pode exibir/esconder os eixos e seus valores, escolher em quais eixos serão exibidas as projeções e as grades, modificar a quantidade de células das grades, alterar a cor do fundo, a cor das projeções e a cor das grades e alterar a escala geral do gráfico. Além dessas configurações, o usuário também pode escolher entre três modos de desenho dos polígonos (preenchidos, usando linhas ou usando pontos), parâmetro que pode ser utilizado para melhorar o desempenho de exibição do gráfico.

Algumas outras opções de configuração para os componentes existentes no gráfico poderiam ser adicionadas e muitas outras seriam necessárias com a adição de novos componentes e funcionalidades, porém foram utilizadas aquelas consideradas as mais importantes para o usuário. Este usuário pode modificar o ambiente para ficar mais adequado e melhor de ser visualizado alterando suas cores, gráficos grandes podem ser visualizados mais facilmente com alterações na escala e modificando o número de células das grades é possível saber mais precisamente o valor das coordenadas de regiões da nuvem de pontos, por exemplo.

4.3 Comparação com ferramentas semelhantes

Na seção 2.5 foram apresentadas duas ferramentas relacionadas à esta, que agora serão utilizadas na comparação para ressaltar alguns pontos positivos e negativos do trabalho desenvolvido.

Serão utilizados os seguintes parâmetros para a comparação: o desempenho, o uso de memória, os espaços de cores suportados, as possibilidades de interação e configuração do gráfico e os parâmetros possíveis para os métodos de conversão.

4.3.1 Desempenho e uso de memória

A ferramenta `Couleur ColorSpace` é, sem dúvidas, a aplicação que apresenta o melhor desempenho, tanto durante as conversões quanto durante as interações com o gráfico. O desempenho do gráfico desta aplicação só é reduzido quando certas opções de exibição forem selecionadas pelo usuário, como exibir a nuvem de pontos formada por esferas e utilizar iluminação, por exemplo.

Porém, utilizando as configurações básicas iniciais, o desempenho desta aplicação é certamente melhor que o das outras duas ferramentas. Um dos motivos é o uso da linguagem Java por este trabalho. Por ser uma linguagem interpretada, entre outros motivos, as aplicações normalmente têm o desempenho mais baixo se comparadas a aplicações desenvolvidas em linguagens como C ou C++, principalmente aplicações que utilizam bibliotecas 3D como o OpenGL.

O desempenho da ferramenta desenvolvida é bastante semelhante ao da aplicação `3D Color Inspector`. Ambas são desenvolvidas em Java, porém a segunda não utiliza OpenGL, e sim uma maneira própria de renderização de gráficos 3D. O processo de conversão entre espaços de cores não apresenta grandes diferenças de desempenho entre as ferramentas, o que também ocorre com a interação com o gráfico 3D. Em alguns momentos, o `3D Color Inspector` apresenta um desempenho um pouco melhor na interação com o gráfico, mas em outros casos, como quando é utilizada a rotação automática, seu desempenho é pior.

Em relação ao uso de memória, novamente o `Couleur ColorSpace` possui vantagem em relação às outras duas ferramentas. As diferenças são pequenas, pois a quantidade de memória utilizada depende muito do tamanho da imagem que está sendo visualizada. A forma como cada ferramenta armazena os dados da imagem internamente irá determi-

nar a diferença entre o uso de memória de cada ferramenta.

Quanto ao `3D Color Inspector`, seu uso de memória é praticamente igual ao da ferramenta desenvolvida. Esta pode utilizar mais memória por possibilitar a visualização de diversos gráficos da mesma imagem ao mesmo tempo, enquanto que o `3D Color Inspector` possibilita a exibição de apenas um gráfico para cada imagem e cada instância da ferramenta. Porém, visualizando apenas um gráfico da mesma imagem, a diferença de memória utilizada é pequena.

4.3.2 Interação com o gráfico e opções de configuração

As opções de interação com o gráfico das três ferramentas são basicamente as mesmas, embora disponibilizadas de maneiras diferentes, através de comandos diferentes. O `Couleur ColorSpace` disponibiliza as rotações e translações com o uso do *mouse*, enquanto o `3D Color Inspector` faz uso do *mouse* e de alguns controles na interface para certas operações. A vantagem desta ferramenta em relação às outras é possibilitar tanto o uso do *mouse* quanto do teclado, para todos os comandos de interação possíveis.

Em relação às opções de configuração, o `Couleur ColorSpace` é o que apresenta mais opções ao usuário. Ele possibilita exibir ou esconder eixos, valores dos eixos, projeções, grades, utilizar iluminação, modificar a cor da maioria dos componentes, modificar a maneira de exibição dos polígonos, escolher entre diversos objetos para montagem da nuvem de pontos, entre outras opções. Apesar de que a ferramenta desenvolvida possibilite a configuração de várias opções citadas, o `Couleur ColorSpace` ainda possui mais formas de configurar o gráfico.

Quanto ao `3D Color Inspector`, ele não possui diversas das opções existentes nas outras duas ferramentas, entretanto ele possibilita a visualização do gráfico de maneiras alternativas. A forma geométrica gerada pelas coordenadas irá depender do espaço de cores utilizado, o que só é disponibilizado por este aplicativo. O espaço HSI, por exemplo, possui suas coordenadas em formato de cone, e não formato cúbico como o RGB. Além disso ele também possibilita a redução do número de cores exibidas no gráfico e da visualização em forma de histograma.

Estas são duas vantagens que o `3D Color Inspector` possui em relação à ferramenta desenvolvida, porém, ele não disponibiliza as outras opções existentes na última (exibição de projeções, por exemplo). Portanto, o `Couleur ColorSpace` pode ser

considerado como o superior, enquanto o `3D Color Inspector` e a ferramenta desenvolvida se assemelham, mesmo que disponibilizando opções diferentes de configuração.

4.3.3 Espaços suportados e parâmetros de configuração

Em números, este trabalho suporta 10 espaços de cores, o `3D Color Inspector` suporta 11 e o `Couleur ColorSpace` suporta 24, além de suportar espaços híbridos, que permitem a criação de inúmeros novos espaços caso forem necessários. Os espaços escolhidos mudam de ferramenta para ferramenta, porém, alguns espaços mais utilizados estão presentes em todas, como o RGB, HSI, CIE XYZ e CIE L*a*b*.

A maior diferença entre elas está nas opções de configuração das conversões. O `3D Color Inspector` utiliza valores pré-definidos e não permite que nenhum parâmetro seja configurado durante as conversões. Já o `Couleur ColorSpace` permite que sejam definidos o *white point* e o espaço RGB absoluto que será utilizado, além de possibilitar a alteração dos valores de cromaticidade de ambos.

Neste aspecto, o `3D Color Inspector` é certamente inferior à ferramenta desenvolvida, enquanto o `Couleur ColorSpace` possui a vantagem de possibilitar a alteração da cromaticidade dos parâmetros escolhidos e de salvar e carregar todas essas configurações para uso posterior. As vantagens do trabalho realizado são possibilitar a escolha do método de adaptação cromática a ser utilizado e disponibilizar mais opções de *white points* e espaços RGB absolutos do que o `Couleur ColorSpace`.

4.3.4 Comparações adicionais

Uma característica importante que deve ser citada é a possibilidade de alterar o brilho, o contraste e a saturação da imagem com o uso do `3D Color Inspector`, o que torna possível ajustar a imagem para melhorar sua visualização. A ferramenta desenvolvida neste trabalho não inclui essas funcionalidades por fazer parte do sistema `Arthemis`. Neste sistema, estão disponíveis as funcionalidades citadas e diversos outros métodos de processamento de imagens, que podem ser utilizados nas imagens que serão analisadas no gráfica 3D, portanto não há necessidade de inclusão desses métodos diretamente na ferramenta.

Algumas outras funcionalidades importantes que não foram incluídas na ferramenta desenvolvida mas que são disponibilizadas por uma das outras duas ferramentas, ou am-

bas, são: a possibilidade de segmentação de cores na imagem, exibição do gráfico utilizando espaços de cores híbridos, possibilidade de exibir a imagem em 2D após a conversão para o espaço escolhido e as opções de exibição do gráfico em forma de histograma e com redução de cores.

Apesar de não possuir estas funcionalidades citadas, a ferramenta desenvolvida possui toda a base necessária para fácil inclusão de novos espaços de cores e também todas as funcionalidades básicas para interação e configuração do gráfico, fazendo uso de algumas outras características importantes das duas ferramentas citadas.

5 CONCLUSÃO

Este trabalho apresentou o desenvolvimento de uma ferramenta para análise da distribuição de cores de imagens digitais, utilizando conceitos de espaços de cores, visualização tridimensional e engenharia de *software* para a modelagem e desenvolvimento do aplicativo.

A modelagem adotada foi um ponto importante para tornar fácil e rápida a ampliação da ferramenta, principalmente em relação ao suporte a novos espaços de cores. As otimizações utilizadas no ambiente 3D também foram muito importantes para melhorar o desempenho da ferramenta durante as interações com o usuário.

Depois de finalizada, foi criada uma aplicação para a utilização da ferramenta (uma aplicação *stand-alone*) e ela também foi incluída no sistema `Arthemis`, onde poderá ser utilizada e ampliada em trabalhos futuros.

A seção 4 avaliou a ferramenta comparando-a com outros 2 trabalhos já desenvolvidos, onde ficou claro que ela não é a melhor ferramenta no momento, mas que apresenta a base necessária para cumprir os objetivos propostos e que possui alguns diferenciais importantes em relação às outras.

Diversas funcionalidades podem ser incluídas na ferramenta para melhorá-la e ampliar sua utilidade. A seção 5.1 apresentará algumas dessas funcionalidades.

5.1 Trabalhos futuros propostos

As possibilidades de melhora da ferramenta desenvolvida são inúmeras. A seguir serão listadas e comentadas algumas sugestões de trabalhos futuros que podem tornar mais robusto o que já foi desenvolvido, incluindo alguns tópicos que já foram citados ao longo deste documento.

- Exibir a imagem em 2D após a conversão para o espaço de cores selecionado;
- Segmentar a imagem de acordo com áreas selecionadas no gráfico 3D. Estas áreas podem possuir diversos formatos (cubos, esferas, entre outros), e podem ser interpretadas de diversas maneiras. Pode-se converter a imagem para preto e branco, onde a área selecionada corresponde a uma das cores e o resto da imagem à outra, ou simplesmente converter a área selecionada para uma cor definida enquanto o resto da imagem continua com os mesmo valores. Estas são apenas duas de diversas opções que podem ser dadas ao usuário para segmentação da imagem através da seleção de cores no ambiente 3D;
- Disponibilizar maneiras de agrupamento e seleção das cores que serão exibidas, como o caso de exibir apenas aquelas que possuem freqüência maior que um determinado número (definido pelo usuário) e o caso de exibir o gráfico em forma de histograma;
- Inclusão de novas funcionalidades no gráfico 3D, como a possibilidade de escolher entre diversas formas geométricas para representação de cada item da nuvem de pontos, incluir iluminação, mostrar a representação do cubo RGB no espaço, entre tantas;
- Criar novas formas de visualização dos espaços, alternativas à forma existente que exibe 3 eixos em um formato cúbico. Alguns espaços, como o HSI, podem ser melhor representados e visualizados de outras maneiras, como através de um cone ou de um cilindro, por exemplo. Podem ser implementadas diversas formas de visualização e disponibilizadas ao usuário, para que este escolha a que melhor representa o espaço de cores utilizado;
- Ampliar o número de espaços de cores suportados através da inclusão de novos conversores;
- Possibilitar o carregamento de imagens em espaços de cores alternativos ao RGB. Para isso é necessário implementar as equações de conversão dos modelos suportados para o RGB e também disponibilizar novos comandos na interface para que o usuário selecione o espaço no qual a imagem será carregada.

- Melhorar o desempenho através de otimizações no gráfico 3D. O desempenho das conversões dificilmente poderá ser melhorado, pois ele depende da fórmula de conversão, que dificilmente será alterada. Porém, o desempenho dos processos de interação com o gráfico 3D pode ser melhorado através de otimizações no código do componente, além daquelas que já foram feitas.

REFERÊNCIAS

BARTHEL, K. U. [3D Color Inspector/Color Histogram]. Internationale Medieninformatik, Berlin, Germany, 2006. Disponível em: <http://rsb.info.nih.gov/ij/plugins/color-inspector.html>. Acesso em: 02 fev. 2007.

COLANTONI, P. **Color Space Transformations**. Disponível em: <http://colantoni.nerim.net/download/colorspacetransform-1.0.pdf>. Acesso em: 02 fev. 2007.

COULEUR. [Couleur.org Website]. Disponível em: <http://www.couleur.org>. Acesso em: 02 fev. 2007.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. 2.ed. Upper Saddle River, New Jersey: Prentice Hall, 2002. cap 6.

HOFFMAN, G. [Documents by Gernot Hoffmann]. Emden, Germany. Disponível em: <http://www.fho-emden.de/~hoffmann/howww41a.html>. Acesso em: 02 fev. 2007.

HOFFMAN, G. **CIE Color Space**. Disponível em: <http://www.fho-emden.de/~hoffmann/ciexyz29082000.pdf>. Acesso em: 05 fev. 2007.

ICC. [Introduction to the ICC profile format]. Website do ICC. Disponível em: <http://www.color.org/iccprofile.html>. Acesso em: 07 fev. 2007.

IMAGEJ. [ImageJ - Image Processing and Analysis in Java]. Disponível em: <http://rsb.info.nih.gov/ij/>. Acesso em: 05 fev. 2007.

JOGL. [JOGL API Project Home Page]. Disponível em: <https://jogl.dev.java.net>. Acesso em: 05 fev. 2007.

LINDBLOOM, B. J. [BruceLindbloom.com]. Disponível em: <http://brucelindbloom.com>. Acesso em: 02 fev. 2007.

MACROSYSTEM, M. [Aplicação 3D AFM Surf ActiveX control]. Disponível em: <http://www.msmacrosystem.nl/spm_afm/index.html>. Acesso em: 02 fev. 2007. Disponível como aplicações executáveis e componentes ActiveX para VC++, C#, .NET, VB. Sistemas operacionais Windows NT/9x/ME/2000/XP/Vista.

OPENGL. [OpenGL Overview]. Website oficial do OpenGL. Disponível em: <http://www.opengl.org/about/overview>. Acesso em: 08 fev. 2007.

PETERSEN, D.; RUSSELL, K. **3D Application and Game Development With OpenGL**. 2004. p 10. Apresentação para o evento JavaOne. Disponível em: <https://jogl.dev.java.net/ts1361.pdf>. Acesso em: 06 fev. 2007.

POYNTON, C. [Color FAQ - Frequently Asked Questions Color]. Nov, 2006. Disponível em: <http://www.poynton.com/PDFs/GammaFAQ.pdf>. Acesso em: 05 fev. 2007.

PRATT, W. K. **Digital Image Processing**: piks inside. 3.ed. Los Altos, California: John Wiley and Sons, Inc., 2001.

RUSS, J. C. **The Image Processing Handbook**. 3.ed. Raleigh, North Carolina: CRC Press LLC, 1999.

SHAPIRO, L. G.; STOCKMAN, G. C. **Computer Vision**. 1.ed. [S.l.]: Prentice Hall, 2001.

WIKIPEDIA. [YCbCr]. Disponível em: <http://en.wikipedia.org/wiki/YCbCr>. Acesso em: 02 fev. 2007.

WU, X. Color Quantization by Dynamic Programming and Principal Analysis. **ACM Transactions on Graphics**, [S.l.], v.11, n.4, p.348–372, Oct. 1992.