UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

BIBLIOTECA PARA ANÁLISE DE DADOS EM IMAGENS ESTEREOSCÓPICAS

RICARDO IURI SALVADOR

BLUMENAU
2014

2014/1-21

RICARDO IURI SALVADOR

BIBLIOTECA PARA ANÁLISE DE DADOS EM IMAGENS ESTEREOSCÓPICAS

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Marcel Hugo, Mestre - Orientador

BLUMENAU
2014

2014/1-21

BIBLIOTECA PARA ANÁLISE DE DADOS EM IMAGENS ESTEREOSCÓPICAS

Por

RICARDO IURI SALVADOR

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Marcel Hugo, Mestre – Orientador, FURB
Membro:	Prof. Dalton Solano dos Reis, Mestre, FURB
Membro:	Prof. Joyce Martins, Mestre, FURB

Blumenau, 10 de julho de 2014

RESUMO

Diversos dispositivos conseguem capturar imagens estereoscópicas. No entanto, não existem muitas ferramentas que auxiliam desenvolvedores de software a trabalhar com esse tipo de imagem. Este trabalho tem como objetivo não só auxiliar como também motivar programadores a criar novas aplicações estereoscópicas e/ou adicionar essa funcionalidade em aplicações já existentes. Com auxílio da biblioteca que foi desenvolvida neste trabalho é possível abrir, extrair informações, aplicar filtros e gerar novas imagens estereoscópicas a partir de arquivos de vídeo, imagens ou até mesmo *streaming* de câmeras. Para explicar o funcionamento dessa biblioteca foi desenvolvida uma ferramenta que aplica todas as funções disponibilizadas pela mesma. A ferramenta, além de demonstrar, também explica o funcionamento de cada método presente na biblioteca. Além disso, os códigos, tanto da biblioteca quanto da ferramenta, estão disponibilizados abertamente para o público, permitindo assim que outros realizem modificações ou melhorias neste. Os resultados obtidos mostram que a biblioteca desenvolvida neste trabalho pode abstrair detalhes que precisam ser considerados quando o objetivo é trabalhar com imagens estereoscópicas. Estes podem não só facilitar passos da implementação de aplicações estéreo como também aumentar o espoco que a sua aplicação abrange.

Palavras-chave: Visão computacional. Estereoscopia. Biblioteca de software.

ABSTRACT

Several devices can capture stereoscopic images. However, there aren't many tools available to help developers to work with stereo images. This work aims not only to assist but also to motivate developers to create new stereo applications or even enable stereo support into existing applications. Using the library that was developed in this work you can open, extract information, apply filters and generate new stereoscopic images from image files, video files or even camera streams. To explain how this library works, a tool was also developed; this tool applies all the functions provided by the library. This tool not only demonstrates but also explains the function of each algorithm available in the library. Furthermore, the codes from both, the library and the tool are openly available to the public, thus allowing others to perform modifications and improvements in it. The results show that the library developed in this paper can abstract details that need to be considered when you want to work with stereoscopic images, these steps may not only ease the implementation of your stereo application but as well increase the scope that your application currently covers.

Keywords: Computer vision. Stereoscopy. Software library.

LISTA DE ILUSTRAÇÕES

Figura 1 - Experimento de visão estereoscópica. 14
Figura 2 – Possíveis opções para o desenvolvimento de conteúdo 3D em cada estágio do seu ciclo de vida 15
Figura 3 – Técnica usada pelo *driver* da Nvidia para obter uma imagem estereoscópica a partir de um ambiente 3D 17
Figura 4 – O *frustum* para cada olho é uma versão horizontalmente traduzida do mono *frustum*. 18
Figura 5 – Resolução completa (superior) *frame compatible*/compressão espacial (inferior) . 20
Figura 6 – Representação de uma figura 2D junto da informação de profundidade. 20
Figura 7 – Óculos 3D para visualizar anáglifos. 22
Figura 8 – Criação de imagem estereoscópica usando o método anáglifo. 23
Figura 9 – Técnica *active shutter glasses* 24
Figura 10 - Funcionamento da técnica *dual displays* 25
Figura 11 – *Oculus Rift* (*Head-mounted display*), fabricado pela Oculus VR 26
Figura 12 - Funcionamento da técnica de *Parallax Barrier* 27
Figura 13 – Diferença de paralaxe obtida com o movimento da esteira. 30
Figura 14 – Aplicação demo antes e depois da conversão 32
Figura 15 – Método *toe-in* 33
Figura 16 – Método *ff-axis* 33
Figura 17 - Diagrama de casos de uso da ferramenta 35
Quadro 1 – Caso de uso UC01 em detalhes 36
Quadro 2 – Caso de uso UC02 em detalhes 36
Quadro 3 – Caso de uso UC03 em detalhes 36
Quadro 4 – Caso de uso UC04 em detalhes 36
Quadro 5 – Caso de uso UC05 em detalhes 37
Quadro 6 – Caso de uso UC06 em detalhes 37
Figura 18 - Diagrama de classes 38
Figura 19 - Diagrama de sequência 40
Figura 20 - Itens necessários para compilar a biblioteca em modo debug 42
Figura 21 - Itens necessários para compilar a biblioteca em modo release 42
Figura 22 – Referência para os *headers* da biblioteca 43
Figura 23 – Referência para as *libraries* da biblioteca. 43

Figura 24 - Estrutura da biblioteca 44
Quadro 7 – Conteúdo da classe `StereoVideo` 44
Quadro 8 – Conteúdo da classe `StereoTools` 44
Quadro 9 – Metodo `applyFilter` da classe `StereoTools`. 45
Quadro 10 – Função `splitFrames` da classe `StereoTools`. 46
Quadro 11 – Função `joinFrames` da classe `StereoTools`. 47
Figura 25- Estrutura da ferramenta 48
Quadro 12 – Código que instancia a tela da ferramenta 48
Quadro 13 – Laço na ferramenta que percorre todos os quadros de um vídeo. 48
Quadro 14 - Metodo setFrameToPicture 49
Figura 26 - Tela inicial da ferramenta 50
Figura 27 – Seleção do modo de captura 50
Figura 28 – Vídeo sendo reproduzido na ferramenta 50
Figura 29 – Selecionando compressão do arquivo. 51
Figura 30 – Imagem carregada pela ferramenta 51
Quadro 15 – Comparação entre trabalhos correlatos 53

LISTA DE SIGLAS

2D – Duas dimensões

3D – Três dimensões

CGI – *Computer Generated Images*

DVI – *Digital Visual Interface*

HD – *High Definition*

HDMI – *High-Definition Multimedia Interface*

OCR – *Optical Character Recognition*

OpenCV – *Open source Computer Vision library*

OpenGL – *Open Graphics Library*

OS – *Operational System*

OU – *Over/Under*

RF – Requisito Funcional

RNF – Requisito Não Funcional

SBS – *Side By Side*

TOF – *Time Of Flight*

UC – *Use Case*

UML – *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO 10
1.1 OBJETIVOS DO TRABALHO 12
2 FUNDAMENTAÇÃO TEÓRICA 13
2.1 ESTEREOSCOPIA 13
2.2 CICLO DE VIDA DO CONTEÚDO 3D 15
2.2.1 Produção do conteúdo 16
2.2.1.1 Captura por câmeras 16
2.2.1.2 Imagens geradas por computador (CGI) 16
2.2.1.3 Conversão de imagens 2D para 3D 18
2.2.2 DISTRIBUIÇÃO DO CONTEÚDO 19
2.2.2.1 Codificação do conteúdo 19
2.2.2.2 Transmissão do conteúdo 21
2.2.2.3 Decodificação do conteúdo 21
2.2.3 Técnicas de exibição 21
2.2.3.1 Anaglyph Glasses 22
2.2.3.2 Active shutter glasses 23
2.2.3.3 Polarized 3D glasses 24
2.2.3.4 Head-mounted displays 25
2.2.3.5 Autostereoscopy 26
2.3 VISÃO COMPUTACIONAL 27
2.3.1 OPENCV 28
2.4 TRABALHOS CORRELATOS 30
2.4.1 CÁLCULO DE VOLUME EFETIVO DE OBJETOS EM MOVIMENTO USANDO ESTEREOSCOPIA 30
2.4.2 PROTÓTIPO DE UM AMBIENTE DE VISUALIZAÇÃO COM TÉCNICAS DE ESTEREOSCOPIA 31
2.4.3 OPENSTEREO: UMA BIBLIOTECA PARA SUPORTE AO DESENVOLVIMENTO DE APLICAÇÕES ESTEREOSCÓPICAS 31
3 DESENVOLVIMENTO DA BIBLIOTECA E FERRAMENTA 34
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO 34
3.2 ESPECIFICAÇÃO 34

3.2.1 Diagramas de casos de uso 34
3.2.2 Diagrama de classes 37
3.2.2.1 Classe StereoTools 38
3.2.2.2 Classe StereoVideo 38
3.2.2.3 Classe `Filter` 39
3.2.2.4 Classe `StereoFrame` 39
3.2.3 Diagrama de sequência 39
3.3 IMPLEMENTAÇÃO 40
3.3.1 Técnicas e ferramentas utilizadas 40
3.3.2 Preparar o ambiente 41
3.3.3 Operacionalidade da implementação 43
3.3.3.1 Estrutura da biblioteca 43
3.3.3.2 Estrutura da ferramenta 47
3.4 RESULTADOS E DISCUSSÃO 52
4 CONCLUSÕES 54
4.1 EXTENSÕES 54
REFERÊNCIAS 55

1 INTRODUÇÃO

A visão faz parte do sistema nervoso central e é ela que permite que organismos processem detalhes visuais (VISUAL, 2013). A visão realiza diversas tarefas, desde a recepção da luz e a formação de imagens monoculares, até a construção de uma visão binocular através de um par de duas projeções dimensionais.

Para facilitar o entendimento das limitações da visão monocular basta fechar um dos olhos e tentar realizar tarefas do cotidiano. O simples gesto de pegar um copo sobre a mesa passará a ser um desafio com apenas um dos olhos abertos. A dificuldade mais evidente, neste caso, é a de perceber a profundidade e avaliar a distância que separa o objeto do observador.

A visão monocular conta com elementos para uma percepção rudimentar da profundidade, valendo-se apenas das leis da perspectiva, onde o tamanho aparente dos objetos diminui à medida que esses se afastam do observador. Assim, os objetos mais próximos acabam escondendo, atrás de si, os objetos mais distantes que se encontram sobre o mesmo eixo de perspectiva (MOMM, 2001).

A visão humana é binocular e adquire a informação a partir de dois olhos que recebem a luz de dois pontos de vista diferentes e com isso conseguem gerar uma imagem estereoscópica. Estereoscopia é um termo utilizado para definir a ilusão de profundidade, fazendo o uso da visão binocular (STEREOSCOPY, 2013).

Ao assistir uma televisão, nota-se que a imagem ali, independente da resolução, é projetada em duas dimensões. Portanto, ao convergir os olhos para que estes foquem na tela da televisão, a imagem é exibida naquela distância. Desta maneira, não existe percepção de profundidade, pois a imagem está sendo exibida em um plano.

Para se exibir uma imagem estereoscópica numa televisão, em vez de gravar apenas uma imagem, deve-se realizar a captura de duas e exibi-las individualmente para que o olho esquerdo veja apenas a imagem esquerda, e o olho direito apenas a direita. Desta maneira pode-se simular a sensação de profundidade com duas imagens planas.

O método mais comum para capturar uma imagem estereoscópica é usar duas câmeras com as lentes em paralelo e com uma certa distância horizontal entre elas e assim simular o comportamento/distância dos olhos humanos.

Esta tecnologia encontra-se presente no dia a dia cada vez mais, como filmes sendo gravados com auxílio de duas câmeras, televisões/projetores/monitores que exibem imagens estereoscópicas, celulares/câmeras que capturam imagens tridimensionais. Em razão disso deve-se facilitar a existência de técnicas e ferramentas para trabalhar com estas imagens. Ou

seja, algoritmos que antes funcionavam com apenas uma câmera podem agora ser implementados para funcionar com duas câmeras paralelas, recebendo então o dobro de informação. Desta maneira, a aquisição de informações do mundo real pode ser mais completa.

Visão computacional é uma área da computação cujo propósito é possibilitar a um computador entender um ambiente através das informações visuais disponíveis (SHIRAI, 1987, p. 1). Esta área estuda a aquisição e assimilação de informações extraídas de imagens, transformando-as em dados que podem ser mensurados.

Algumas das técnicas de visão computacional atualmente utilizadas são ainda muito precárias, pois alguns tipos de reconhecimentos não podem ser feitos a partir de uma simples imagem 2D [1], especialmente quando o objetivo é obter medidas de profundidade ou reconhecer imagens que devido à iluminação não controlada acabam perdendo a definição e não podem ser reconhecidas pelos algoritmos. Uma das maneiras de tentar minimizar este problema é com a aquisição de mais pontos de vista, ou seja, câmeras de ângulos diferentes. Quanto mais câmeras, maior a quantidade de informações que pode ser obtida. Por isso, a estereoscopia é uma grande aliada da área de visão computacional.

Diante do exposto, o objetivo deste trabalho é desenvolver uma biblioteca que vai fornecer diversos algoritmos já conhecidos na área de visão computacional também para imagens estereoscópicas. Desta forma, alguns dos algoritmos de visão computacional que podem se beneficiar com a adição de uma câmera são estudados e utilizados nesta biblioteca, que por sua vez está disponibilizada para outros desenvolvedores que pretendem trabalhar com imagens estereoscópicas.

[1] São gráficos representados em uma matriz (3D COMPUTER, 2013).

1.1 OBJETIVOS DO TRABALHO

Este trabalho tem como objetivo disponibilizar uma biblioteca para facilitar a implementação de técnicas utilizadas na visão computacional em imagens 3D estereoscópicas.

Os objetivos específicos do trabalho são:

a) desenvolver uma biblioteca que trabalha imagens estereoscópicas;
b) fazer uso de algoritmos do OpenCV na biblioteca;
c) desenvolver uma ferramenta que faz uso e demonstra o funcionamento da biblioteca desenvolvida;
d) disponibilizar a biblioteca e o seu código fonte para uso público;
e) disponibilizar a ferramenta e o seu código fonte para uso público com intuito de explicar o funcionamento da biblioteca.

2 FUNDAMENTAÇÃO TEÓRICA

A seção 2.1 explica o que é estereoscopia e quais suas vantagens e desvantagens. A seção 2.2 identifica o ciclo de vida do conteúdo 3D, isto é, todos os passos necessários desde a produção do conteúdo (2.2.1) até a exibição do mesmo (2.2.3). A seção 2.3 trata da visão computacional e quais são algoritmos frequentemente utilizados na área. Na seção 2.3.1 o foco é a biblioteca de visão computacional *Open source Computer Vision library* (OpenCV*).* Por fim, a seção 2.4 mostra alguns trabalhos correlatos sobre o tema.

2.1 ESTEREOSCOPIA

O termo estereoscopia deriva das palavras gregas *stereos* e *skopein*, que significam respectivamente sólido/relevo e olhar/ver, que quer dizer, visão em relevo. A interpretação frequente de *stereo* no sentido de dois resulta do fato do observador necessitar de dois olhos para observar este tipo de imagem (KOHLER, 2001, p. 27).

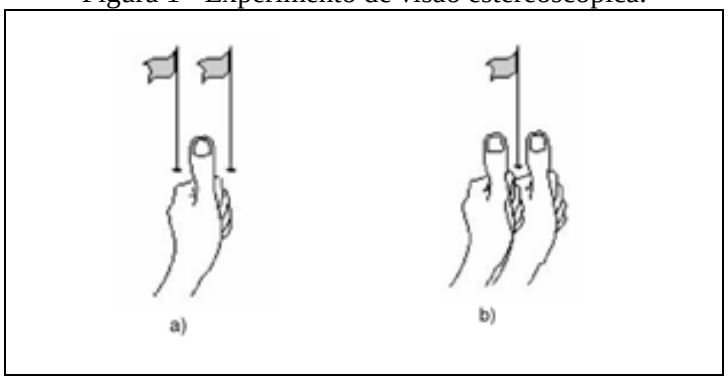
No decorrer da evolução, alguns animais (incluindo o homem) perderam o campo visual de 360 graus em favor de uma visão direcionada, com os olhos postados à frente da cabeça. A percepção de profundidade, nessa visão direcionada, ocorre quando se apresenta uma imagem para cada olho. Essas imagens devem possuir o mesmo foco, porém com um pequeno deslocamento do ponto de visão; este deslocamento é o responsável pela geração de pequenas diferenças entre as imagens, quase imperceptíveis quando observadas separadamente. Essas pequenas diferenças possibilitam a percepção 3D do ambiente no cérebro (LEITE, 2006).

Na natureza, há muitos animais que, à semelhança do homem, possuem um par de olhos localizados na parte frontal da cabeça. Esta característica é típica dos caçadores, como leões, gatos, gaviões, corujas, entre outros. Isto é importante para permitir a sobrevivência da espécie, pois tais animais precisam de uma correta percepção da distância até a sua presa. Por outro lado, animais que tipicamente se tornam presas dos caçadores, como coelhos, gazelas ou roedores, têm olhos posicionados lateralmente, o que não lhes dá uma boa percepção de distância, mas permite uma observação contínua do que há em sua volta para, ao menor sinal de perigo, poder escapar (LEITE, 2006).

Os olhos humanos capturam duas imagens ligeiramente diferentes, devido ao fato de estarem afastados por uma distância de aproximadamente 65 milímetros. Esta diferença entre as imagens é responsável por proporcionar uma visão binocular e consequentemente uma noção tridimensional do ambiente. Isso pode ser facilmente observado através de um experimento bastante simples: alinha-se o polegar esquerdo com um objeto à frente do nariz e

focaliza-se o olhar no dedo. Notam-se dois objetos no fundo, conforme ilustrado na Figura 1(a). Agora foque a visão no objeto por detrás do polegar: Notam-se dois dedos conforme ilustrado na Figura 1(b) (BARROS, 2009, p. 14).

Figura 1 - Experimento de visão estereoscópica.



Fonte: Stereographics (1997).

Segundo Barros (2009, p. 14), "A visão tridimensional que temos do mundo é resultado da interpretação, pelo cérebro, das duas imagens bidimensionais que cada olho capta a partir de seu ponto de vista e das informações sobre o grau de convergência e divergência".

Paiva et al, (2004) lembram que o funcionamento da percepção da profundidade foi descrito pela primeira vez por Charles Wheatstone em 1838. A partir daí não levou muito tempo para que outros inventores procurassem por novas maneiras de capturar imagens da mesma forma que olhos humanos.

Normalmente, em computação, quando é gerada uma cena tridimensional em um ambiente gráfico, gravam-se as imagens utilizando apenas uma câmera, o que é suficiente para exibição bidimensional (um monitor simples). Quando se deseja obter estereoscopia a partir de um ambiente gráfico tridimensional, é necessário renderizar duas imagens da cena utilizando duas câmeras virtuais, ou seja, renderizando duas imagens da mesma cena a partir de pontos de vista ligeiramente diferentes (GATEAU; NASH, 2010).

Uma das dificuldades de se trabalhar com estereoscopia em aplicações gráficas já aparece quando a aplicação pretende gerar imagens estereoscópicas. Várias precauções deverão ser tomadas na hora da criação das cenas, efeitos gráficos que funcionam perfeitamente quando vistos de um único ângulo podem não funcionar da maneira esperada quando vistos a partir de dois ângulos diferentes. Várias técnicas funcionais que são usadas para gerar uma imagem bidimensional a partir de um cenário gráfico tridimensional, como reflexão, profundidade de objetos, objetos renderizados apenas em uma câmera, efeitos renderizados em 2D e projeção de sombras, são só exemplos de técnicas que podem funcionar perfeitamente em um ambiente 2D e podem causar problemas em cenas gráficas

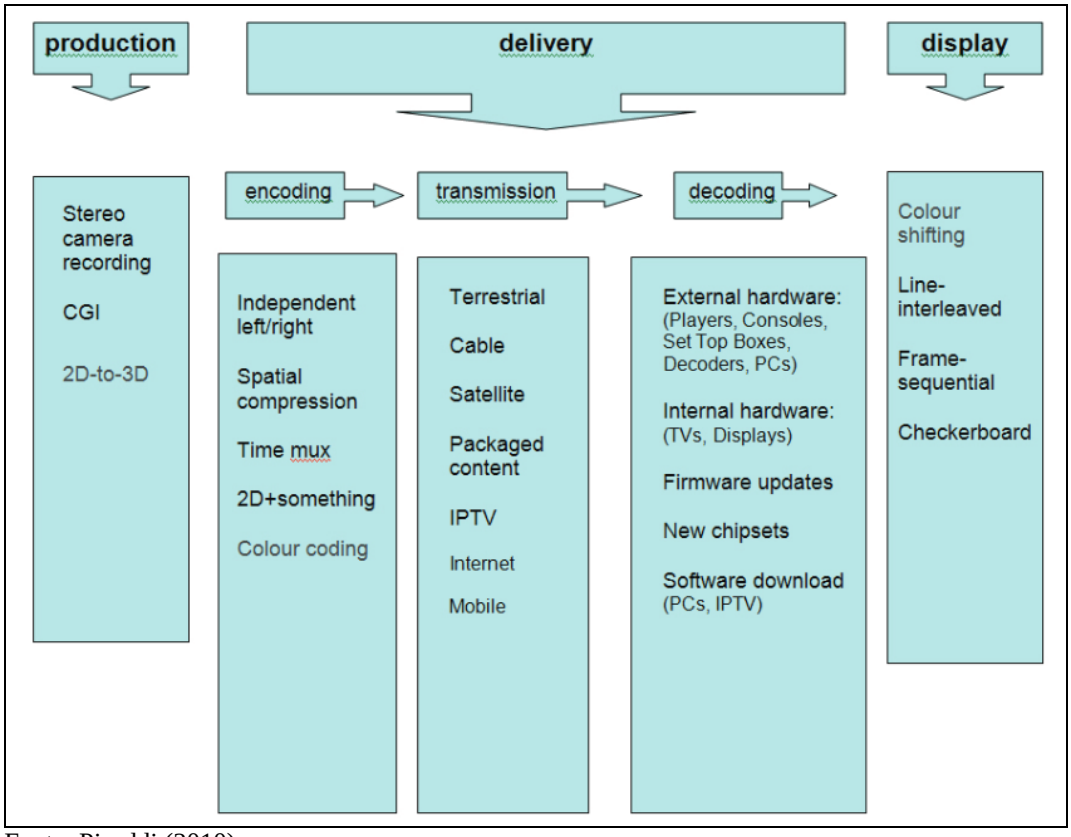
estereoscópicas (SCHNEIDER; NIKSHYCH, 2010). Outra dificuldade que é de grande impacto para a área gráfica é a questão do desempenho. Quando se deseja obter uma imagem estereoscópica de um ambiente gráfico 3D, é necessário pelo menos o dobro de processamento gráfico, pois para cada quadro que se obtém de um ambiente 3D, duas imagens devem ser renderizadas, ou seja, uma imagem com o dobro da resolução final para cada quadro.

Outra desvantagem em se trabalhar com imagens estereoscópicas é a necessidade de alguma técnica de visualização que permita a percepção óptica tridimensional para o observador. Essas técnicas são mencionadas na seção 2.2.3.

2.2 CICLO DE VIDA DO CONTEÚDO 3D

Para entender melhor o que acontece desde a geração do conteúdo 3D até a sua exibição final pode-se fazer uso do gráfico representado na Figura 2.

Figura 2 – Possíveis opções para o desenvolvimento de conteúdo 3D em cada estágio do seu ciclo de vida



Fonte: Piroddi (2010).

A Figura 2 sumariza os passos que podem ser tomados durante todo o ciclo de vida de um conteúdo 3D, ou seja, produção, entrega e exibição para o espectador. Usando como base essa figura, pode-se explicar com mais detalhes o que acontece em cada estágio deste ciclo.

2.2.1 Produção do conteúdo

As técnicas mais comumente utilizadas atualmente para a produção de imagens e vídeos 3D são (PIRODDI, 2010):

a) captura por câmeras;
b) imagens geradas pelo computador (c*omputer generated images* - CGI);
c) conversão de imagens 2D para 3D.

2.2.1.1 Captura por câmeras

A captura por câmeras acontece quando é gravada uma sequência de imagens estéreo [2] simultaneamente. Hoje já existem câmeras específicas para realizar este tipo de gravação. Além disso, é possível obter informações de profundidade usando câmeras *Time-Of-Flight* (TOF), *rangefinder* ou ainda câmeras adicionais em posições específicas. Mapas de profundidade também podem ser calculados a partir da informação obtida por duas ou mais câmeras (PIRODDI, 2010). Enquanto todas as técnicas apresentam suas dificuldades, o ponto principal aqui é que hardware adicional e/ou processamento são necessários para gerar a informação de profundidade e realizar a sua integração com a imagem espacial convencional (PIRODDI, 2010).

2.2.1.2 Imagens geradas por computador (CGI)

Uma imagem estéreo pode ser criada a partir de um ambiente 3D simulado no computador. O princípio é o mesmo, utilizar duas câmeras no ambiente virtual para obter uma imagem contendo dois pontos de vista diferentes de uma mesma cena.

O conteúdo estéreo gerado pelo computador é tipicamente considerado o método mais simples para gerar imagens estéreo. Neste caso, toda a informação sobre profundidade, oclusão e transparência é conhecida e está prontamente disponível sob a forma de um denso mapa que pode ser facilmente integrado com a imagem digital (PIRODDI, 2010).

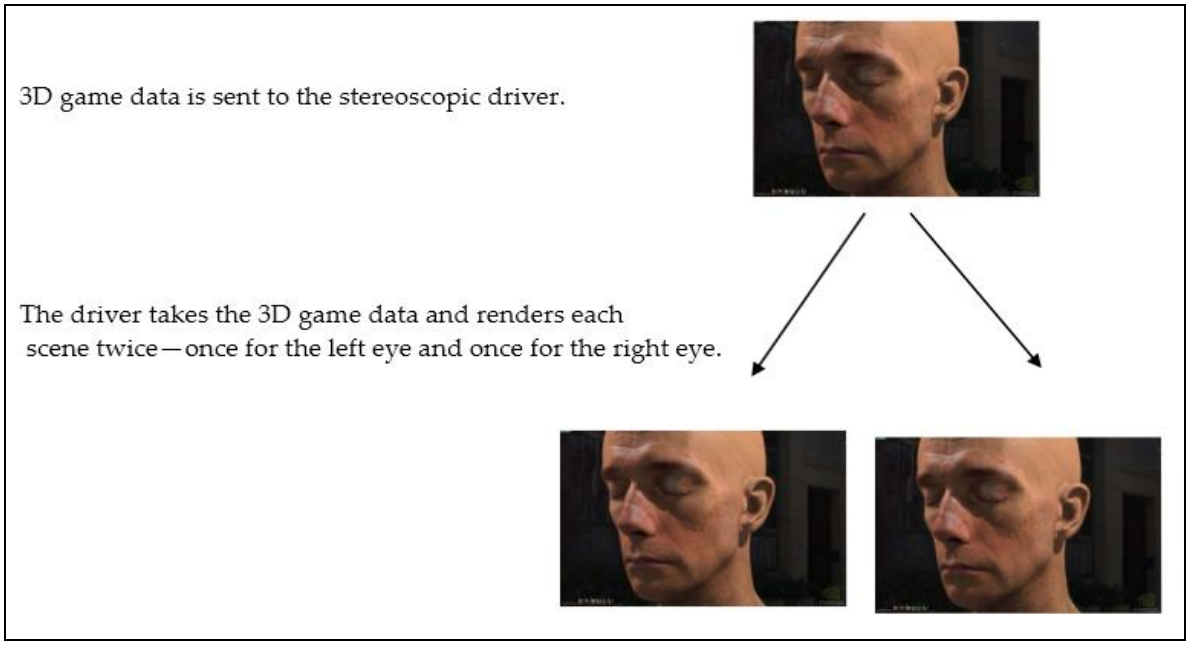
Aplicações gráficas 3D já possuem as informações necessárias para renderizar uma cena 3D de qualquer ângulo, ou seja, teoricamente, o esforço adicional para gerar uma imagem estéreo de um ambiente 3D seria apenas adicionar uma nova câmera no ambiente virtual seguindo o mesmo princípio utilizado para a captura de uma imagem estéreo do mundo real.

---

[2] Uma imagem que possui informações de dois pontos de vista diferentes.

A empresa Nvidia desenvolveu um *driver* para suas placas gráficas que permite gerar uma imagem 3D estéreo de basicamente qualquer ambiente 3D, conforme ilustrado na Figura 3.

Figura 3 – Técnica usada pelo *driver* da Nvidia para obter uma imagem estereoscópica a partir de um ambiente 3D



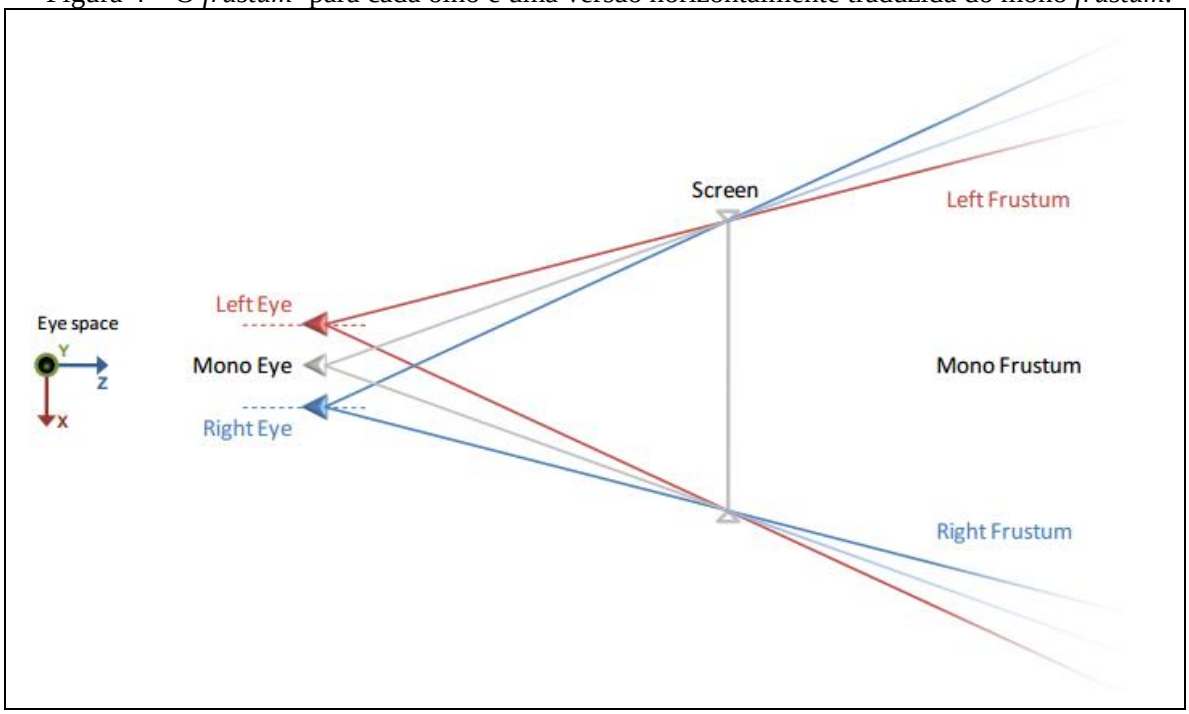
Fonte: Nvidia Corporation (2010).

Esta técnica funciona muito bem para vários jogos, mas a experiência final poderia ser melhorada se os desenvolvedores prestassem mais atenção em alguns itens como (NVIDIA corporation, 2010):

a) garantir que a interface do usuário seja exibida na profundidade da tela *mono eye* (Figura 4), pois desta maneira os itens de tela parecem estar na distancia natural da tela;
b) tomar cuidado para desenhar os objetos na profundidade correta, por exemplo, a *sky box* deve ser desenhada em uma profundidade válida, ou seja, neste caso a distância máxima da cena;
c) ter certeza de que todos os objetos 3D tem uma profundidade relativa válida com o resto da cena.

Escolher cuidadosamente a profundidade dos objetos na cena é importante, particularmente para aqueles que parecem sair da tela. Objetos que parecem sair da tela são desconfortáveis de se olhar, pois o cérebro do expectador deve superar o fato de que ele está vendo algo mais à frente de onde seus olhos estão focados. O objeto não deve sair das bordas da janela, pois ele será cortado. E o objeto deve mover-se lentamente de dentro da tela para fora, para dar tempo ao expectador de ajustar foco (NVIDIA corporation, 2010).

Figura 4 – O *frustum*[3] para cada olho é uma versão horizontalmente traduzida do mono *frustum*.



Fonte: Nvidia Corporation (2010).

A Figura 4 mostra que o ângulo de visão da tela a partir do ponto de vista de cada olho é relativamente diferente, onde a posição *mono eye* seria a visualização monodimensional da tela.

O que é muito interessante e vale a pena mencionar aqui sobre as imagens estéreo geradas a partir de ambientes 3D é que mesmo jogos e outros aplicativos antigos que foram desenvolvidos em um ambiente 3D e nunca se imaginou reproduzi-los estereoscopicamente, agora, com auxílio destes *drivers* da Nvidia, podem ser reproduzidos em 3D estéreo. Por este motivo, esta técnica de obtenção de imagens estéreo é a mais simples em comparação às outras duas mencionadas neste trabalho.

2.2.1.3 Conversão de imagens 2D para 3D

Uma imagem estereoscópica 3D pode ser gerada também a partir da interpretação da profundidade de uma imagem 2D.

Este processo requer a segmentação da imagem 2D. Para cada objeto é necessário calcular (pelo uso de pistas visuais 2D) e atribuir sua profundidade relativa. Também é necessário localizar áreas de oclusão e preenchê-las com porções adequadas de outros objetos (PIRODDI, 2010). Essa conversão pode ser feita em tempo real ou não. A conversão em tempo real é particularmente problemática pois a segmentação de imagens atualmente é um

[3] Parte de um cone que permanece após o corte de sua parte superior por um plano paralelo à sua base.

campo aberto às pesquisas na área de processamento de imagem digital. Este procedimento pode criar profundidade e mapas de oclusão que são adequados para a integração com imagens. No entanto, o resultado pode parecer “amassado” e por isso surrealista. Essa técnica é normalmente utilizada apenas para atualizar conteúdo legado 2D para 3D e não deve ser utilizada na criação de conteúdo novo (PIRODDI, 2010).

Muitas vezes essa técnica é aplicada para converter filmes que foram gravados em 2D para estéreo 3D, porém o processo não é automático e requer bastante trabalho pois é necessário definir e separar os objetos em cada cena do filme e muitas vezes adicionar novos objetos que não existiam antes, para poder gerar um efeito estéreo falso. Ou seja, diferentemente do processo anterior (2.2.1.2) que funciona até em aplicações legadas, cenas gravadas em 2D nunca poderão ser facilmente convertidas para 3D, pois uma imagem 2D não possui informação suficiente para essa conversão.

2.2.2 DISTRIBUIÇÃO DO CONTEÚDO

A distribuição explica todo o processo existente para armazenar, transmitir e enviar o conteúdo estéreo obtido.

2.2.2.1 Codificação do conteúdo

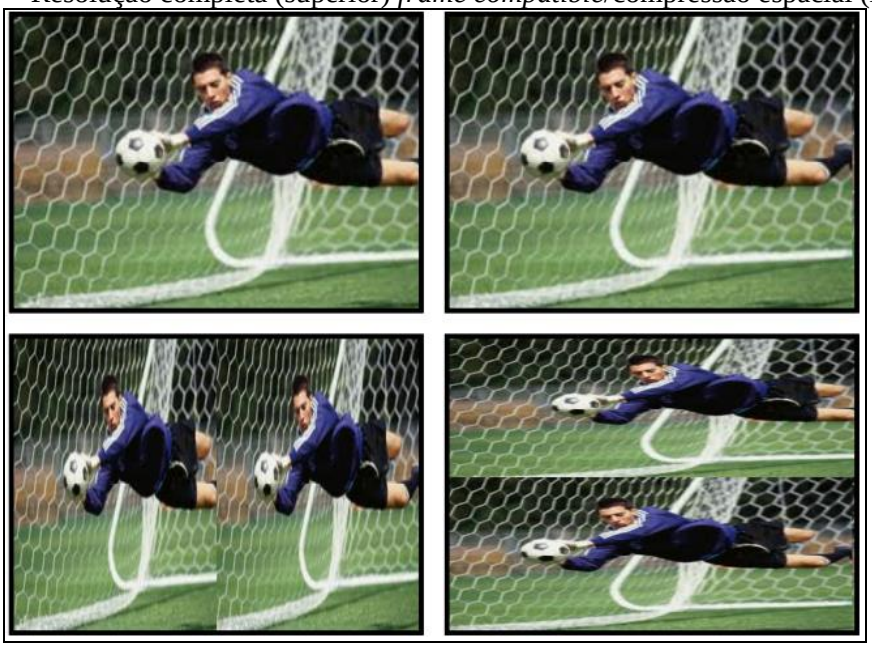
Para codificar o conteúdo é necessário garantir que as informações dos dois quadros, esquerdo e direito, poderão ser lidas separadamente depois, pois independente da técnica de exibição utilizada, o quadro esquerdo precisa ser enviado para o olho esquerdo e o quadro direito para o olho direito.

Existem 4 métodos para a codificação de conteúdo 3D, são eles (PIRODDI, 2010):

a) compressão espacial;
b) interlação temporal;
c) 2D mais alguma forma de metadados;
d) deslocação de cores.

A compressão espacial, também conhecida por *frame compatible*, nada mais é do que agregar as duas imagens em uma única. Dessa maneira é possível colocar dois quadros em um único, normalmente utilizando algum tipo de subamostragem dos pixels, por isso pode existir perda de resolução neste formato (PIRODDI, 2010). A Figura 5 mostra os dois resultados mais comuns obtidos após a codificação do conteúdo pela compressão espacial, são eles *Side-By-Side* (SBS) e *Over/Under* (OU) (PIRODDI, 2010).

Figura 5 – Resolução completa (superior) *frame compatible*/compressão espacial (inferior)



Fonte: Vetro, Tourapis, Muller e Chen (2011).

O método de interlação espacial exibe sequencialmente o quadro esquerdo e o direito, ambos com a resolução completa da imagem. Ou seja, é preciso dobrar a taxa de quadros por segundo e consequentemente o tamanho da banda necessária para enviar um sinal 2D HD (PIRODDI, 2010).

A ideia do 2D mais alguma forma de metadados é transmitir o sinal 2D e junto dele alguma outra informação que permita reconstruir aquela imagem 2D em 3D (PIRODDI, 2010). Na Figura 6 pode ser visto um exemplo desta codificação.

Figura 6 – Representação de uma figura 2D junto da informação de profundidade.

Fonte: Vetro, Tourapis, Muller e Chen (2011).

A maior desvantagem do formato 2D mais informação de profundidade é que ele é capaz de renderizar uma profundidade limitada. Além disso, não foi especificamente projetado para lidar com oclusões. Ainda, o sinal estéreo não é facilmente acessível por este

formato, pois é necessário converter este formato para duas imagens, o que não é a convenção dos *displays* atuais (VETRO; TOURAPIS; MULLER; CHEN, 2011).

Deslocação de cores é a técnica utilizada para a reprodução de imagens 3D anáglifo, que já é considerada uma tecnologia legada (PIRODDI, 2010). Mais informações sobre essa técnica poderão ser encontradas no item 2.2.3.1.

2.2.2.2 Transmissão do conteúdo

Depois do coteúdo codificado, a transmissão do conteúdo 3D ocorre da mesma maneira de qualquer outro conteúdo, pois são apenas informações de audio e video que serão transmitidas.

Uma coisa que deve ser notada é que, dependendo do método de codificação utilizado, o tamanho do conteúdo pode chegar a ser até o dobro do que este teria se fosse 2D. Isso ocorre pelo simples fato de conter o dobro de informação em cada quadro.

Atualmente algumas limitações podem existir dependendo da técnica de exibição utilizada. Essas limitações são discutidas na seção 2.2.3.

2.2.2.3 Decodificação do conteúdo

Existem diversas opções para realizar a decodificação do conteúdo, que podem variar de acordo com a codificação realizada. Por exemplo, se o quadro estéreo foi codificado fazendo uso da deslocação de cores (item d da seção 2.2.2.1), não poderá ser separado sem perder a informação das cores, por isso este formato só pode ser visto corretamente fazendo uso da técnica de anáglifos que é mencionada na seção 2.2.3.1.

A decodificação do conteúdo também pode variar dependendo da técnica de exibição que será selecionada, conforme consta na seção 2.2.3.

2.2.3 Técnicas de exibição

Cada hardware possui uma técnica específica para a reprodução das imagens estereoscópicas. O importante é saber que, independente da técnica que será utilizada para reproduzir a imagem final em 3D, a única entrada necessária é uma imagem estereoscópica, ou seja, duas imagens bidimensionais em uma. Desta maneira o efeito 3D poderá ser adquirido independente da técnica utilizada pelo projetor da imagem.

2.2.3.1 Anaglyph Glasses

Anaglyph Glasses (anáglifo) é o nome dado a figuras planas cujo relevo é obtido por cores complementares. Neste caso, cada um dos olhos utilizará um filtro diferente para visualizar as imagens do par estereoscópico: o olho que estiver com o filtro vermelho refletirá apenas a cor vermelha e o olho que estiver com o filtro verde/azul refletirá apenas a imagem verde/azul (MOMM, 2001, p. 19).

Dentre as vantagens da técnica anáglifo, destaca-se o baixo custo envolvido na utilização, pois o único dispositivo necessário para a visualização destas imagens são óculos anáglifos, como mostrado na Figura 7. Os óculos podem ser facilmente confeccionados e têm um baixo custo. Outra grande vantagem é que o material estereoscópico pode ser impresso, não restringindo a sua visualização apenas a telas e outros dispositivos eletrônicos (LEITE, 2006).

Esta técnica funciona tanto para imagens capturadas a partir de duas câmeras no mundo real como a partir de duas câmeras virtuais em um ambiente 3D. Na Figura 8 tem-se um exemplo de como é realizado o processo de junção das imagens.

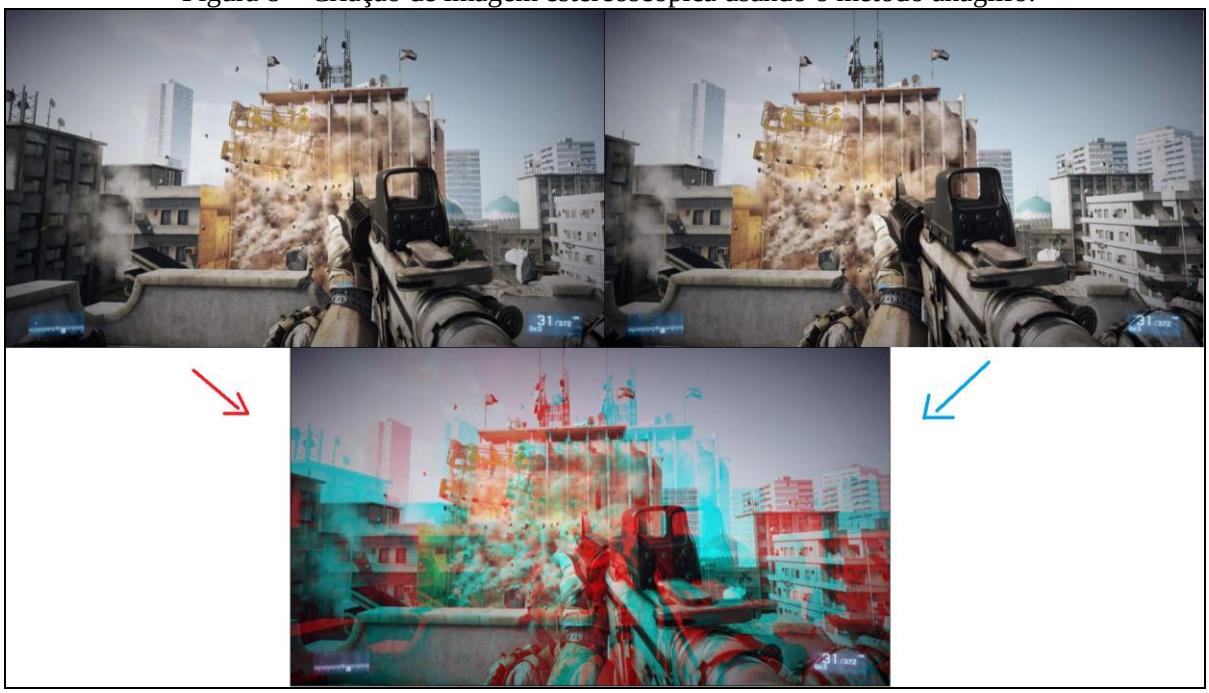
Existem diversos filtros que podem ser utilizados para gerar imagens anáglifas, mas todos eles apresentam uma desvantagem em comum que é a dificuldade de reproduzir as cores originais da imagem, além da presença de *crosstalk* ou *ghosting* (LEITE, 2006).

Crosstalk pode ser definido como o isolamento incompleto dos canais esquerdo e direito da imagem, de modo que ocorra vazamento da imagem de um canal para outro e *ghosting* nada mais é do que a percepção do efeito de *crosstalk* (WOODS, 2011).

Figura 7 – Óculos 3D para visualizar anáglifos.

Fonte: (LEITE, 2006).

Figura 8 – Criação de imagem estereoscópica usando o método anáglifo.



Fonte: Nvidia Gallery (2012).

2.2.3.2 Active shutter glasses

Active shutter glasses consiste de um par de óculos externo que fica nos olhos do espectador que possui 2 lentes de cristal líquido. É necessário um monitor capaz de sincronizar sua frequência com a dos óculos para que o processo ocorra normalmente. A imagem estereoscópica do lado esquerdo será exibida por alguns milissegundos em tela e logo após a imagem do lado direito será exibida também por alguns milissegundos.

Para o funcionamento do sistema deve haver um controle da seguinte forma: exibe-se na tela a imagem correspondente à do olho esquerdo e bloqueia-se a visão do olho direito. A seguir faz-se o contrário, ou seja, exibe-se a imagem do olho direito e bloqueia-se a visão do esquerdo (GATEAU; NASH, 2010). O processo descrito pode ser visualizado na Figura 9.

Figura 9 – Técnica *active shutter glasses*

Fonte: Gateau e Nash (2010).

A resolução é mantida, a qualidade das cores é mantida, porém sacrifica-se a taxa de atualização do monitor para exibir o efeito corretamente. Isso quer dizer que a frequência de atualização do monitor é efetivamente dividida ao meio para cada olho. Um monitor rodando a 120hz na verdade está exibindo 60hz para cada olho (GATEAU; NASH, 2010, pg 5).

Atualmente existem algumas limitações na transmissão deste tipo de exibição pois algumas interfaces mais lentas (tais como HDMI e single-link DVI) não conseguem transmitir 120hz de informação e estão limitadas aos 60hz (30hz para cada olho) (AMD, 2011).

2.2.3.3 Polarized 3D glasses

Polarized 3D Glasses representa um processo de estereoscopia por polarização da luz. São utilizados filtros polarizadores, os quais fazem com que as imagens projetadas do par estereoscópico sejam polarizadas em planos ortogonais. Desta forma, o observador utiliza filtros polarizadores ortogonais correspondentes aos planos de projeção e vê com cada olho apenas uma das imagens projetadas. A fusão das imagens vistas em cada olho resultará a visão estereoscópica (MACHADO, 1997, p. 31).

Este processo também conhecido por 3D estéreo passivo se refere ao uso de óculos que não possuem nenhum componente ativo (não mudam de estado conforme o quadro). Estes utilizam apenas filtros de luz para garantir que cada olho receba a luz referente a imagem correta (AMD, 2011).

Existem três maneiras principais para exibir o efeito estéreo em óculos passivos. São elas (AMD, 2011):

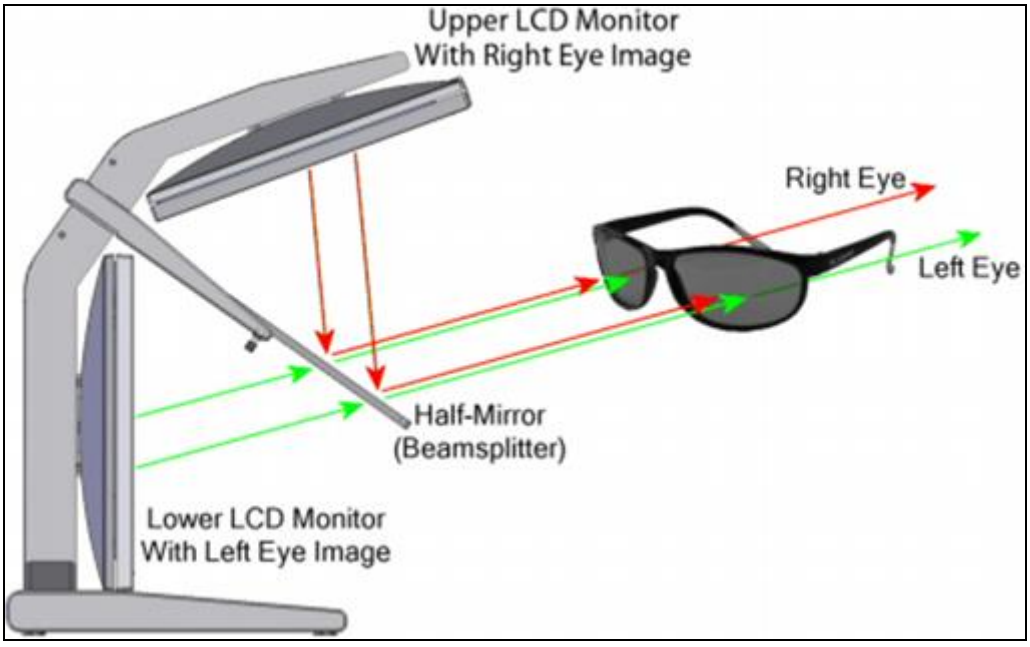
a) *dual displays*: fazer o uso de dois *displays* que apresentam imagens com polarização diferente. Eles normalmente são alinhados a um meio-espelho que permite que a luz de ambos os *displays* seja enviada simultaneamente para o olho

do espectador. Essa técnica permite a entrega de imagem com brilho total e resolução total para o espectador, conforme Figura 10;

b) *single display*: tipicamente, um filtro polarizador é adicionado na frente do *display* de vidro, cuidadosamente alinhado às linhas ou colunas de pixels (ou alguns em padrão xadrez). As imagens esquerda e direita são alinhadas de acordo com os pixels correspondentes. Essa técnica é atrativa devido ao seu baixo custo em comparação às outras técnicas, mas existe perda de brilho e resolução na imagem resultado;

c) *projected*: imagens polarizadas podem ser apresentadas por meio de um ou dois projetores e projetadas em uma tela que preserva a polarização. Essa é a técnica utilizada na maioria dos cinemas 3D atuais.

Figura 10 - Funcionamento da técnica *dual displays*



Fonte: Amd (2011).

A Figura 10 mostra que os raios de luz da tela superior refletem no espelho e a polarização das lentes do óculos faz com que a luz do monitor superior só seja visualizada pela lende direita, e a luz do monitor inferior só seja visualizada por trás da lente esquerda.

2.2.3.4 Head-mounted displays

A ideia dos *head-mounted displays* é exibir o conteúdo estéreo em duas pequenas telas (uma para cada olho) (MOMM, 2001, p. 07). Esta técnica é ilustrada na Figura 11.

Figura 11 – *Oculus Rift* (*Head-mounted display*), fabricado pela Oculus VR

Fonte: Oculusvr (2013).

O *oculus rift* cria uma imagem estereoscópica 3D com profundidade, escala e paralaxe[4]. Diferente de qualquer outro efeito 3D em uma televisão ou filme, aqui o efeito estereoscópico é realizado apresentando imagens únicas e paralelas para cada olho. Essa é a maneira que imagens do mundo real são apresentadas aos olhos, assim criando uma experiência muito mais natural e confortável (OCULUSVR, 2013).

A principal vantagem do uso de *head-mounted displays* em relação ao efeito de estereoscopia acontece na percepção do efeito de *ghosting* que é nulo, isso pelo fato de que cada imagem está sendo exibida em um *display* diferente, cada qual posicionado diretamente em cima do respectivo olho do usuário. Assim sendo não é possível ocorrer vazamento da imagem esquerda no olho direito e vice-versa.

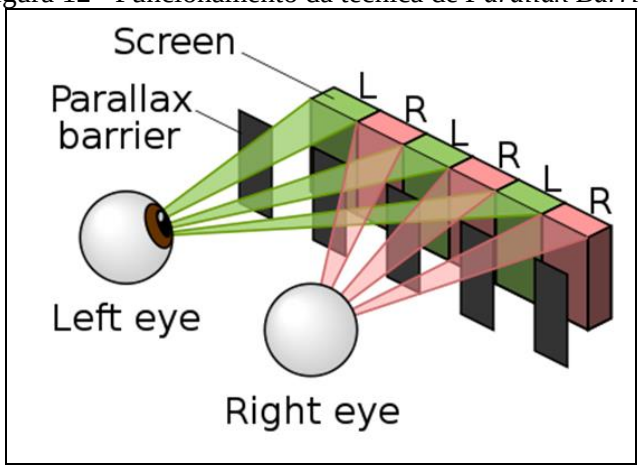
2.2.3.5 Autostereoscopy

Autostereoscopy é uma técnica que já é utilizada em alguns dispositivos móveis e em alguns televisores. O conceito é simples, bloquear a luz diretamente no *display*, para que a luz que saia do *display* seja direcionada diretamente para cada um dos olhos (3D WITHOUT GLASSES, 2012).

Existem diversas maneiras de se trabalhar com isso, mas a principal é usando uma barreira de paralaxe (Figura 12).

[4] Paralaxe é a distância aparente da posição de um objeto quando observado de dois pontos distanciados entre si.

Figura 12 - Funcionamento da técnica de *Parallax Barrier*



Fonte: 3D Without Glasses (2012).

Barreira de paralaxe é a técnica mais utilizada atualmente para geração do efeito de autoestereoscopia em telas de cristal líquido. A barreira de paralaxe é um dispositivo especial que consiste em uma série de micro ranhuras gravadas na tela e que atuam como um filtro para a percepção da imagem de saída. As ranhuras permitem que cada olho do expectador veja a imagem correspondente, cada qual produzida por um diferente conjunto de pixels. É assim que a ilusão de visão em 3D é criada com barreiras de paralaxe (3D WITHOUT GLASSES, 2012).

A desvantagem é que esta técnica requer que o usuário posicione sua cabeça em um ponto específico em relação ao *display* para que o efeito estéreo seja válido (AMD, 2011).

2.3 VISÃO COMPUTACIONAL

Visão computacional é um campo que estuda métodos para aquisição, processamento, análise e entendimento de imagens, realizando a aquisição de dados a partir de imagens do mundo real com o intuito de produzir informação numérica ou simbólica (COMPUTER VISION, 2013).

No entanto, o sonho de se ter um computador interpretando uma imagem no mesmo nível que uma criança de dois anos de idade (por exemplo, contar todos os animais existentes em uma imagem) continua sendo uma ilusão, pois interpretação de imagens é um problema em que se buscam algumas incógnitas em uma fonte em que as informações são insuficientes para especificar completamente a solução. Deve-se, portanto, recorrer a modelos baseados na física e probabilística para remover a ambiguidade entre as soluções possíveis. No entanto, a modelagem do mundo visual em toda a sua rica complexidade é muito mais difícil do que parece (SZELISKI, 2011, p. 3).

Diversas técnicas são utilizadas em imagens para realizar tarefas como reconhecimento, análise de movimento, reconstrução de cenários, restauração de imagens e outras (COMPUTER VISION,2013).

Algumas aplicações do mundo real que fazem uso da visão computacional são (SZELISKI, 2011, p. 5):

a) reconhecimento óptico de caracteres (*Optical Character Recognition* - OCR);
b) inspeção de máquinas, para inspeção da qualidade final de produtos;
c) reconhecimento de objetos;
d) construção de estruturas 3D a partir de fotografias aéreas;
e) segurança automotiva, detectando obstáculos inesperados na pista;
f) captura de movimentos, transformando movimentos humanos em tempo real para movimentos virtuais em ambientes gerados por computador;
g) vigilância, monitorando intrusos, analisando tráfego e monitorando piscinas por vítimas de afogamento;
h) reconhecimento biométrico, analisando características humanas para através delas gerar autenticação.

2.3.1 OPENCV

Conforme Bradski e Kaehler (2008, p. 1), a *Open source Computer Vision library* (OpenCV) é "uma biblioteca de código fonte aberto desenhada para eficiência computacional e com forte foco em aplicações de tempo real. A biblioteca foi escrita em C e suporta a utilização de múltiplos processadores".

OpenCV disponibiliza diversos algoritmos do campo de visão computacional para o desenvolvimento de aplicativos na área. Foi desenvolvida nas linguagens de programação C e C++, porém também dá suporte para outras linguagens de programação como Java, Pyton e Visual Basic. As plataformas compatíveis com a OpenCV são Microsoft Windows (95/98/NT/2000/XP,7), POSIX (Linux/BSD/UNIX-like OSes), Linux, OS X, MAC OS Básico (KOCH, 2012, p.46).

A biblioteca OpenCV é estruturada nos seguintes módulos (KOCH, 2012, p. 46):

a) `cv`: módulo das principais funcionalidades e algoritmos de visão computacional;
b) `cvaux`: módulo com algoritmos de visão, está em fase experimental;
c) `cxcore`: módulo de estrutura de dados e álgebra linear;
d) `highgui`: módulo de controle de interface e dispositivos de entrada;
e) `ml`: módulo de *machine learning* – aprendizado por máquina;

f) `ed`: manual de estrutura de dados e operações.

Na parte de processamento de imagens o OpenCV disponibiliza de diversos algoritmos que podem ser aplicados em imagens tanto por questões visuais como para extração de informações especificas de imagens, como bordas, cores e dimensões. Segue uma lista com os funções disponibilizados pelo OpenCV na sua versão atual (OPENCV, 2014):

a) *smoothing images*: suavizar imagens com filtros lineares;
b) *eroding and dilating*: alterar forma dos objetos;
c) *more morphology transformations*: investigar diferentes operadores morfológicos;
d) *image pyramids*: alterar tamanho das imagens;
e) *basic thresholding operations*: operações lineares;
f) *making your own linear filters!*: fazer seu filtro linear;
g) *adding borders to your images*: adicionar bordas laterais em imagens;
h) *sobel derivatives*: usar gradientes para detectar bordas;
i) *laplace operator*: detectar bordas com a função *laplace*;
j) *canny edge detector*: alternativa sofisticada para detecção de bordas;
k) *hough line transform*: detectar linhas;
l) *hough circle transform*: detectar círculos;
m) *remapping*: manipular a localização dos pixels;
n) *affine transformations*: rotacionar, transladar e escalar imagens;
o) *histogram equalization*: melhorar o contraste em imagens;
p) *histogram calculation*: como gerar e criar histogramas;
q) *histogram comparison*: calcular métricas entre histogramas;
r) *back projection*: usar histogramas para encontrar objetos similares;
s) *template matching*: comparar *templates* de imagens;
t) *finding contours in your image*: encontrar contornos em objetos na imagem;
u) *convex hull*: aplicar o algoritmo *convex hull* nas imagens;
v) *creating bounding boxes and circles for contour*: obter *bound boxes* de contornos;
w) *creating bounding rotated boxes and ellipses for contours*: obter *bound boxes* com rotação a partir de contornos;
x) *image moments*: calcular momento de imagen;
y) *point polygon test*: calcular distâncias a partir de contornos da imagem.

2.4 TRABALHOS CORRELATOS

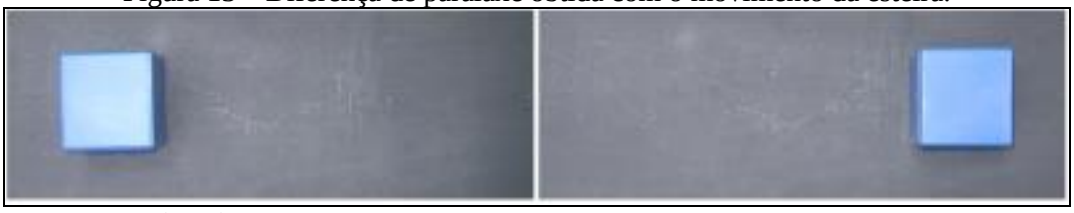
A seguir são mencionados alguns trabalhos já desenvolvidos que também fizeram o uso de técnicas de estereoscopia em busca de um aprimoramento na quantidade de características que podem ser extraídas de uma imagem.

2.4.1 CÁLCULO DE VOLUME EFETIVO DE OBJETOS EM MOVIMENTO USANDO ESTEREOSCOPIA

Neste trabalho se faz a aquisição de um par de imagens estéreo para estimar a altura de objetos em movimento. Tal par de imagens é obtido através de um sistema de esteira com suporte para uma câmera posicionada acima dela. A câmera é fixada a uma distância conhecida e precisamente medida sobre a esteira. Ao mover-se na esteira, o objeto é fotografado por essa câmera em posicões deslocadas, a fim de se adquirir um par de imagens estéreo do mesmo (BARROS, 2009, p.23).

O procedimento necessário para medição de altura de objetos neste trabalho se inicia com a aquisição do par de imagens estéreo, utilizando-se do mecanismo de esteira mencionado anteriormente. Barros convencionou fotografar o objeto a cada 15cm percorridos ao longo da esteira, com isso a paralaxe absoluta em cada par estéreo será de 30,45 ou 60cm, dependendo da escolha das imagens do par (BARROS, 2009, p.23). A diferença de paralaxe do objeto ao se mover na esteira pode ser verificada na figura 13.

Figura 13 – Diferença de paralaxe obtida com o movimento da esteira.



Fonte: Barros (2009).

Depois da aquisição das imagens, são utilizados utilizados alguns filtros para pós-processamento das imagens para adquirir então a região de interesse daquele objeto. Com a região de interesse definida, são utilizados dois algoritmos para calcular as disparidades de paralaxe e as dimensões do objeto, um para analisar a imagem no sentido vertical e outro para analisá-la no sentido horizontal (BARROS, 2009, p.23).

Foi necessário calibrar o sistema para encontrar o valor em pixels da paralaxe absoluta. Depois disso a altura pode ser calculada e o calculo do volume do objeto final é realizado com a aquisição de uma terceira imagem, ortogonal ao eixo da câmera, para então verificar a área do objeto (BARROS, 2009, p.23).

Os resultados obtidos neste trabalho variaram de acordo com fatores como: iluminação da cena, textura do objeto e altura da câmera em relação à esteira. A dificuldade de se controlar esses parâmetros refletiu na exatidão da estimativa da altura e volume do objeto. Entretanto, os resultados foram satisfatórios, considerando a inexistência de condições precisamente controladas para a aquisição de imagens (BARROS, 2009, p.23).

2.4.2 PROTÓTIPO DE UM AMBIENTE DE VISUALIZAÇÃO COM TÉCNICAS DE ESTEREOSCOPIA

Este trabalho apresenta o desenvolvimento de um ambiente de visualização, no ambiente de programação Delphi, aplicando a técnica de fotografia estereoscópica, possibilitando a percepção do efeito estereoscópico (MOMM, 2001).

O protótipo foi desenvolvido com o intuito de gerar uma imagem estereoscópica com técnicas de luz intermitente ou anáglifo, sobrepondo duas imagens onde cada uma fica com tons de azul e vermelho. O resultado obtido é uma imagem que pode ser vista por um par de óculos anáglifo que será utilizado para filtrar a entrada de luz para cada olho.

O autor trabalha com duas técnicas para exibição da imagem, são elas a técnica anáglifo e a técnica de luz intermitente (mencionadas no item d) da seção 2.2.2.1).

É interessante mencionar que neste trabalho é utilizada apenas uma imagem 2D, que por sua vez, é convertida em estéreo por um valor de paralaxe escolhido pelo usuário, no caso é a distância do observador à tela para evitar *ghosting* (mencionado na seção 2.2.3.1). Portanto o efeito estéreo se encaixa no efeito mencionado no item 2.2.1.3 (Conversão de imagens 2D para 3D).

A obtenção do efeito estéreo através da técnica de luz intermitente é mais demorada do que a técnica anáglifo, pois cada vez que a paralaxe é alterada, e deseja-se que a nova imagem estéreo seja baseada neste valor, a imagem tem que ser novamente inicializada para que a paralaxe possa ser aplicada e o efeito estéreo possa ser obtido (MOMM, 2001). Esta diferença se dá por causa do acerto da janela estéreo que é diferente para as duas técnicas (MOMM, 2001). Como resultados obtidos também é mencionado o próprio protótipo que converte imagens 2D em 3D anáglifo.

2.4.3 OPENSTEREO: UMA BIBLIOTECA PARA SUPORTE AO DESENVOLVIMENTO DE APLICAÇÕES ESTEREOSCÓPICAS

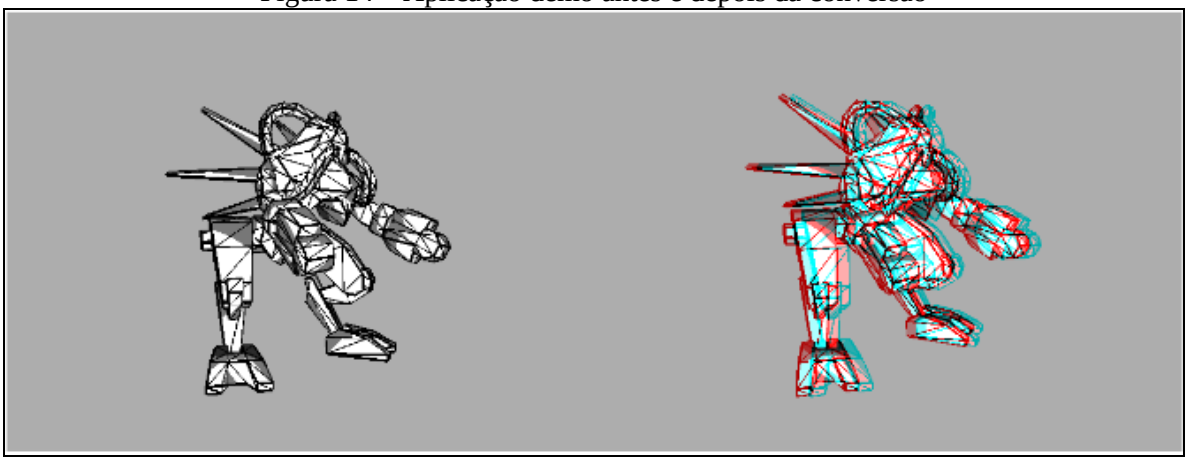
Este trabalho apresenta a biblioteca OpenStereo, cujo objetivo principal é fornecer suporte à conversão de aplicações 3D de tempo real (baseadas na biblioteca *Open Graphics Library* (OpenGL)) em aplicações estereoscópicas (LEITE, 2006).

Este trabalho desenvolveu uma ferramenta que através de um subsistema de *plugins* é capaz de transformar uma aplicação de OpenGL numa aplicação estereoscópica. O usuário escolhe a cena a ser renderizada e a partir deste ponto o OpenStereo gera os pares de imagem esquerda/direita, cria a imagem estéreo e finalmente renderiza a cena em um *viewport*.

Para a conversão, o método escolhido foi o estéreo anáglifo, uma vez que os custos envolvidos em equipamentos para a visualização da aplicaçao são baixos. Isso viabiliza tanto o seu desenvolvimento quanto a sua execução (LEITE, 2006).

A Figura 14 representa o resultado da conversão da aplicação demo. No lado esquerdo pode-se observar o resultado antes da conversão, enquanto que no lado direito, o resultado pós conversão.

Figura 14 – Aplicação demo antes e depois da conversão

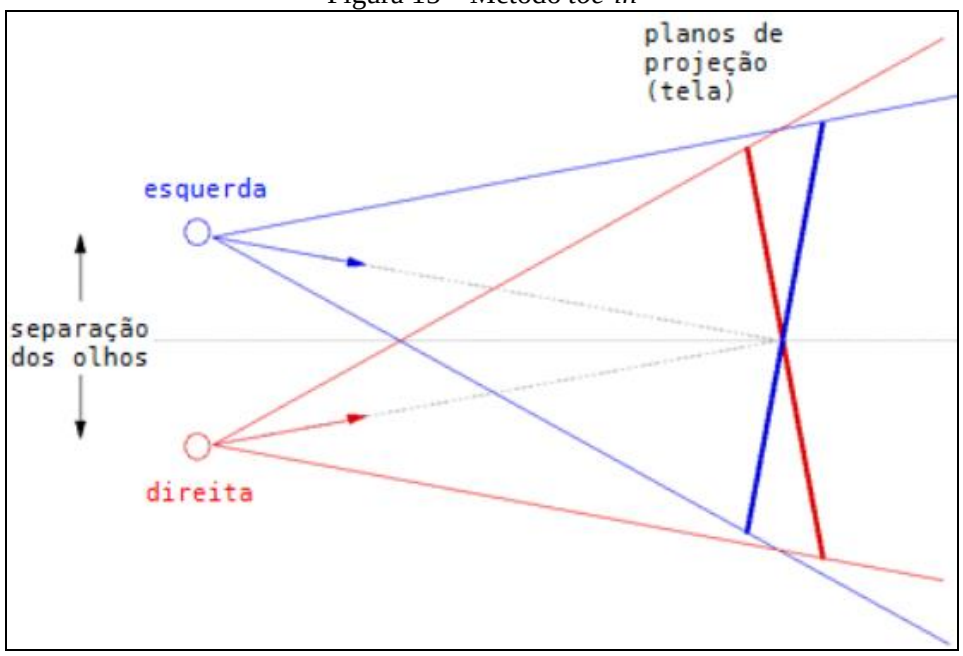


Fonte: Leite (2006).

Para obtenção do efeito estéreo a partir do ambiente virtual foi utilizado o método *toe-in* (Figura 15). Apesar desse método apresentar um resultado satisfatório na parte visual, ele é incorreto, pois introduz uma paralaxe vertical na imagem estéreo. O metodo correto seria o *ff-axis* (Figura 16).

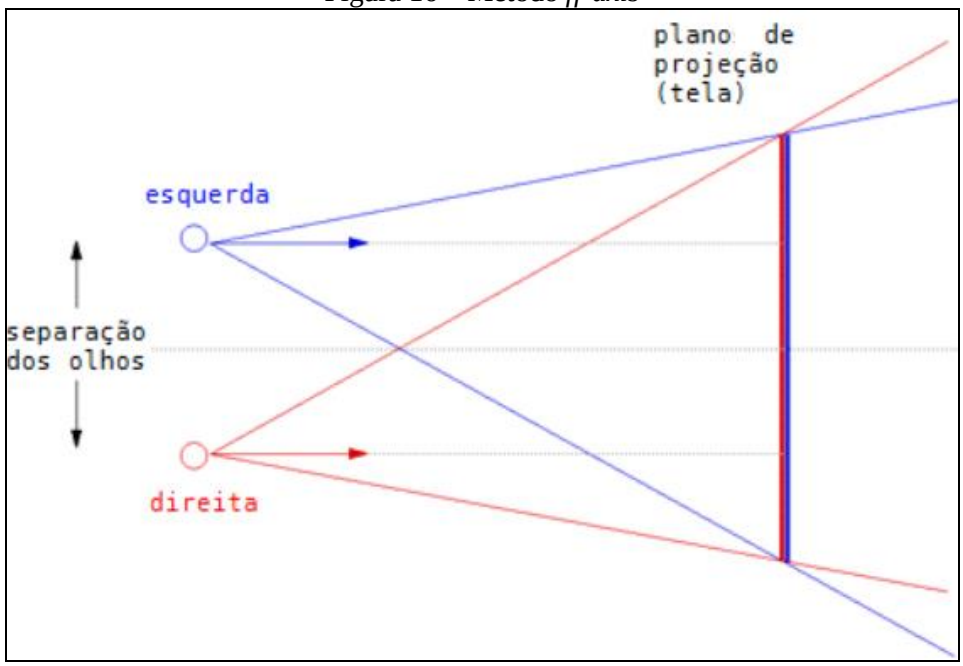
A arquitetura e implementação do OpenStereo foram detalhadamente explicadas, facilitando a reprodução desse trabalho como também a sua extensibilidade.Os resultados foram considerados satisfatórios, os testes realizados contaram o não comprometimento do desempenho da aplicação.

Figura 15 – Método *toe-in*



Fonte: Leite (2006).

Figura 16 – Método *ff-axis*



Fonte: Leite (2006).

3 DESENVOLVIMENTO DA BIBLIOTECA E FERRAMENTA

Neste capítulo são tratados assuntos pertinentes à implementação da biblioteca e da ferramenta. Aqui encontram-se requisitos principais, especificação, implementação, técnicas e ferramentas utilizadas, operacionalidade da implementação e instruções para a utilização da biblioteca. Encerrando o capítulo, encontram-se os resultados e discussões.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os requisitos identificados para o desenvolvimento deste trabalho são:

a) disponibilizar uma biblioteca que permite a aplicação de algoritmos de visão computacional em imagens estereoscópicas (Requisito Funcional - RF);
b) a biblioteca deve permitir o carregamento de vídeos estéreo (RF);
c) a biblioteca deve permitir a aplicação de filtros do OpenCV nos vídeos carregados (RF);
d) a biblioteca deve permitir salvar os vídeos alterados em arquivos compatíveis com dispositivos de exibição estéreo (RF);
e) disponibilizar uma ferramenta que faz uso e demonstra o funcionamento da biblioteca que foi desenvolvida (RF);
f) utilizar algoritmos da biblioteca *Open source computer vision library* (OpenCV) e aplicá-los em vídeos 3D (RF);
g) ser implementado para a plataforma Windows (Requisito Não Funcional – RNF).
h) ser desenvolvido no ambiente Microsoft Visual Studio (RNF);
i) utilizar a ferramenta Astah Community para especificar o trabalho (RNF).

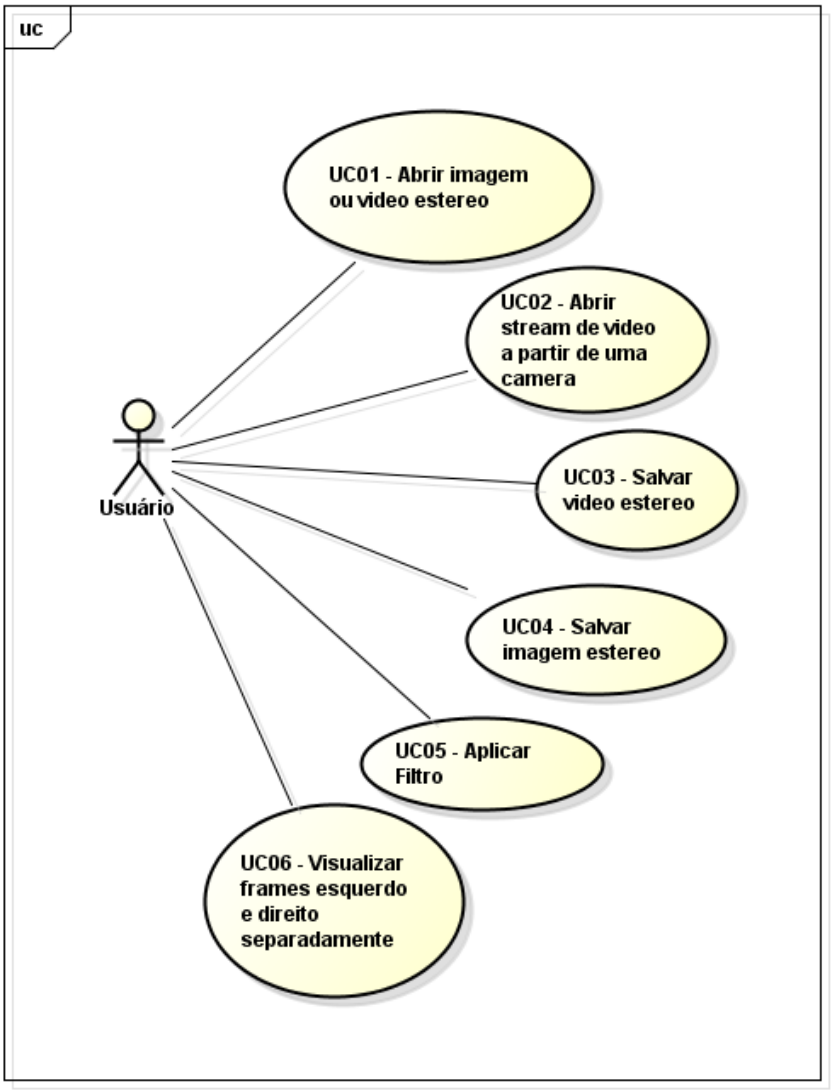
3.2 ESPECIFICAÇÃO

Para a especificação do projeto, foram utilizados diagramas da *Unified Modeling Language* (UML), sendo eles os diagramas de caso de uso, classes e sequência. A ferramenta Astah Community foi utilizada para modelar e gerar os diagramas.

3.2.1 Diagramas de casos de uso

Nesta seção encontram-se os casos de uso referentes à ferramenta, lembrando que a ferramenta foi desenvolvida com o intuito de testar as funcionalidades presentes na biblioteca. Também são apresentadas descrições detalhadas dos casos de uso UC01 a UC06 presentes no diagrama da Figura 17.

Figura 17 - Diagrama de casos de uso da ferramenta



O caso de uso UC01 descreve a funcionalidade que permite ao usuário abrir uma imagem estereoscópica a partir de um arquivo (detalhes no Quadro 1). O caso de uso UC02 descreve a funcionalidade que permite carregar um fluxo de imagens a partir de uma ou mais câmeras conectadas (

Quadro 2). O caso de uso UC03 descreve a criação e armazenamento de vídeos estereoscópicos pelo usuário (detalhes no

Quadro 3). O caso de uso UC04 descreve a criação e armazenamento de imagens pelo usuário. A descrição completa está presente no

Quadro 4.

O caso de uso UC05 descreve a aplicação de filtros nas imagens (

Quadro 5). O caso de uso UC06 descreve a exibição das imagens separadas para o usuário (detalhes no

Quadro 6).

Quadro 1 – Caso de uso UC01 em detalhes

UC01 – Abrir imagem ou vídeo estéreo.	
Descrição	Permite ao usuário carregar imagens e ou vídeos 3D estéreo do computador
Pré-Condição	O formato da imagem/vídeo deve ser *Side-By-Side* ou *Over/Under*.
Cenário Principal	1. O usuário obtém o caminho do arquivo que contem a imagem/video `char *fileName;` 2. O usuário passa o nome do arquivo para a função `openVideoFromFile` da biblioteca contida na classe estática `StereoVideo`; 3. O usuário chama a função `getNextFrame()` da classe `StereoVideo` para capturar o próximo quadro (se existe); 4. O quadro pode ser obtido do objeto `currentFrame` da classe `StereoVideo`.
Pós-condição	O usuário passa a ter um quadro estéreo. Este poderá ser utilizado em outras funções da biblioteca.

Quadro 2 – Caso de uso UC02 em detalhes

UC02 – Abrir *stream* de vídeo a partir de uma câmera.	
Descrição	Permite ao usuário abrir um fluxo de imagens a partir de uma ou mais câmeras conectadas ao computador.
Pré-Condição	Deve existir uma câmera compatível conectada ao computador.
Cenário Principal	1. O usuário chama a função `openVideoFromCamera()` da biblioteca contida na classe estática `StereoVideo`; 2. O usuário chama a função `getNextFrame()` da classe `StereoVideo` para capturar o quadro atual da câmera; 3. O último quadro obtido está dentro da variável `currentFrame` da classe `StereoVideo`.
Pós-condição	O usuário passa a ter um quadro diretamente da câmera.

Quadro 3 – Caso de uso UC03 em detalhes

UC03 – Salvar vídeo estéreo.	
Descrição	Permite a criação de um arquivo de vídeo estéreo pelo usuário. Que será armazenado em disco.
Pré-Condição	É necessário efetuar o carregamento de um vídeo antes da geração de outro.
Cenário Principal	1. O usuário obtém o nome e caminho do local onde será armazenado o vídeo na variável `fileName;` 2. O usuário passa o caminho obtido em `fileName` para a função `saveVideo` da biblioteca contida na classe estática `StereoVideo`.
Exceção	Na etapa 2, a extensão do arquivo deve ser compatível com o codec selecionado, caso contrário o arquivo gerado será inválido.
Pós-condição	Um novo arquivo será criado contendo o vídeo estéreo gerado pelo usuário.

Quadro 4 – Caso de uso UC04 em detalhes

UC04 – Salvar imagem estéreo.	
Descrição	Permite a criação de um arquivo de imagem estéreo pelo usuário. Que será armazenado em disco.
Pré-Condição	É necessário que algum quadro tenha sido carregado pela biblioteca.

Cenário Principal	1. O usuário obtém o nome e caminho onde será armazenada a imagem na variável `fileName`; 2. O usuário passa o caminho obtido em `fileName` para a função `saveImage` da biblioteca contida na classe estática `StereoVideo`.
Exceção	Na etapa 2, a extensão do arquivo deve ser compatível com o OpenCV, caso contrário, uma exceção será lançada.
Pós-condição	Um novo arquivo será criado contendo a imagem gerada pelo usuário.

Quadro 5 – Caso de uso UC05 em detalhes

UC05 – Aplicar filtro.	
Descrição	Permite ao usuário aplicar filtros nas imagens carregadas pela biblioteca.
Pré-Condição	Deve existir alguma imagem previamente carregada na biblioteca para que o filtro possa ser aplicado.
Cenário Principal	1. O usuário chama a função `StereoTools::splitFrames` que vai quebrar o quadro em dois quadros; 2. Obter o resultado da função `applySelectedFilter` ou `applyFilter` da classe `StereoTools` passando como parâmetro para ela os quadros separados na etapa 1; 3. Para obter o quadro inteiro novamente, o usuário pode chamar a função `joinFrames` da classe `StereoTools` que vai gerar uma única imagem com os 2 quadros obtidos na etapa 2.
Pós-condição	O resultado obtido pela etapa 3 é um frame estéreo com o filtro selecionado na etapa 2 aplicado.

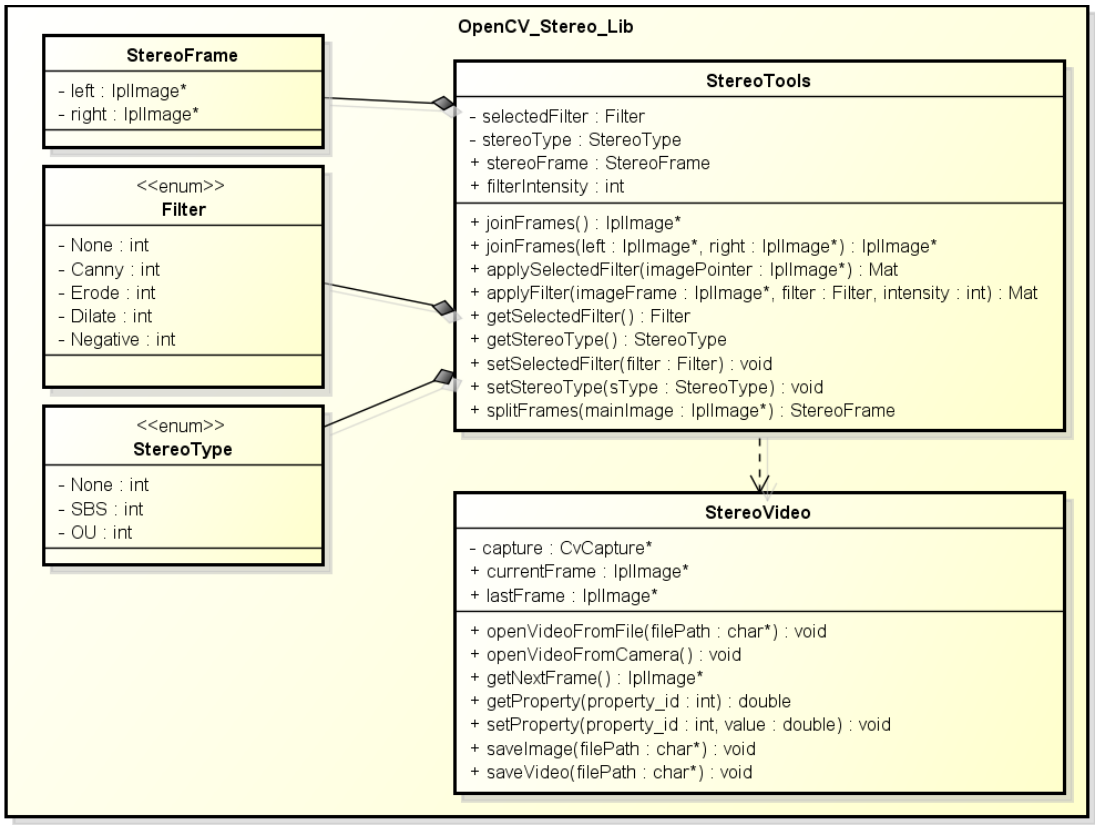
Quadro 6 – Caso de uso UC06 em detalhes

UC06 – Visualizar *frames* esquerdo e direito separadamente.	
Descrição	Permite ao usuário exibir o quadro esquerdo e direito separadamente de uma imagem estereoscópica.
Pré-Condição	Deve existir algum quadro carregado na biblioteca.
Cenário Principal	1. O usuário chama a função `StereoTools::splitFrames` que vai quebrar o quadro em dois quadros sendo estes esquerdo e direito; 2. Pegar os quadros resultado da etapa 1 e jogar esses em uma `PictureBox` para visualização em tela.
Pós-condição	O usuário pode visualizar os quadros esquerdo e direito em duas `PictureBox`.

3.2.2 Diagrama de classes

Especificações das classes da biblioteca juntamente com suas relações estão definidas no diagrama de classes ilustrado na Figura 18.

Figura 18 - Diagrama de classes



3.2.2.1 Classe StereoTools

A classe `StereoTools` é a classe que contém as ferramentas da biblioteca. Esta classe é usada para realizar as diversas tarefas como transformar um vídeo estéreo em dois vídeos 2D a partir da função `splitFrames` e também juntar dois quadros 2D em um único quadro estéreo através da função `joinFrames`. Também é possível aplicar filtros nos quadros fazendo uso da função `applyFilter` ou `applySelectedFilter`.

Essa classe também é responsável em armazenar algumas informações como `selectedFilter` que contém a informação do último filtro selecionado, juntamente com sua intensidade `filterIntensity`. O objeto `stereoFrame` da classe armazena o último quadro estéreo capturado. Está classe também possui o enumerador `StereoType` que armazena a informação referente ao tipo de método de códificação e decodificação do conteudo estéreo.

3.2.2.2 Classe StereoVideo

A classe `StereoVideo` armazena informações referentes ao arquivo de vídeo ou imagem carregado como o quadro atual dentro de `currentFrame` e o último quadro capturado em `lastFrame`. Essa classe também traz funções como `openVideoFromFile` para iniciar a captura de imagens a partir de um arquivo e `openVideoFromCamera` para iniciar um *stream* de

imagens a partir de uma câmera. Quando existe algum imagem para ser capturada a função `getNextFrame` traz o próximo quadro do objeto `capture`.

Por fim, tem-se à disposição os métodos `saveImage` e `saveVideo` que permitem gerar um arquivo contendo a imagem modificada pela biblioteca.

3.2.2.3 Classe `Filter`

A classe `Filter` é apenas um enumerador utilizado para guardar o identificador dos filtros do programa. Para adicionar novos filtros é necessário adicioná-los a esta classe.

3.2.2.4 Classe StereoFrame

A classe `StereoFrame` contém os atributos que definem as propriedades de um quadro estéreo, que são basicamente as informações do quadro esquerdo `left` e do quadro direito `right`.

3.2.3 Diagrama de sequência

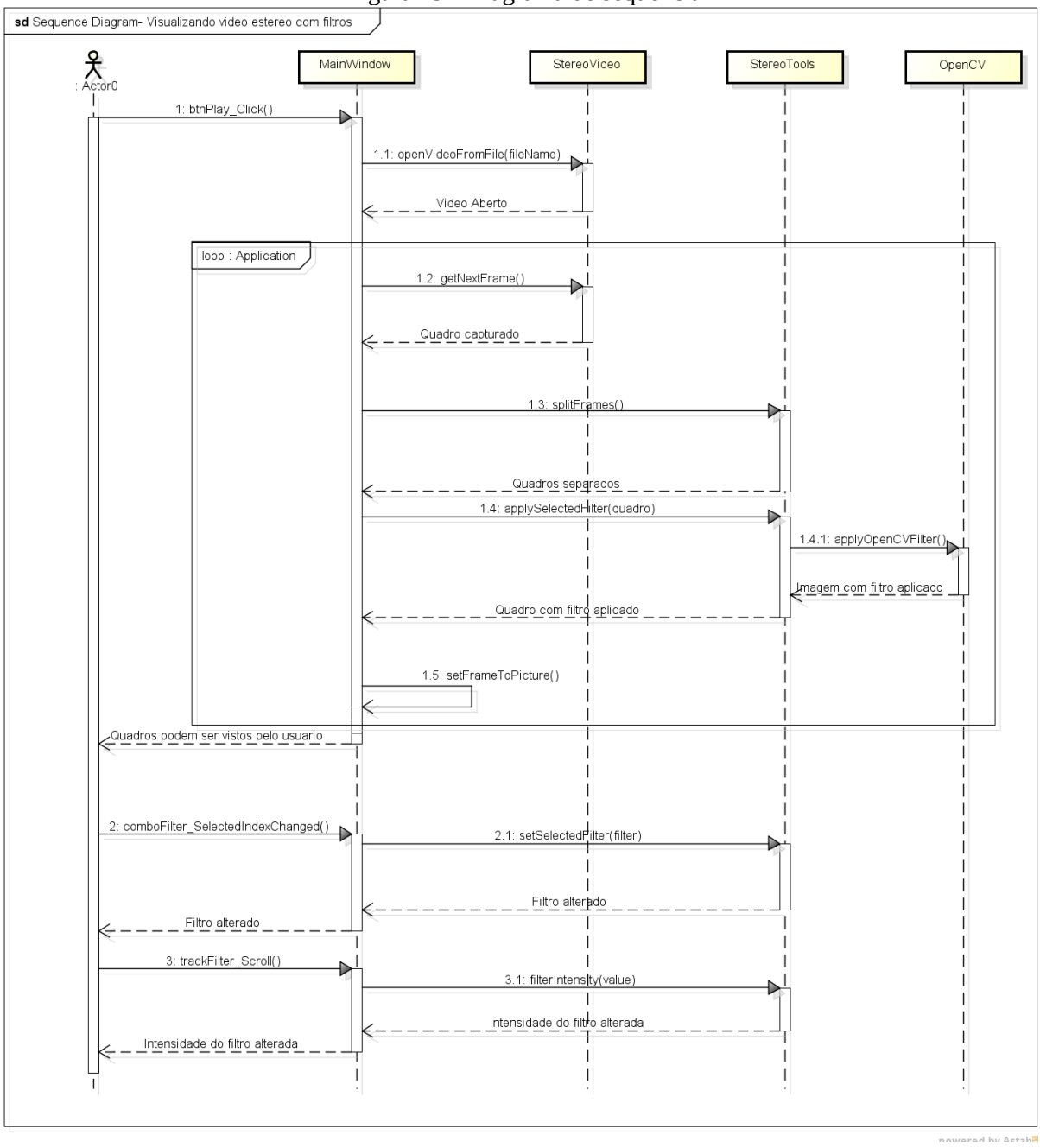
Para ilustrar o funcionamento da biblioteca na aplicação de filtros do OpenCV em vídeos e a exibição deste para o usuário é apresentadaa a sequência de chamadas desde o carregamento de um vídeo estéreo até a exibição deste separado e com filtros aplicados. O diagrama de sequência da Figura 19 descreve especificamente essas etapas abstraindo os procedimentos executados pela aplicação.

O fluxo inicia quando o usuário pressiona o botão `play` que vai chamar a função `openVideoFromFile` que prepara a biblioteca para efetuar a leitura dos quadros do vídeo aberto pela função. No próximo passo a aplicação entra em um laço que fica lendo os quadros do arquivo e vai exibir estes para o usuário.

Os processos que ocorrem dentro do laço principal começam com a chamada da função `getNextFrame()`, que traz o próximo quadro da classe `StereoVideo`. Com este quadro capturado é possível chamar a função `splitFrames()`, que vai quebrar a imagem estéreo em duas imagens 2D. Então a função `applySelectedFilter()` poderá ser chamada passando como parâmetro as duas imagens 2D. Essa função utiliza o filtro selecionado (`selectedFilter`) juntamente com a sua intensidade (`filterIntensity`) e retorna uma imagem filtrada.

Ao final do fluxo, as duas imagens filtradas serão exibidas para o usuário por meio da função `setFrameToPicture()` que aplica a textura das imagens na tela.

Figura 19 - Diagrama de sequência



3.3 IMPLEMENTAÇÃO

Nesta seção são apresentadas técnicas e ferramentas utilizadas na implementação da biblioteca. Além disso, também a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

Para o desenvolvimento da biblioteca e da ferramenta na plataforma Windows utilizou-se o Microsoft Visual Studio em conjunto com a biblioteca OpenCV e o Microsoft .NET Framework. Utilizou-se a versão 8.1 do Windows, a versão 2.4.8 do OpenCV, a versão 12.0 do Microsoft Visual Studio Ultimate 2013 e a versão 4.5 do Microsoft .NET Framework.

No desenvolvimento da biblioteca utilizou-se a linguagem C++ e para a interface da ferramenta foram utilizados recursos da linguagem Visual C++ pois esta disponibiliza algumas facilidades para o desenvolvimento das telas explicativas na ferramenta.

A biblioteca do OpenCV foi utilizada no desenvolvimento deste trabalho. Por este motivo ela é necessária tanto para compilar a biblioteca quanto para executar a ferramenta, pois é necessário referenciar o OpenCV antes de fazer uso da biblioteca e ferramenta apresentadas neste trabalho.

Tanto a biblioteca como a ferramenta estão hospedadas em um repositório público no serviço Bitbucket (STEREOLIBRARY, 2014). Para registrar mudanças na codificação e manter o repositório atualizado utilizou-se a ferramenta Git (GIT, 2014).

3.3.2 Preparar o ambiente

Para preparar o ambiente é de suma importância referenciar o OpenCV tanto na biblioteca como na ferramenta. Para realizar a configuração destas é necessário realizar um processo inicial que é explicado logo a seguir.

Para facilitar o processo de configuração do OpenCV no Windows, o projeto no Visual Studio está previamente configurado com caminhos relativos a uma variável de ambiente. Ou seja, após a instalação do OpenCV é recomendado atribuir uma variável de ambiente `OPENCV_DIR` com o caminho até a pasta `/vc12` que deve estar dentro da estrutura de pastas do OpenCV (“`OPENCV_DIR = C:\opencv\build\x86\vc12`”). Este caminho irá variar dependendo do local em que o openCV está instalado. Para atribuir um valor a uma variável no ambiente Windows, basta executar o comando no terminal `SET OPENCV_DIR=X`, onde `X` é o caminho relativo até a pasta `\vc12` do OpenCV. O comando `SET` só salva esta variável na sessão atual. Se o objetivo for salvar a variável de ambiente permanentemente no sistema deve ser usado o comando `SETX`.

Para compilar a biblioteca em *debug* ou *release* é necessário referenciar no mínimo as seguintes bibliotecas do OpenCV: `opencv_core`, `opencv_highgui`, `opencv_imgproc` e `opencv_objdetect`. As bibliotecas necessárias para compilar a biblioteca em modo *debug* podem ser vistas na Figura 20 dentro do projeto do Visual Studio e para compilar a biblioteca em modo *release* é necessário utilizar as dependências exibidas na Figura 21.

Figura 20 - Itens necessários para compilar a biblioteca em modo debug

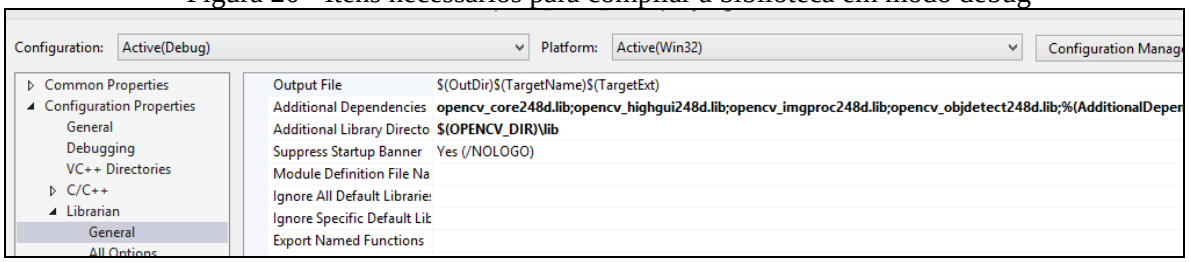
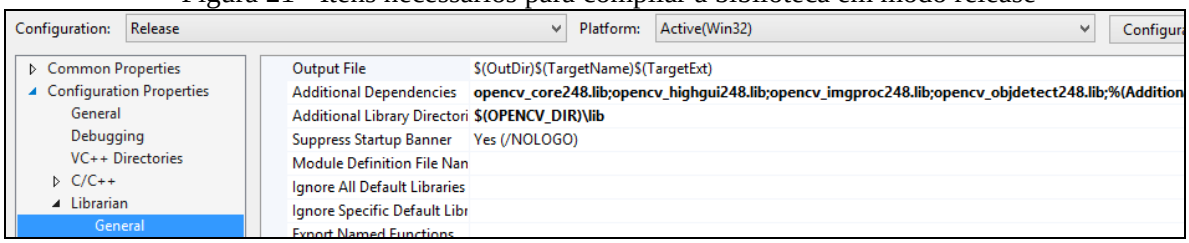


Figura 21 - Itens necessários para compilar a biblioteca em modo release



Pode ser verificado que a única diferença existente nas dependências de *debug* para *release* é que o arquivo `.lib` usado possui um `d` no fim de sua nomenclatura, sinalizando o modo *debug*.

As configurações explicadas anteriormente são importantes não só para a biblioteca como também para a ferramenta que vai usar a biblioteca, seja aquela disponibilizada neste projeto ou seja uma ferramenta nova que vai fazer uso da biblioteca. Na criação de um novo projeto será necessário além de referenciar o OpenCV também referenciar a referida biblioteca. Seguem os passos para realizar esta tarefa:

a) referenciar os *includes* da biblioteca referida nas propriedades do projeto novo em *`Configuration Properties > C/C++ > General > Additional Include Directories`* como mostrado na Figura 22;

b) referenciar o caminho do arquivo `.lib` gerado pela biblioteca referida após compilação dentro de *`Configuration Properties > Linker > General > Additional Library Directories`* (Figura 23);

c) importar a biblioteca para poder fazer uso das classes presentes na mesma a partir de `#include "StereoTools.h"` e `#include "StereoVideo.h"`.

Figura 22 – Referência para os *headers* da biblioteca

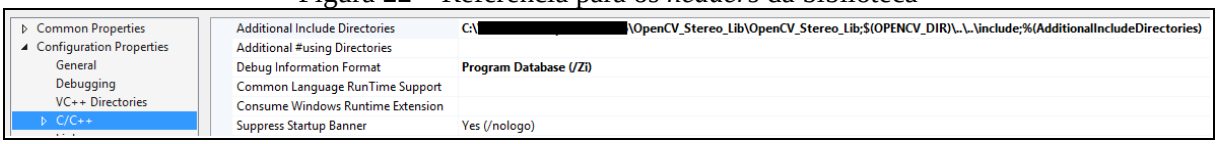
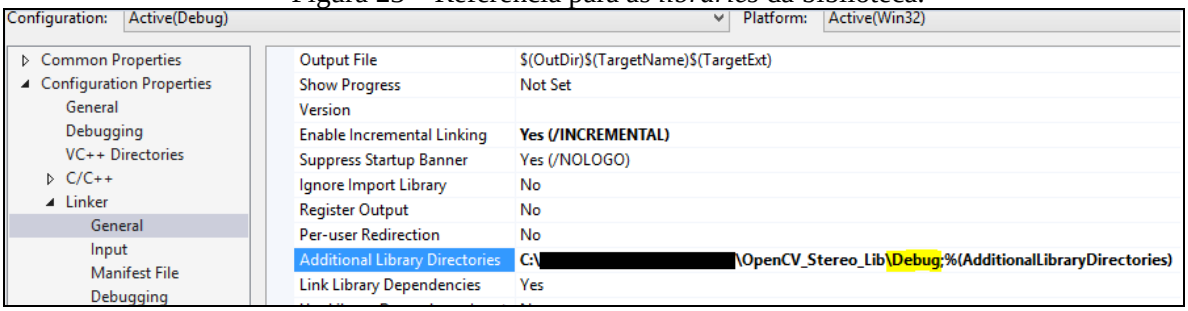


Figura 23 – Referência para as *libraries* da biblioteca.



3.3.3 Operacionalidade da implementação

Nesta seção são explorados assuntos pertinentes à utilização da biblioteca e da ferramenta. A seção está dividida em duas partes, a primeira dedicada à biblioteca, que é do interesse de desenvolvedores; e a segunda dedicada à ferramenta, que pode ser de interesse tanto de desenvolvedores como de usuários.

3.3.3.1 Estrutura da biblioteca

Na Figura 24 pode ser observada a estrutura da biblioteca na versão 1.0 (BIBLIOTECA, 2014), o código fonte é disponibilizado abertamente em um repositório bitbucket (STEREOLIBRARY, 2014) e alterações nesta estrutura serão esperadas.

Atualmente a biblioteca está dividida em duas classes principais: a classe `StereoTools` (Quadro 8) que agrupa um conjunto de ferramentas para realizar modificações nos videos estereoscópicos, que por sua vez estão na classe `StereoVideo` (

Quadro 7) que agrupa diversas propriedades dos vídeos estéreo.

A biblioteca está fazendo uso de alguns filtros do OpenCV, atualmente estão sendo utilizados os seguintes algoritmos do OpenCV:

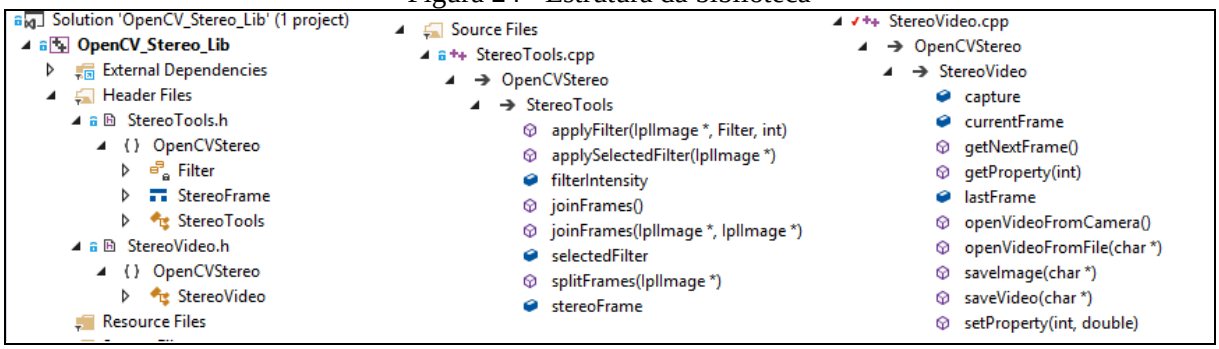
a) detector de borda de *Canny* (item j na seção 2.3.1);
b) erosão (item b da seção 2.3.1);
c) dilatação (item b da seção 2.3.1);
d) negativo.

Os filtros usados pela biblioteca estão dentro da classe enumeradora `Filter` para facilitar a adição e remoção de filtros presentes na biblioteca OpenCV. A aplicação de um filtro acontece com uma verificação na enumeração dessa classe e o filtro selecionado é

aplicado. Este processo acontece na função `StereoTools::applyFilter`, o qual pode ser verificado no

Quadro 9.

Figura 24 - Estrutura da biblioteca



Quadro 7 – Conteúdo da classe `StereoVideo`

```
class StereoVideo
{
private:
	static CvCapture* capture;
public:
	static IplImage* currentFrame;
	static IplImage* lastFrame;
	static void openVideoFromFile(char* filePath);
	static void openVideoFromCamera();
	static IplImage* getNextFrame();
	static double getProperty(int property_id);
	static void setProperty(int proparty_id, double value);

	static void saveImage(char* filePath);
	static void saveVideo(char* filePath);
};
```

Quadro 8 – Conteúdo da classe `StereoTools`

```
class StereoTools
{
private:
	static Filter selectedFilter;
public:
	static StereoFrame stereoFrame;
	static int filterIntensity;

	static IplImage* joinFrames();
	static IplImage* joinFrames(IplImage* left, IplImage* right);

	static cv::Mat applySelectedFilter(IplImage* imagePointer);
	static cv::Mat applyFilter(IplImage* leftFrame, Filter filter =
Filter::None, int intensity = 0);

	static Filter getSelectedFilter(){ return selectedFilter; }

	static void setSelectedFilter(Filter filter){ selectedFilter =
filter; }

	static StereoFrame splitFrames(IplImage* mainImage);

};
```

Quadro 9 – Metodo `applyFilter` da classe `StereoTools`.

```
01 Mat OpenCVStereo::StereoTools::applyFilter(IplImage* frame, Filter
filter, int intensity)
02 {
03  cv::Mat myFrameResult = frame;
04  try
05  {
06      switch (filter)
07      {
08
09      case Filter::Canny:
10          cvtColor(cv::Mat(frame), myFrameResult, CV_BGR2GRAY, 3);
11          GaussianBlur(myFrameResult, myFrameResult, Size(7, 7), 1.5,
1.5);
12          Canny(myFrameResult, myFrameResult, 0, 20 + intensity, 3);
13          break;
14
15      case Filter::Erode:
16          cvErode(frame, frame, 0, 2 + intensity);
17          myFrameResult = frame;
18          break;
19
20      case Filter::Dilate:
21          cvDilate(frame, frame, 0, 2 + intensity);
22          myFrameResult = frame;
23          break;
24      case Filter::Negative:
25          cvNot(frame, frame);
26          myFrameResult = frame;
27          break;
28      case Filter::None:
29      default:
30          return frame;
31          break;
32      }
33  }
34  catch (Exception ex)     {}
35
36  return myFrameResult;
37 }
```

Na linha 3 o quadro que será alterado é armazenado. Na linha 6 há o `switch` que verifica o tipo de filtro que será aplicado. Na linha 10 o quadro é convertido de RGB para escala de cinza. Na linha 11 um filtro gaussiano é aplicado no quadro para melhorar o resultado da detecção de bordas, e na linha 12 o filtro *canny* é aplicado.

Na linha 16 o filtro erode é aplicado em cima da própria imagem. Na linha 21 a função de dilatação é aplicada no próprio quadro. Na linha 25 a função para negar pixels do OpenCV é chamada passando o próprio quadro como parametro, dessa maneira as cores são invertidas. Por fim o quadro resultante é retornado na linha 36.

Para contribuir com novos filtros na biblioteca basta adicionar o algoritmo do filtro no `switch` da função `applyFilter` que este será aplicado sobre a textura respectiva do quadro estéreo.

O OpenCV também é utilizado para outras funções como o carregamento e armazenamento de texturas. Esse é outro motivo que fez com que o OpenCV fosse escolhido como biblioteca de suporte à biblioteca referida.

Dentro da classe `StereoTools` se encontram os métodos que são responsáveis pela separação e unificação dos quadros estéreo. Estes são respectivamente `splitFrames` e `joinFrames`. Estas funções fazem o papel de codificação e decodificação do conteudo como mencionado anteriormente nas seções 2.2.2.1 e 2.2.2.3, o funcionamento delas pode ser verificado nos quadros Quadro 10 e Quadro 11.

Quadro 10 – Função `splitFrames` da classe `StereoTools`.

```
01 StereoFrame OpenCVStereo::StereoTools::splitFrames(IplImage*
mainImage){
02   if (!mainImage)
03   {
04       mainImage = StereoVideo::currentFrame;
05   }
06   cvReleaseImage(&stereoFrame.left);
07   cvReleaseImage(&stereoFrame.right);
08   stereoFrame.left = NULL;
09   stereoFrame.right = NULL;
10   cvSetImageROI(mainImage, cvRect(0, 0, mainImage->width / 2, mainImage-
>height));
11   stereoFrame.left = cvCreateImage(cvGetSize(mainImage), mainImage-
>depth, mainImage->nChannels);
12   cvCopy(mainImage, stereoFrame.left, NULL);
13   cvResetImageROI(mainImage);
14
15   cvSetImageROI(mainImage, cvRect(mainImage->width / 2, 0, mainImage-
>width / 2, mainImage->height));
16   stereoFrame.right = cvCreateImage(cvGetSize(mainImage), mainImage-
>depth, mainImage->nChannels);
17   cvCopy(mainImage, stereoFrame.right, NULL);
18   cvResetImageROI(mainImage);
19
20   return stereoFrame;
21 }
```

Na linha 4 o quadro atual é capturado como imagem principal. Nas linhas 6, 7, 8 e 9 ocorre a limpeza da memoria para evitar *memory leak*. Na linha 10 a região de interesse é selecionada, que nesse exemplo é toda a parte esquerda da imagem, e na linha 11 a imagem esquerda é criada a partir da região de interesse marcada. Lembrar sempre de limpar a região de interesse quando concluir a tarefa como mostrado na linha 13.

O mesmo procedimento é repetido para o quadro direito, das linhas 15 até 18. Por fim o objeto `stereoFrame` é retornado, este contendo as 2 imagens extraídas (esquerda e direita).

Quadro 11 – Função `joinFrames` da classe `StereoTools`.

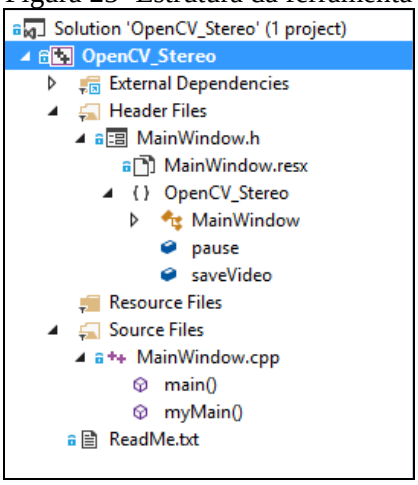
```
01 IplImage*  OpenCVStereo::StereoTools::joinFrames(IplImage* left,
IplImage* right)
02 {
03  IplImage* returnFrame;
04  CvSize cvSize = CvSize();
05  cvSize.height = cvGetSize(left).height;
06  cvSize.width = cvGetSize(left).width + cvGetSize(right).width;
07
08
09  returnFrame = cvCreateImage(cvSize, left->depth, left->nChannels);
10
11
12  cvSetImageROI(returnFrame, cvRect(0, 0, left->width, left->height));
13  cvCopy(left, returnFrame, NULL);
14
15  cvSetImageROI(returnFrame, cvRect(left->width, 0, returnFrame->width,
returnFrame->height));
16  cvCopy(right, returnFrame, NULL);
17
18  cvResetImageROI(returnFrame);
19
20  return returnFrame;
21 }
```

Na linha 5 e 6 são calculados os valores do quadro que será montado. Na linha 9 a imagem de retorno é criada com os tamanhos préviamente calculados. Na linha 12 uma região de interesse é posicionada no quadro estéreo onde será armazenado o quadro esquerdo. Na linha 13 as informações do quadro esquerdo são copiadas para dentro do quadro estéreo e o mesmo se repete nas linhas 15 e 16 para o quadro direito. Na linha 18 a região de interesse é resetada e por fim na linha 20 o quadro estéreo é retornado.

3.3.3.2 Estrutura da ferramenta

Na Figura 25 pode ser analisada a estrutura inicial de pastas da ferramenta na versão 1.

Figura 25- Estrutura da ferramenta



O arquivo MainWindow.cpp contém apenas a chamada da interface do Visual C++ que é feita dentro do método principal como pode ser visto no Quadro 12.

Quadro 12 – Código que instancia a tela da ferramenta

```
int myMain()
{
    OpenCV_Stereo::MainWindow ^mw = gcnew OpenCV_Stereo::MainWindow();
    mw->ShowDialog();
    return 0;
}
```

Para realizar a reprodução de um vídeo se faz necessário a implementação de um laço. No Quadro 13 podem ser verificadas as chamadas necessárias da biblioteca para capturar as informações necessárias para a execução de um vídeo estéreo usando a biblioteca. Este laço é executado dentro de um intervalo fixo de tempo.

Quadro 13 – Laço na ferramenta que percorre todos os quadros de um vídeo.

```
01 private: System::Void timer1_Tick(System::Object^  sender,
System::EventArgs^  e){
02      ...
03      OpenCVStereo::StereoVideo::getNextFrame();
04      ...
05      if (OpenCVStereo::StereoVideo::currentFrame != NULL)
06      {
07          OpenCVStereo::StereoFrame splittedFrame =
OpenCVStereo::StereoTools::splitFrames(nullptr);
08
09          Mat left =
OpenCVStereo::StereoTools::applySelectedFilter(splittedFrame.left);
10          Mat right =
OpenCVStereo::StereoTools::applySelectedFilter(splittedFrame.right);
11          setFrameToPicture(left, pbMainVideo);
12          setFrameToPicture(right, pbRightVideo);
13          ...
14      }
15 }
```

Na linha 3 acontece a chamada na biblioteca para pegar o próximo quadro. Na linha 7 o quadro capturado é quebrado em 2 para a exibição dele em partes. Nas linhas 9 e 10 o filtro

selecionado é aplicado nos quadros esquerdo e direito respectivamente. Por fim, nas linhas 11 e 12 os quadros são desenhados na tela da ferramenta para visualização do usuário por meio do método `setFrameToPicture`. Este pode ser verificado no Quadro 14. Ele é usado quando se deseja aplicar o conteúdo de uma imagem em forma de matriz `Mat` em uma `PictureBox` para visualização em tela.

Quadro 14 - Metodo setFrameToPicture

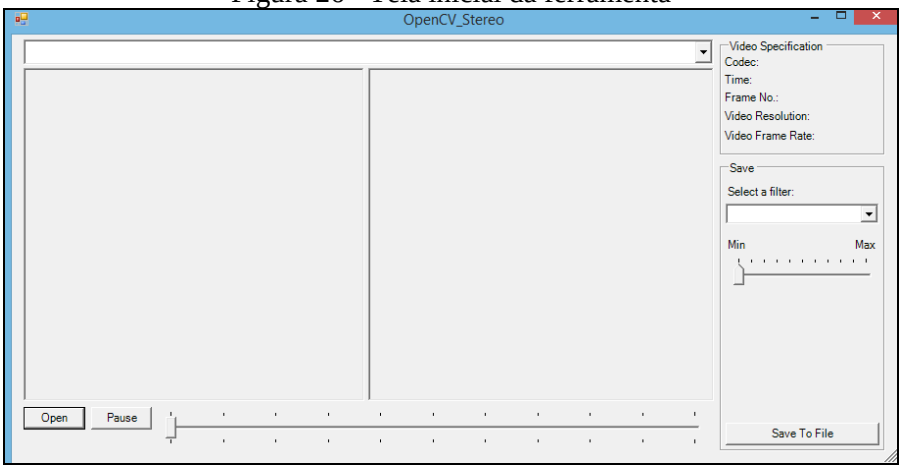
```
01 private: System::Void setFrameToPicture(Mat frame, PictureBox^ picture)
02 {
03     System::Drawing::Graphics^ graphicsLeft = picture-
>CreateGraphics();
04     System::IntPtr ptrLeft(frame.ptr());
05     System::Drawing::Bitmap^ bL;
06     switch (frame.type())
07     {
08     case CV_8UC3:
09      bL = gcnew System::Drawing::Bitmap(frame.cols, frame.rows,
frame.step,
10          System::Drawing::Imaging::PixelFormat::Format24bppRgb,
ptrLeft);
11      break;
12     case CV_8UC1:
13      bL = gcnew System::Drawing::Bitmap(frame.cols, frame.rows,
frame.step,
14          System::Drawing::Imaging::PixelFormat::Format8bppIndexed,
ptrLeft);
15      break;
16     default:
17      break;
18     }
19     System::Drawing::RectangleF rectL((float)0, (float)0,
(float)picture->Width, (float)picture->Height);
20     graphicsLeft->DrawImage(bL, rectL);
21 }
```

Caso a imagem tenha três canais, o formato dos pixels selecionado é o `Format24bppRgb`. Caso a imagem tenha apenas um canal (ex: tons de cinza) o formato de pixel selecionado é o formato `Format8bppIndexed`. Esse controle é realizado para evitar problemas na exibição de alguns tipos de imagem.

Para testar a ferramenta basta executar o projeto OpenCV_Stereo no Visual Studio que a aplicação será compilada e aberta. Considerando que a ferramenta foi compilada com sucesso, o executável da ferramenta está armazenado no computador, sendo assim, também possível fazer a execução do mesmo.

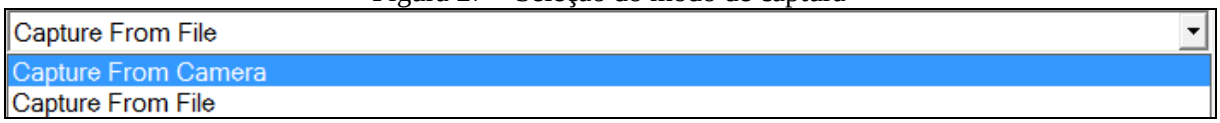
A tela inicial da ferramenta pode ser vista na figura 26. A ferramenta faz uso das funções presentes na biblioteca. Ou seja o usuário pode abrir, visualizar, aplicar filtros e salvar imagens estereoscópicas.

Figura 26 - Tela inicial da ferramenta



O primeiro passo é realizar a seleção do modo de captura como ilustrado na Figura 27.

Figura 27 – Seleção do modo de captura



Se o modo *Capture From Camera* (Capturar da Câmera) for selecionado, a ferramenta já começa a captura de quadros da câmera, e a imagem é exibida diretamente para o usuário. Se o modo selecionado for *Capture From File* (Capturar do arquivo) o botão `open` (figura 26) deve ser pressionado. Este irá solicitar que um vídeo ou imagem seja selecionada em seu computador. Se um vídeo for selecionado o aplicativo vai carregar o vídeo que será exibido para o usuário já com os quadros separados, como pode ser visto na Figura 28.

Figura 28 – Vídeo sendo reproduzido na ferramenta

Depois do vídeo carregado é possível selecionar um filtro na caixa de seleção *Select a filter*. Este será aplicado diretamente na imagem que está sendo reproduzida e o resultado pode ser imediatamente visto. Também é possível selecionar a intensidade do mesmo.

O tempo do vídeo pode ser controlado pela *trackbar* logo abaixo do vídeo, sendo que está barra só é visível quando um vídeo está carregado.

Se o botão `save to file` for pressionado, uma caixa de diálogo será exibida e basta selecionar um local para salvar o vídeo. Depois disso é necessário selecionar o codec que será utilizado para a compressão do vídeo, como pode ser visualizado na Figura 29.

Depois do codec selecionado o vídeo será salvo com os dois quadros mesclados em um arquivo estéreo, que poderá ser reproduzido em qualquer dispositivo capaz de reproduzir conteúdo 3D.

Todo o processo mencionado anteriormente funciona da mesma forma para imagens, como pode ser visto na Figura 30, ou seja, ocorre a separação do quadro estereo, aplicação de filtros seleção de intensidade e armazenamento da imagem resultado em arquivo. Basicamente, a diferença de um vídeo e de uma imagem para a ferramenta é a quantidade de quadros que o arquivo possui.

Figura 29 – Selecionando compressão do arquivo.

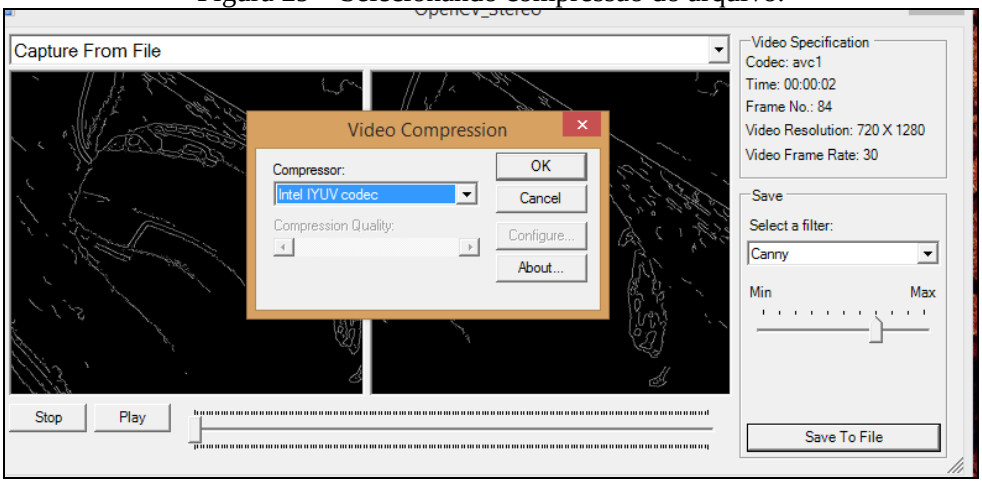


Figura 30 – Imagem carregada pela ferramenta

3.4 RESULTADOS E DISCUSSÃO

Os resultados obtidos neste trabalho são a própria biblioteca e ferramenta que estão disponibilizadas para uso e alteração pública.

A biblioteca realiza abstração de tarefas como: aquisição de uma imagem estéreo; separação dos quadros estéreos presentes nesta imagem, permitindo a alteração destes quadros separadamente; e por fim a geração de um arquivo estéreo no mesmo formato ou de formato diferente.

O consumo de memória da ferramenta mostrou-se estável ao longo da execução da ferramenta. A alocação da memória utilizada pela ferramenta depende do tamanho e compressão utilizada nos quadros que estão sendo carregados. Para realização do teste de memória no tocador de vídeo da ferramenta foi utilizado um vídeo na resolução 1280x720. Foi constatado a média de 42.960K de memória alocada, o que foi considerado um valor aceitável já que o valor fica próximo do valor mais baixo obtido utilizando outros *players* de vídeo como o Windows Media Player e o Media Player Classic.

O desempenho da ferramenta também foi considerado aceitável, pois ela consegue reproduzir os vídeos realizando a separação dos quadros em tempo de execução. A aplicação dos filtros também acontece em tempo de execução, com exceções obviamente, já que filtros mais pesados podem demorar um tempo considerável para serem aplicados o que vai desacelerar a reprodução dos quadros. Como por exemplo o filtro de detecção de bordas, que faz com que a taxa normal de 29.9 quadros por segundo caia para 9 quadros por segundo, o que é um resultado aceitável pela quantidade de operações executadas pelo filtro.

No Quadro 15 podem ser verificadas algumas comparações feitas com os trabalhos correlatos. Nas comparações realizadas, nota-se que a maior diferença está nos formatos que os outros trabalhos suportam.

Pode ser verificado nos trabalhos correlatos que o formato de saída é um arquivo de imagem anáglifo, isso acaba sendo uma limitação deles. Então os trabalhos correlatos podem servir de exemplo para mostrar a importância de se utilizar uma biblioteca como a que foi desenvolvida neste trabalho, pois se eles tivessem utilizado a biblioteca desenvolvida o resultado de seus trabalhos poderiam ser exportados para outros formatos aumentando o escopo de seus projetos.

Quadro 15 – Comparação entre trabalhos correlatos

Características / Nome dos trabalhos	Biblioteca Desenvolvida	OPENSTEREO	Cálculo de volume efetivo usando estereoscopia.	Protótipo de um ambiente de visualização com técnicas de estereoscopia.
Suporte a câmeras estéreo	Sim	Não	Não (usa uma única câmera com auxilio de esteira)	Não
Suporte a abertura de arquivos estéreo	Sim	Não	Não	Sim
Suporte a captura de ambiente gráfico	Não	Sim	Não	Não
Saída *Anaglyph*	Não*	Sim	(Não gera imagens)	Sim
Saída *Side By Side*	Sim	Não	(Não gera imagens)	Não
Saída *Over/Under*	Sim	Não	(Não gera imagens)	Não
Realiza cálculos de volume	Não*	Não	Sim	Não

*Pode ser implementado.

Após os estudos relacionados à estereoscopia, a tarefa que parecia ser complexa, que é a aplicação dos algoritmos do OpenCV nos vídeos estéreo, acabou se tornando simples, pois percebeu-se que um vídeo estéreo não passa de duas imagens de ângulos diferentes. Portanto a maior complexidade computacional se encontra nas tarefas de codificação e decodificação do conteúdo. Depois da decodificação tem-se dois quadros 2D, bastando então aplicar os algoritmos em cada um dos quadros independente da mesma maneira que seria feita em um único quadro. Depois disso basta realizar a junção dos dois quadros em um quadro estéreo novamente.

Por fim, vale mencionar que o valor da ferramenta está na abstração do processo de abertura de vídeos estereoscópicos e gravação, pois, como já explicado na fundamentação teórica, existem diversas maneiras de realizar a compactação, transmissão e descompactação do conteúdo 3D. E com o uso da biblioteca desenvolvida essas tarefas se tornam mais simples, o que vai permitir aumentar o escopo da sua aplicação estereoscópica, pois a biblioteca traz suporte a diferentes formatos estéreo tanto de entrada como de saída.

4 CONCLUSÕES

Este trabalho apresentou a criação de uma biblioteca, cujo objetivo principal é fornecer suporte à aplicação de algoritmos da área de visão computacional fornecidos pela biblioteca OpenCV em imagens estereoscópicas. Informações sobre estereoscopia e percepção foram fornecidas, além de explicação de conceitos envolvidos na área. A biblioteca desenvolvida conseguiu abstrair diversas tarefas do OpenCV com o intuito de auxiliar e motivar outros desenvolvedores no desenvolvimento de aplicações futuras.

Também foi apresentada uma ferramenta que faz uso da biblioteca criada. O objetivo principal da ferramenta é explicar o funcionamento da biblioteca por meio da implementação das funcionalidades presentes nela. A ferramenta que foi desenvolvida possui todos os algoritmos que estão disponibilizados na biblioteca, e nela é possível verificar qual a maneira correta de fazer uso da biblioteca.

A implementação tanto da biblioteca como da ferramenta foram detalhadamente explicadas facilitando assim a reprodução deste trabalho bem como sua extensibilidade.

É importante ressaltar que tanto a biblioteca quanto a ferramenta utilizam OpenCV e, portanto, ainda existe o esforço inicial para poder compilar corretamente o produto. Este processo também foi detalhado neste trabalho.

Os recursos do ambiente VisualStudio foram amplamente utilizados, juntamente com ferramentas como GIT e o serviço de repositórios BitBucket. Tanto a biblioteca como a ferramenta estão disponibilizadas abertamente no serviço BitBucket com o intuito de permitir que outros desenvolvedores façam uso e contribuam com o código da biblioteca e da ferramenta.

4.1 EXTENSÕES

Como trabalhos futuros, sugere-se as seguintes extensões:

a) otimizar o tempo que a biblioteca leva para aplicar os algoritmos nas imagens e vídeos;
b) adicionar suporte a outros algoritmos do OpenCV que não foram usados neste trabalho;
c) tornar a biblioteca mais independente da plataforma Windows, visando compilar para outras plataformas, como Mac e Linux;
d) adicionar suporte a outros formatos de arquivo 3D que não foram explorados neste trabalho. Como por exemplo: Anaglyph, Checkerboard, e outros que podem ser encontrados na seção 2.2.2.1.

REFERÊNCIAS

3D COMPUTER graphics. In: WIKIPÉDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2013. Disponível em: <http://en.wikipedia.org/wiki/3D_computer_graphics>. Acesso em: 02 nov. 2013.

3D WITHOUT GLASSES: the future of 3D technology? 2012. Disponivel em: <http://convert-to-3d.com/blog/3d_without_glasses_the_future_of_3d_technology.html>. Acesso em: 14 set. 2013.

AMD: Stereoscopic 3D For Professionals. 2011. Disponível em: <http://www.cgw.com/documents/pdfs/amd_hd_3d_whitepaperhrscreen.pdf>. Acesso em: 14 jun. 2014.

BARROS, Renato C. **Cálculo de volume efetivo de objetos em movimento usando estereoscopia.** 2009. 48 f. Trabalho de Conclusão de Curso (Bacharel em Engenharia da Computação) – Escola Politécnica de Pernambuco, Universidade de Pernambuco, Recife.

BIBLIOTECA: Versão 1.0 da biblioteca no BitBucket. 2014. Diponível em: <https://bitbucket.org/ricardois/stereolibrary/src/0f677dd9ab25?at=master>. Acesso em: 2 jul. 2014.

BRADSKI, Gary; KAEHLER, Adrian. **Learning OpenCV**. Sebastopol: O'Reilly Books, 2008.

COMPUTER VISION. In: WIKIPÉDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2013. Disponível em: <http://en.wikipedia.org/wiki/Computer_vision>. Acesso em: 02 nov. 2013.

GATEAU, Samuel; NASH, Steve. **Implementing stereoscopic 3D in your applications**. 2010. Disponível em: <www.nvidia.com/content/GTC-2010/pdfs/2010_GTC2010.pdf>. Acesso em: 14 set. 2013.

GIT. **Open source distributed version control system.** 2014. Disponível em: <http://git-scm.com/>. Acesso em: 14 jun. 2014.

KOCH, Márcio. **Visão computacional para reconhecimento de faces aplicado na identificação e autenticação de usuários na web.** 2012. 138 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

KOHLER, Eduardo. **Protótipo de software para análise da percepção de profundidade aparente em computação gráfica.** 2001. 72 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

LEITE, Pedro. J. **OpenStereo: Uma Biblioteca Para Suporte ao Desenvolvimento de Aplicações Estereoscópicas.** 2006. 53 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife.

MACHADO, Liliane. S. **A realidade virtual em aplicações científicas.** São José dos Campos, 1997.

MOMM, Edson. **Protótipo de um ambiente de visualização com técnicas de estereoscopia.** 2001. 50 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

NVIDIA Corporation: **Nvidia 3D Vision Pro and Stereoscopic 3D.** 2010. Disponível em: <http://www.nvidia.com/docs/IO/40505/WP-05482-001_v01-final.pdf>. Acesso em: 14 jun. 2014.

NVIDIA Gallery. **3D Vision Gallery.** 2012. Disponível em: <http://photos.3dvisionlive.com/ricardois/image/4f4d500a378501884a000002/>. Acesso em: 14 jun. 2014.

OCULUSVR: oculus rift. 2014. Disponível em: <http://www.oculusvr.com/rift/>. Acesso em: 14 jun. 2014.

OPENCV: image processing. 2014. Disponível em: <http://docs.opencv.org/doc/tutorials/imgproc/table_of_content_imgproc/table_of_content_imgproc.html>. Acesso em: 14 jun. 2014.

PAIVA, J. et al. Estereoscopia no Ensino da Química. Química e Ensino. Disponível em: <http://nautilus.fis.uc.pt/personal/jtrindade/~jtrindade/pub/estereo.pdf>. Acesso em: 14 jun. 2014.

PIRODDI, R. **Stereoscopic 3D Techonologies.** 2010. Disponível em: <http://www.snellgroup.com/documents/white-papers/white-paper-stereoscopic_3d_technologies.pdf>. Acesso em: 14 jun. 2014.

SCHNEIDER, Neil; NIKSHYCH, Yuriy. **MTBS' official S-3D gaming anomaly guide**. 2010. Disponível em: <http://www.mtbs3d.com/index.php?option=com_content&view=article&id=11207&Itemid=94>. Acesso em: 14 set. 2013.

SHIRAI, Yoshiaki. **Three-dimensional computer vision**. Berlin: Springer-Verlag, 1987.

STEREOGRAPHICS: developers handbook. 1997. Disponível em: <www.cs.unc.edu/Research/stc/FAQs/Stereo/stereo-handbook.pdf>. Acesso em: 14 set. 2013.

STEREOLIBRARY: Repositório no serviço BitBucket. 2014. Disponível em: <https://bitbucket.org/ricardois/stereolibrary>. Acesso em 2 jul. 2014.

STEREOSCOPY. In: WIKIPÉDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2013. Disponível em: <http://en.wikipedia.org/wiki/Stereoscopy>. Acesso em: 16 ago. 2013.

SZELISKI, Richard. **Computer vision**: algorithms and applications. 2nd ed. London: Springer-Verlag, 2011.

VETRO, A. et al. **3D-TV Content Storage and Transmission**. IEEE Transactions on Broadcasting, v. 57, n. 2, p. 384-394, jun 2011.

VISUAL system. In: WIKIPÉDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2013. Disponível em: <http://en.wikipedia.org/wiki/Visual_system>. Acesso em: 16 ago. 2013.

WOODS, A. **How are Crosstalk and Ghosting defined in the Stereoscopic Literature?.** 2011. Diponível em: <http://cmst.curtin.edu.au/local/docs/pubs/2011-14-woods-crosstalk_and_ghosting.pdf>. Acesso em: 14 jun. 2014.