BEATRIZ CASTRO AMARAL RIZZUTO
WESLLEY MORELLATO BUENO

PASSEIO IMERSIVO UTILIZANDO REALIDADE VIRTUAL EM TEMPO REAL

São Paulo
2016

BEATRIZ CASTRO AMARAL RIZZUTO
WESLLEY MORELLATO BUENO

PASSEIO IMERSIVO UTILIZANDO REALIDADE VIRTUAL EM TEMPO REAL

Trabalho de Formatura da Escola Politécnica
da Universidade de São Paulo

Área de Concentração:
Engenharia Elétrica - Ênfase em Computação

Orientador: Romero Tori
Coorientador: Ellen Collaço de Oliveira

São Paulo
2016

Agradecimentos

Primeiramente à Escola Politécnica da Universidade de São Paulo, por todo o conhecimento passado a nós durante os anos de graduação, essenciais para a criação deste trabalho.

Ao orientador Romero Tori e à coorientadora Ellen Colaço, pela mentoria no Trabalho de Formatura, pelas incontáveis sugestões e pela inabalável paciência.

Aos meus professores e colegas que dividiram comigo essa jornada e à minha família pela paciência infinita.

Beatriz Rizzuto

Aos amigos que conheci durante a graduação, aos familiares que me acompanharam durante esses anos e em especial à minha mãe, Sonia Morellato, fonte inesgotável de apoio e de dedicação e razão de todas as minhas conquistas.

Weslley Morellato

"Os átomos de nosso corpo podem ser rastreados às estrelas que os produziram em seus núcleos e então explodiram estes ricos ingredientes através da galáxia, há bilhões de anos atrás. Por essa razão, nós somos biologicamente conectados a todos os outros seres vivos deste planeta. Nós somos quimicamente conectados à todas as moléculas na Terra. E somos atomicamente conectados a todos os átomos do universo. Nós somos, não figurativamente, mas literalmente, poeira estelar."

(Neil deGrasse Tyson)

"Your vision will become clear only when you can look into your own heart. Who looks outside, dreams; who looks inside, awakes."

(Carl Gustav Jung)

Resumo

Passeio imersivo é um termo que define a experiência que um indivíduo tem ao observar um local à distância em tempo real e utilizando a tecnologia de realidade virtual. Tal experiência exige que um segundo indivíduo esteja presente no local portando uma câmera capaz de transmitir, em tempo real, a imagem do ambiente ao seu redor para o observador. Esse projeto tem como objetivo propor uma sistema que implementa o conceito de passeio imersivo utilizando tecnologias disponíveis atualmente. A implementação difere de produtos semelhantes em relação ao grau de imersão do observador, que tem a agência da cabeça como forma de interação com o local remoto. A metodologia utilizada no trabalho consiste na pesquisa de tecnologias relacionadas, estudo de viabilidade para implementação e desenvolvimento da solução. O trabalho também explora os possíveis casos de uso da proposta, os mais evidentes sendo entretenimento, educação e monitoramento. Adicionalmente, os resultados obtidos são apresentados em função de métricas vinculadas aos requisitos do projeto, que incluem desempenho da solução e grau de imersão para os usuários finais.

Palavras-chave: *streaming*. imersão. realidade virtual.

Abstract

Immersive tour defines the experience that an individual has by observing a remote location in real-time and utilizing the virtual reality technology. Such experience requires that a second individual be present at the remote location carrying a video camera capable of transmitting, in real-time, an image of the surrounding environment to the observer. The goal of this work is to propose a system that implements the concept of immersive tour by using technologies currently available. This implementation differs from similar products regarding the level of immersion experienced by the observer, using head agency as a medium of interaction with the remote location. The methodology applied on this work consists on the research of related technologies, viability study of the implementation and development of the solution. This work also explores the possible use cases of the proposed framework, the most evident being entertainment, education and monitoring. Additionally, the achieved results are presented as a function of metrics bound to the project requirements, including performance of the system and level of immersion to the final user.

Keywords: streaming. immersion. virtual reality.

Lista de ilustrações

Figura 1 – Exemplos de *rickshaw*. 10
Figura 2 – Ilustração dos dois indivíduos necessários ao projeto: *virtual rickshaw* e observador. 11
Figura 3 – a. Visões da mesma cena pelos dois olhos e b. Superposição das imagens e a disparidade na retina. 15
Figura 4 – Estereoscópio. 15
Figura 5 – Esquema de funcionamento do estereoscópio. 16
Figura 6 – Esquema de funcionamento de um Anáglifo. 16
Figura 7 – Esquema de funcionamento de um óculos de polarização. 17
Figura 8 – Tipos básicos de paralaxe: a) Paralaxe zero; b) Paralaxe Negativa; c) Paralaxe Positiva. 18
Figura 9 – Efeito da paralaxe de acordo com a distância do observador à tela. . . 18
Figura 10 – Intervalo de controle do ângulo de paralaxe. 19
Figura 11 – Foto promocional de um protótipo do Sensorama. 22
Figura 12 – Piloto usando o capacete do projeto "Super Cockpit" de Tom Furness. 22
Figura 13 – Visão do usuário do "Super Cockpit". 23
Figura 14 – Modelo de referência TCP/IP. 28
Figura 15 – Exemplo de um arquivo de metadados m3u8. 29
Figura 16 – Especificação técnica para o corpo do Cardboard. 31
Figura 17 – Especificação técnica para a lente do Cardboard. 31
Figura 18 – Google Cardboard montado. 32
Figura 19 – Estrutura Analítica do Projeto. 38
Figura 20 – Arquitetura do sistema. 39
Figura 21 – Sequência de operações realizadas entre a gravação e reprodução do vídeo. 39
Figura 22 – Diagrama de classes do aplicativo cliente. 41
Figura 23 – Tela do ambiente de desenvolvimento Android Studio. 42
Figura 24 – Ciclo de vida de uma instância da classe `MediaPlayer`. 43
Figura 25 – Ciclo de vida de uma instância da classe `MediaPlayer`. 46
Figura 26 – Latência ideal 48
Figura 27 – Latência aproximada 48

Lista de tabelas

Tabela 1 – Comparação entre as ferramentas de aquisição FFmpeg e OpenCV . . 27
Tabela 2 – Testes de Latência 49
Tabela 3 – Testes de Recursos 49
Tabela 4 – Testes de Recursos 50
Tabela 5 – Testes de Imersão 50

Sumário

1 INTRODUÇÃO 10
1.1 Objetivo 11
1.2 Justificativa 11

I PESQUISA 13

2 CONCEITOS TEÓRICOS 14
2.1 Estereoscopia 14
2.2 Paralaxe 17
2.3 Imersão 19
2.4 Presença 20
2.5 Realidade Virtual 21

3 METODOLOGIA 24
3.1 Conceitos e Tecnologias Relacionadas 24
3.2 Estudo de Viabilidade 24
3.2.1 Aquisição de Imagem 25
3.2.1.1 OpenCV 25
3.2.1.2 FFmpeg 26
3.2.2 Transmissão 27
3.2.2.1 RTSP 28
3.2.2.2 HTTP Live Streaming 28
3.2.3 Reprodução 30
3.2.3.1 Android 30
3.2.3.2 Google Cardboard 30
3.2.3.3 Cardboard SDK 32

II DESENVOLVIMENTO 33

4 INTRODUÇÃO 34

5 ESPECIFICAÇÃO 35
5.1 Requisitos 35
5.1.1 Aplicação Cliente 35
5.1.1.1 Reprodução por *Streaming* 35

5.1.1.1.1 Descrição . . . 35
5.1.1.1.2 Requisitos . . . 35
5.1.2 Servidor . . . 36
5.1.2.1 Aquisição do vídeo . . . 36
5.1.2.1.1 Descrição . . . 36
5.1.2.1.2 Requisitos . . . 36
5.1.2.2 Transmissão . . . 37
5.1.2.2.1 Descrição . . . 37
5.1.2.2.2 Requisitos . . . 37
5.2 Planejamento . . . 37
5.2.1 Estrutura Analítica de Projeto . . . 37
5.3 Modelagem . . . 38
5.3.1 *Rickshaw* . . . 39
5.3.2 Servidor . . . 40
5.3.3 Aplicativo . . . 40
5.4 Construção . . . 40
5.4.1 Aplicativo . . . 40
5.4.1.1 Android Studio . . . 41
5.4.1.2 Classe `MediaPlayer` . . . 42
5.4.1.3 Rajawali . . . 43
5.4.2 Servidor HTTP . . . 44
5.4.3 Aquisição . . . 44
5.4.3.1 Aquisição Monoscópica de Imagem . . . 44
5.4.3.2 Aquisição Estereoscópica de Imagem . . . 46
5.5 Plano de Testes . . . 47
5.5.1 Testes Objetivos . . . 47
5.5.1.1 Latência . . . 48
5.5.1.2 Recursos Físicos . . . 49
5.5.2 Testes Subjetivos . . . 49

6 DISCUSSÃO . . . 51
6.1 Sistema em Tempo-Real . . . 51
6.2 Recursos . . . 51
6.3 Imersão . . . 52

7 CONCLUSÃO . . . 53

REFERÊNCIAS . . . 54

1 Introdução

Esse projeto apresenta uma proposta de infraestrutura que implementa a ideia de passeio imersivo. Passeio imersivo pode ser definido como a experiência de um indivíduo observar um local à distância em tempo real, utilizando tecnologias de imersão e realidade virtual. Tal experiência exige que um segundo indivíduo esteja presente no local portando um dispositivo capaz de transmitir em tempo real a imagem do ambiente ao seu redor para o observador.

O termo que utilizaremos para definir o indivíduo transmitindo a imagem do local remoto é *virtual rickshaw*. *Rickshaw* (WIKIPEDIA, 2016) é uma palavra de origem japonesa, literalmente traduzida como "veículo movido por humanos" e era o nome dado a carrinhos puxados e guiados por um trabalhador e que acomodavam um passageiro, como mostra a Figura 1. O termo foi escolhido dado que o indivíduo que carrega o dispositivo de aquisição de imagem, ou *virtual rickshaw*, proporciona ao observador uma imersão contemplativa do ambiente em que o primeiro se encontra, de tal forma que, mesmo não havendo interação do observador com o ambiente, este o observa como se lá estivesse.

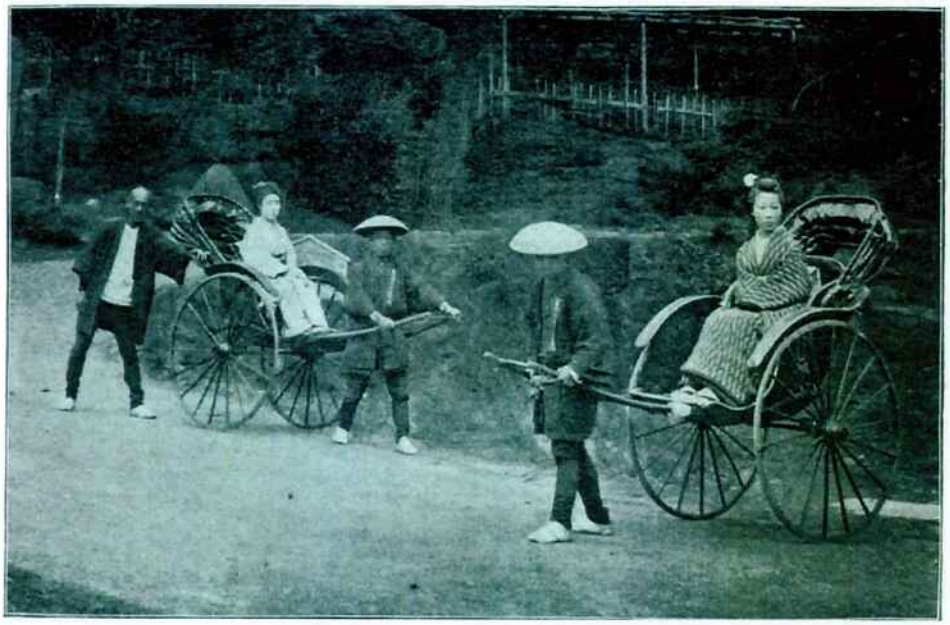

Figura 1 – Exemplos de *rickshaw*.

Fonte: Wikipedia (2016)

Usando tecnologias de imersão virtual e de comunicação de dados disponíveis atualmente, é possível proporcionar um passeio imersivo a um usuário. No escopo deste

projeto, o *virtual rickshaw*, portando uma câmera panorâmica (180), irá transmitir imagens em tempo real para um segundo indivíduo, o observador. Este utilizará alguma das tecnologias de imersão, como OculusRift ou Google Cardboard. A Figura 2 ilustra o passeio imersivo.

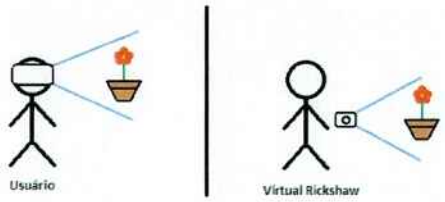

Figura 2 – Ilustração dos dois indivíduos necessários ao projeto: *virtual rickshaw* e observador.

Fonte: Autores

Dessa forma, é possível fazer com que o observador sinta-se imerso no ambiente em que o *virtual rickshaw* encontra-se fisicamente. Apesar da locomoção não ser possível, o usuário pode movimentar livremente a cabeça a fim de reconhecer completamente o ambiente em que foi inserido.

Utilizando protocolos de comunicação para *streaming*, um dos requisitos do projeto é realizar a transmissão em tempo real. Ao evitar atrasos na transmissão da imagem, a experiência de imersão para o usuário será maior.

1.1 Objetivo

Desenvolver um ambiente imersivo utilizando tecnologia de realidade virtual, de forma que um indivíduo à distância e em tempo real possa realizar um passeio virtual com agência da cabeça.

1.2 Justificativa

Com o crescente aumento comercial de vídeos 360 e com o barateamento e o aumento da qualidade da tecnologia de Realidade Virtual (RV), espera-se uma grande disseminação do uso de ambientes virtuais imersivos no setor educacional (JOHNSON et al., 2016). O desenvolvimento de uma ferramenta que permita um passeio virtual imersivo poderia ajudar, por exemplo, aulas de História a proporcionar aos alunos uma viagem para um museu distante sem sair da sala de aula. Portanto, a contribuição deste trabalho é desenvolver um ambiente virtual imersivo, possibilitando que indivíduos à distância e

em tempo real possam realizar um passeio virtual com total agência da cabeça. Ainda que haja evidências e seja de se esperar que, quanto maior a imersão do indivíduo maior será a percepção de presença e envolvimento com o ambiente de RV (ROBERTSON; CARD; MACKINLAY, 1993), é importante ressaltar que o foco deste trabalho não inclui avaliar percepções de presença e que futuros trabalhos possam se beneficiar nesse sentido.

Parte I

Pesquisa

2 Conceitos Teóricos

As seções abaixo descrevem o estudo realizado para o entendimento das tecnologias e conceitos envolvidos no trabalho. Os itens seção 2.1 e seção 2.2 foram integralmente baseados em (TORI; KIRNER; SISCOUTTO, 2006), enquanto que os itens seção 2.3, seção 2.4 e seção 2.5 foram baseadas em múltiplos autores.

2.1 Estereoscopia

A visão estereoscópica é uma característica do sistema visual humano que possibilita a visualização tridimensional do ambiente a partir de imagens bidimensionais captadas pelas retinas. O ser humano é dotado de dois olhos que enxergam o mundo de forma diferente, pois estão separados por uma distância. Desta forma diferentes imagens são geradas pelo olho direito e pelo olho esquerdo que, processadas pelo cérebro, dão noção de profundidade dando a ideia de imersão em um ambiente com objetos posicionados a distâncias diferentes. A estereoscopia diz respeito à superposição dessas duas imagens de um mesmo objeto, possibilitando assim a visualização tridimensional do objeto pelo cérebro humano. A estereoscopia é uma reprodução artificial da visão binocular presente no ser humano, responsável por possibilitar a percepção de profundidade pelo cérebro.

A estereoscopia está relacionada à capacidade de enxergar em três dimensões, isto é, de perceber a profundidade. O princípio de funcionamento da maioria dos dispositivos estereoscópicos é o oferecimento de imagens distintas aos olhos esquerdo e direito do observador, proporcionando sensação de profundidade tal qual quando se observa um objeto real.

Algumas das principais técnicas de estereoscopia são:

- Vídeo Estereoscópico

  A base para a percepção estereoscópica é a disparidade binocular do sistema visual humano, que gera duas imagens ligeiramente diferentes quando uma cena é projetada nas retinas dos olhos (Figura 3 a)). As duas perspectivas diferentes das imagens são fundidas no córtex visual do cérebro, de forma a compor uma simples visão estereoscópica (tridimensional) (Figura 3 b)).

  Esse processo pode ser simulado através de duas câmeras organizadas com a mesma distância interocular dos olhos humanos. Logo, colocando-se as câmeras separadas uma da outra com base nessa distância, simula-se o sistema visual humano. Quando cada imagem das câmeras for apresentada ao seu olho correspondente, as duas

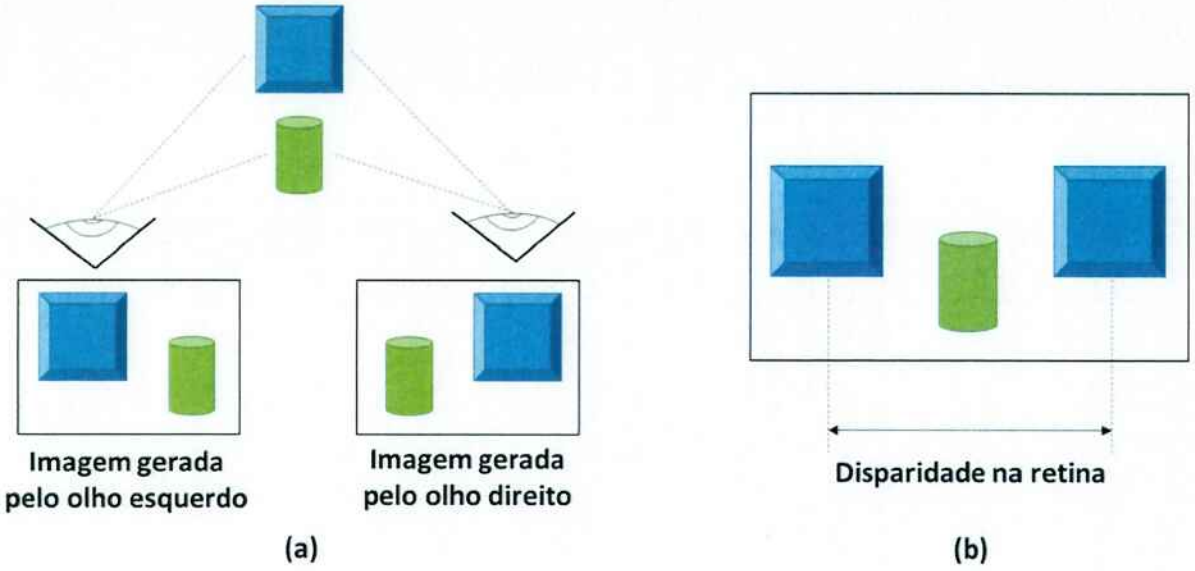

Figura 3 – a. Visões da mesma cena pelos dois olhos e b. Superposição das imagens e a disparidade na retina.

Fonte: (TORI; KIRNER; SISCOUTTO, 2006)

imagens serão fundidas em uma única imagem pelo cérebro, produzindo a ilusão de visão estereoscópica.

- Estereoscópio

  O estereoscópio é um instrumento composto por lentes que direcionam uma das imagens do par estereoscópico para o olho direito e a outra para o olho esquerdo, permitindo visualizar-se a imagem de forma tridimensional. A Figura 4 traz o exemplo de um aparelho estereoscópico. Ele separa fisicamente as visões esquerda e direita, eliminando a possibilidade do cruzamento entre as visões.

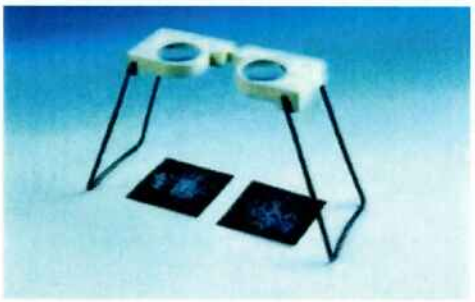

Figura 4 – Estereoscópio.

Fonte: (TORI; KIRNER; SISCOUTTO, 2006)

Uma das grandes vantagens desse tipo de aparelho é permitir que o observador ajuste a distância pupilar entre as lentes, bem como ajuste a distância de visualização. Seu esquema básico pode ser observado na Figura 5.

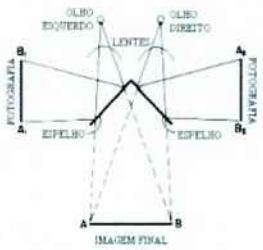

Figura 5 – Esquema de funcionamento do estereoscópio.

Fonte: (TORI; KIRNER; SISCOUTTO, 2006)

- Anáglifo

  O Anáglifo funciona à base de figuras planas cujo relevo se obtém por cores complementares, normalmente vermelho e verde ou vermelho e azul esverdeado (Figura 6). Nesse caso, cada um dos olhos utilizará um filtro diferente, feito de papel celofane, para visualizar as imagens do par estereoscópico.

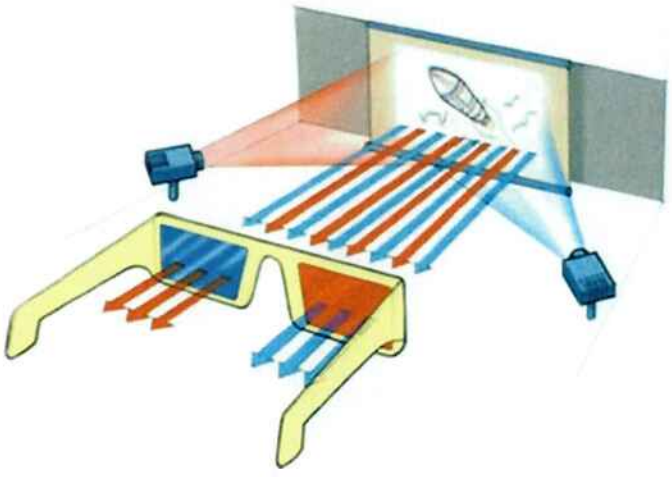

Figura 6 – Esquema de funcionamento de um Anáglifo.

Fonte: (TORI; KIRNER; SISCOUTTO, 2006)

- Polarização da Luz

  No 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 (por exemplo, um plano vertical e um horizontal). Dessa forma, o observador utiliza filtros polarizadores ortogonais correspondentes aos planos de projeção e vê com cada olho apenas uma das imagens projetadas (Figura 7). Da fusão das imagens vistas por cada olho, resultará a visão estereoscópica.

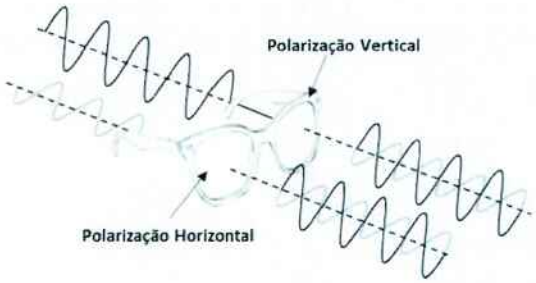

Figura 7 – Esquema de funcionamento de um óculos de polarização.

Fonte: (TORI; KIRNER; SISCOUTTO, 2006)

2.2 Paralaxe

Conforme citado anteriormente, existem diferenças entre imagens formadas nas retinas de cada olho quando sobrepostas. Estas diferenças são na direção horizontal, pois é a distância de separação dos olhos. A disparidade é zero para objetos onde os olhos convergem. Já a paralaxe é a distância entre os pontos correspondentes das imagens do olho direito e do esquerdo na imagem projetada na tela. Em outras palavras, disparidade e paralaxe são duas entidades similares, com a diferença que paralaxe é medida na tela do computador e disparidade, na retina. É a paralaxe que produz a disparidade, que por sua vez, produz o estéreo. A paralaxe possui três tipos básicos:

- Paralaxe zero

  Conhecida como ZPS (do inglês Zero Paralax Setting). Um ponto com paralaxe zero se encontra no plano de projeção, tendo a mesma projeção para os dois olhos (Figura 8 (a)).

- Paralaxe negativa

  Significa que o cruzamento dos raios de projeção para cada olho encontra-se entre os olhos e a tela de projeção, dando a sensação de o objeto estar saindo da tela (Figura 8 (b)).

- Paralaxe positiva

  O cruzamento dos raios é atrás do plano de projeção, dando a sensação de que o objeto está atrás da tela de projeção (Figura 8 (c)).

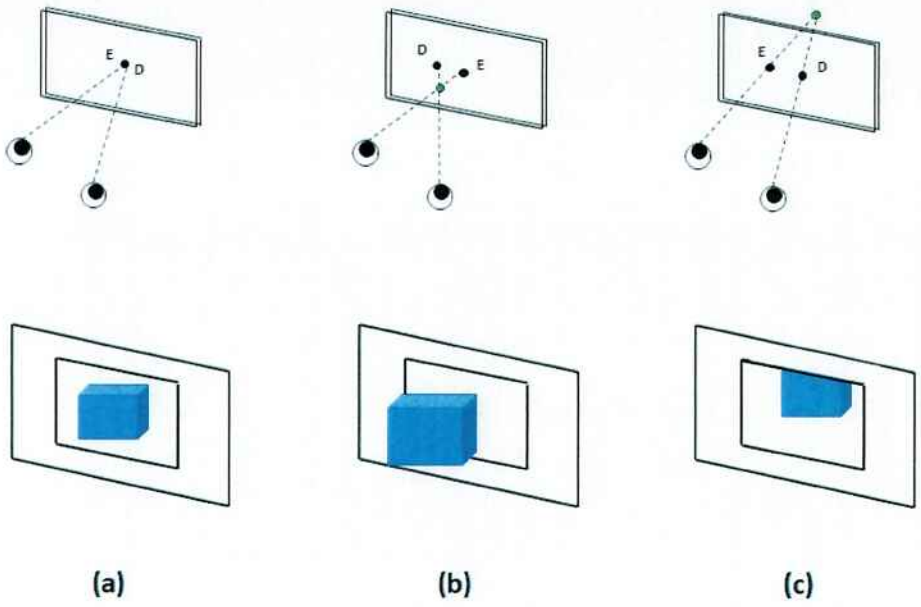

Figura 8 – Tipos básicos de paralaxe: a) Paralaxe zero; b) Paralaxe Negativa; c) Paralaxe Positiva.

Fonte: (TORI; KIRNER; SISCOUTTO, 2006)

Um fator importante que deve ser levado em consideração é que a distância do observador à tela afeta o efeito de estereoscopia. Quanto maior a distância à tela, maior será o efeito estereoscópico (tanto positivo quanto negativo). Na Figura 9 é ilustrado o caso para paralaxe positiva. Note que para o observador 2, que está mais distante da tela, o ponto também se encontra mais distante e para dentro, comparado ao observador 1.

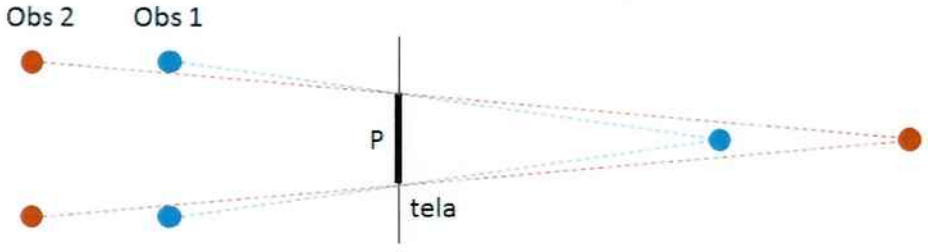

Figura 9 – Efeito da paralaxe de acordo com a distância do observador à tela.

Fonte: (TORI; KIRNER; SISCOUTTO, 2006)

Um grande desafio da estereoscopia é gerar maior efeito de profundidade com menor valor de paralaxe devido ao espaço físico limitado da tela e distância máxima que um ambiente comporta para os observadores. Em regra geral, o ângulo de paralaxe β

deve estar no intervalo [-1,5, 1,5], definindo paralaxes mínimas e máximas. O esquema de controle da paralaxe é ilustrado na Figura 10, onde d é a distância do observador à tela.

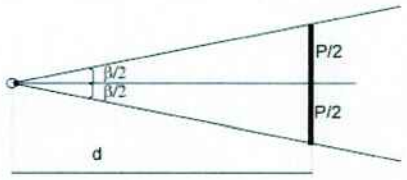

Figura 10 – Intervalo de controle do ângulo de paralaxe.

Fonte: (TORI; KIRNER; SISCOUTTO, 2006)

A distância interaxial também influencia a paralaxe. Quanto maior a distância interaxial, maior é a paralaxe e, consequentemente, maior a sensação de estéreo.

2.3 Imersão

Imersão se refere à sensação real de estar em um local virtual que o usuário tem ao utilizar um dispositivo de realidade virtual, sendo capaz de manipular objetos ali presentes como se eles fossem reais devido à resposta destes às interações realizadas pelo usuário (e.g. deformação, quebra, etc.).

A imersão possui diferentes níveis, dependendo dos estímulos extras que são projetados no usuário através de seus sentidos, sendo o principal a visão. De modo geral e do ponto de vista da visualização, a Realidade Virtual (RV) imersiva utiliza capacete ou cavernas, enquanto a RV não imersiva utiliza monitores. Entretanto, dispositivos baseados nos demais sentidos podem introduzir algum grau de imersão à RV que usa monitores (ROBERTSON; CARD; MACKINLAY, 1993). Os monitores ainda apresentam alguns pontos positivos, como o baixo custo e a facilidade de uso, evitando as limitações técnicas e problemas decorrentes do uso do capacete. Porém, a tendência deve ser a utilização da RV imersiva na grande maioria das aplicações futuras. A interação está ligada à capacidade do computador detectar as entradas do usuário e modificar instantaneamente o mundo virtual em função das ações efetuadas sobre ele (capacidade reativa). As pessoas são cativadas por uma boa simulação em que as cenas mudam em resposta aos seus comandos, que é característica mais marcante dos vídeo games. Para que um sistema de RV pareça mais realista, o ambiente virtual inclui objetos simulados. Outros 11 artifícios para aumentar o realismo são empregados, por exemplo, a texturização dos objetos do ambiente e a inserção de sons tanto ambientais quanto sons associados a objetos específicos (ARAUJO, 1996). A ideia de envolvimento, por sua vez, está ligada ao grau de motivação para o engajamento de uma pessoa em determinada atividade. O envolvimento pode ser passivo,

como ler um livro ou assistir televisão, ou ativo, ao participar de um jogo com algum parceiro. A RV tem potencial para os dois tipos de envolvimento ao permitir a exploração de um ambiente virtual e propiciar a interação do usuário com o mundo virtual dinâmico. Embora a percepção visual seja nosso sentido primário, outros sentidos também devem ser estimulados para proporcionar uma completa imersão, entre os quais o retorno auditivo, o tato e a força de reação.

2.4 Presença

Neste projeto entende-se por presença a imersão de outros indivíduos em um cenário em que o usuário se encontra. Desta forma o usuário tem a impressão da presença de outras pessoas no local em que se encontra por meio de técnicas holográficas.

As aplicações de RV, em geral, são classificadas da seguinte forma: tele-colaboração, tele-presença, visualização científica, visualização de dados 3D e outros. Na tele-colaboração, usuários remotos compartilham um ambiente virtual para realizar uma tarefa em comum. Protótipos de sistemas de tele-colaboração implementados permitem aos usuários compartilhar um mesmo espaço e manipular objetos, sentindo o peso dos mesmos por meio de dispositivos de feedback de força (ARAUJO, 1996).

Um sistema de tele-presença, ou tele-existência, estende as capacidades motoras e sensoriais de um operador humano, bem como a suas habilidades de solução de problemas, para um ambiente remoto. Na tele-presença, também referenciada como tele-operação ou tele-robótica, o robô que executa as tarefas está fisicamente separado de seu operador humano. As ações executadas pelo operador são traduzidas em ações executadas pelo robô em seu ambiente remoto, ao mesmo tempo em que é emitido feedback sensorial ao operador humano, que se sente como se estivesse realmente presente no ambiente remoto (ARAUJO, 1996). A tele-presença pode ser mais claramente vista como uma técnica de visão que realça a função intermediária entre o participante e o ambiente (LATTA; OBERG, 1994).

A Visualização Científica permite que enormes quantidades de dados gerados por simulações computacionais sejam mapeados em representações visuais 3D. Dados podem ser representados como pontos, linhas, curvas, superfícies, volumes, cores, e mesmo como sons. Também podem ser manipulados e observados de vários ângulos e posições, permitindo uma ampla exploração das propriedades globais de soluções numéricas. Uma aplicação relacionada que também envolve sistemas complexos e grandes volumes de dados é denominada visualização de informação, ou visualização de dados 3D. É o caso da visualização de software, cujo objetivo é o facilitar o desenvolvimento de sistemas altamente complexos e de grande porte, por exemplo, para gerenciamento de redes de telecomunicações, controle de tráfego aéreo, gerenciamento de linhas metroviárias e ferroviárias (NETTO; MACHADO; OLIVEIRA, 2002).

2.5 Realidade Virtual

A Realidade Virtual (RV) é, antes de tudo, uma "interface avançada do usuário" para acessar aplicações executadas no computador, tendo como características a visualização de, e movimentação em, ambientes tridimensionais em tempo real e a interação com elementos desse ambiente. Além da visualização em si, a experiência do usuário de RV pode ser enriquecida pela estimulação dos demais sentidos como tato e audição (TORI; KIRNER; SISCOUTTO, 2006).

Por volta de 70% dos receptores do sentido humano encontram-se nos olhos, tornando-os os grandes "monopolistas dos sentidos" (JACOBSON, 1994). A maioria das informações recebidas pelo ser humano tem a forma de imagens visuais, as quais são interpretadas pelo cérebro. Os computadores digitais interpretam informações fornecidas por algum dispositivo de entrada de dados, como um teclado, por exemplo. Atualmente, a RV permite que computadores e mente humana atuem de forma cada vez mais integrada (MACHADO, 1995). O termo Realidade Virtual é creditado a Jaron Lanier, fundador da VPL Research Inc., que o cunhou, no início dos anos 80, para diferenciar as simulações tradicionais feitas por computador de simulações envolvendo múltiplos usuários em um ambiente compartilhado (ARAUJO, 1996).

Na prática, a RV permite que o usuário navegue e observe um mundo tridimensional, em tempo real e com seis graus de liberdade (6DOF). Isso exige a capacidade do software de definir, e a capacidade do hardware de reconhecer, seis tipos de movimento: para frente/para trás, acima/abaixo, esquerda/direita, inclinação para cima/para baixo, angulação à esquerda/à direita e rotação à esquerda/à direita. Na essência, a RV é um "espelho" da realidade física, na qual o indivíduo existe em três dimensões e tem a sensação do tempo real e a capacidade de interagir com o mundo ao seu redor (NETTO; MACHADO; OLIVEIRA, 2002).

A RV começou na indústria de simulação com os simuladores de voo que a força aérea do Estados Unidos passou a construir logo após a Segunda Guerra Mundial (JACOBSON, 1994). A indústria de entretenimento também teve um papel importante, ao construir um simulador chamado Sensorama (Figura 11). O Sensorama era uma espécie de cabine que combinava filmes 3D, som estéreo, vibrações mecânicas, aromas, e ar movimentado por ventiladores, tudo isso para proporcionar ao espectador uma viagem multisensorial (PIMENTEL, 1995). Patenteado em 1962 por Morton Heilig, o equipamento já utilizava um dispositivo para visão estereoscópica.

Em 1982, Thomas Furness demonstrava para a Força Aérea Americana o VCASS (Visually Coupled Airborne Systems Simulator), conhecido como "Super Cockpit". Trata-se de um simulador que usava computadores e vídeo-capacetes interligados para representar o espaço 3D da cabine de um avião (Figura 12 e Figura 13). Os vídeo-capacetes integravam

Figura 11 – Foto promocional de um protótipo do Sensorama.

Fonte: (PIMENTEL, 1995)

as componentes de áudio e vídeo. Assim. os pilotos podiam aprender a voar e lutar em trajetórias com 6 graus de liberdade (6DOF) sem decolar verdadeiramente. O VCASS possuía alta qualidade de resolução nas imagens e era bastante rápido na atualização de imagens complexas. No entanto, o custo representava um problema: milhões de dólares eram necessários apenas para o capacete (PIMENTEL, 1995).

Figura 12 – Piloto usando o capacete do projeto "Super Cockpit" de Tom Furness.

Fonte: (PIMENTEL, 1995)

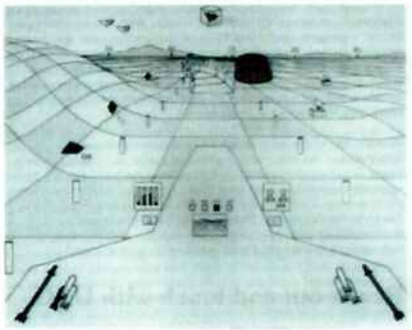

Figura 13 – Visão do usuário do "Super Cockpit".

Fonte: (PIMENTEL, 1995)

3 Metodologia

A metodologia utilizada para a fase de pesquisa do projeto utiliza o método empírico e divide-se em duas fases principais: a) pesquisa de conceitos e tecnologias relacionadas e b) estudo de viabilidade. As duas fases são descritas em detalhes a seguir. Em relação à fase de desenvolvimento do projeto, foi utilizada a metodologia em Cascata descrita em (PRESSMAN, 2009). Tal metodologia divide o desenvolvimento em cinco etapas: levantamento de requisitos, planejamento, modelagem, construção e plano de testes. Tais etapas são descritas em mais detalhes no Capítulo 5.

3.1 Conceitos e Tecnologias Relacionadas

Essa fase foi a primeira fase do projeto e consistiu em uma análise do estado da arte das principais tecnologias e conceitos envolvidos na proposta, discriminados pelos orientadores do projeto. As pesquisas de tópicos conceituais foram realizadas principalmente através de livros texto indicados pelos orientadores, relacionados majoritariamente a tópicos de novas mídias digitais, como realidade virtual e imersão.

As pesquisas referentes às tecnologias que foram utilizadas no projeto foram feitas principalmente através de recursos *online*. Porém, por tais tecnologias serem de uso comercial e portanto não haver detalhes técnicos de implementação, os artigos e páginas consultados fornecem apenas uma visão geral das técnicas utilizadas por cada uma delas, como é o caso do aplicativo Periscope[1] para *smartphones*.

Esta fase também forneceu o ponto de partida para as pesquisas relacionadas às tecnologias que seriam utilizadas na implementação do projeto e nos testes de viabilidade, através de citações e sugestões de algoritmos, ferramentas e técnicas em artigos e fóruns de discussão.

3.2 Estudo de Viabilidade

Nesta fase foram realizados testes com diversas ferramentas para viabilizar o desenvolvimento do projeto. A maior dificuldade encontrada foi na transmissão do vídeo entre o *desktop* e o *smartphone* devido a problemas de compatibilidade das linguagens utilizadas e das limitações de funções fornecidas pelas APIs das respectivas plataformas.

Os resultados dos testes dessa fase foram determinantes nas tecnologias utilizadas para o desenvolvimento efetivo do projeto, como bibliotecas, protocolos de comunicação e

[1] Aplicativo para Android disponível em https://play.google.com/store/apps/details?id=tv.periscope.android

ferramentas. Essa fase também forneceu uma visão geral da arquitetura que seria utilizada no sistema em etapas posteriores.

Abaixo são descritas as tecnologias pesquisadas e testadas para o estudo de viabilidade. São apresentadas as características de cada tecnologia e também os fatores que levaram à decisão de utilizá-la ou não para o desenvolvimento efetivo do projeto. As tecnologias estão organizadas de acordo com os módulos do projeto: aquisição de imagem, transmissão e reprodução.

3.2.1 Aquisição de Imagem

Neste módulo são realizadas a aquisição de imagem pela câmera USB e as etapas necessárias para obtenção de um arquivo de mídia reproduzível em diferentes plataformas e que atenda às especificações do projeto. Tais etapas envolvem a codificação da imagem, configuração ideal da resolução, taxa de amostragem, formato de saída e demais operações usuais de processamento de vídeo.

Na avaliação de cada tecnologia foram levados em conta diversos fatores para a seleção da mais viável para o desenvolvimento. Além dos parâmetros relativos ao processamento do vídeo, também foram considerados fatores como facilidade de uso da ferramenta, flexibilidade de configuração de parâmetros e na integração com outras ferramentas que eventualmente se mostrassem necessárias, licença de uso e desempenho. Outro fator também importante, mas que não está necessariamente fixado nesta etapa é a correção da distorção gerada pela lente *fisheye* da câmera.

Nesta etapa, avaliou-se a biblioteca OpenCV e o *framework* FFmpeg, ambos descritos a seguir. No final da seção é realizada uma comparação geral entre as ferramentas e a discriminação da selecionada para o desenvolvimento.

3.2.1.1 OpenCV

OpenCV (OpenCV, 2016b) (*Open Source Computer Vision Library*) é uma biblioteca *open source* de visão computacional e aprendizado de máquina. Sendo amplamente conhecida para o desenvolvimento de aplicações que utilizam processamento de imagem e vídeo, a biblioteca foi a primeira a ser estudada como candidata à aquisição de imagem pela câmera. Um grande vantagem do OpenCV é possuir interfaces para mais de uma linguagem, sendo elas C++, C, Python e Java. Porém, apesar da grande portabilidade, há certas limitações na API para as linguagens Python e Java.

Quanto à aquisição propriamente dita, a implementação da operação é bastante simples, sendo necessárias apenas algumas linhas de código. A grande desvantagem nesta etapa foi a incapacidade de encontrar, na documentação oficial ou na *web*, uma forma acessível de ajustar os parâmetros da aquisição como resolução, *codec* e formato de saída.

Os *frames* que são lidos da câmera possuem formato JPEG e relativa baixa resolução, além do fato de que a aquisição era subitamente interrompida por motivos desconhecidos e em momentos aleatórios.

Por possuir uma ampla gama de algoritmos otimizados, foi suposto que também forneceria algum algoritmo para correção da distorção da lente *fisheye*. A suposição foi validada, e na documentação oficial (OpenCV, 2016c)(OpenCV, 2016a) foi possível encontrar as classes e métodos necessários para a correção das distorções, porém não foi encontrado qual o algoritmo utilizado para correção e nem informações quanto ao seu desempenho.

A biblioteca fornece uma documentação oficial em seu *website*, bem detalhada e acessível. Esta, porém, só está disponível para a linguagem C++, enquanto que para as outras linguagens é possível encontrar referências não oficiais. Apesar disso, no *website* há tutoriais para a implementação de diversas funcionalidades para todas as linguagens e que auxiliam o usuário iniciante.

Finalmente, outro fator importante nesta etapa é o suporte que a biblioteca deveria fornecer para a posterior transmissão do vídeo obtido. Como já estava assumida a necessidade de um protocolo específico para aplicações em tempo-real que implementasse o protocolo de transporte RTP (Real-time Transport Protocol) (SCHULZRINNE et al., 2003), o OpenCV foi descartado como alternativa por não possuir uma interface para este protocolo.

3.2.1.2 FFmpeg

FFmpeg (FFMPEG, 2016a) é um *framework* multimídia capaz de realizar todas as operações de aquisição, gravação e reprodução de mídia. FFmpeg fornece um grande número de bibliotecas, cada uma com funções específicas, e que juntas dão uma gama completa de funcionalidades para multimídia. Ao contrário do OpenCV, porém, todas as bibliotecas são implementadas em C++. Apesar disso, é possível portar o *framework* para um grande número de plataformas com diferentes arquiteturas.

Na etapa de aquisição, o FFmpeg torna extremamente simples a operação de aquisição e gravação e possui uma grande flexibilidade na configuração de parâmetros como *codec*, resolução, formato de saída e transmissão do vídeo. A variedade de formatos para codificação é uma grande vantagem, pois fornece flexibilidade para utilizar o vídeo gerado em diferentes plataformas, problema que seria encontrado na etapa de reprodução do vídeo no *smartphone*. Quanto à correção da distorção da lente, o FFmpeg não dá esse suporte pelo fato de não ser um *framework* voltado ao processamento de vídeos. Caso fosse utilizado, portanto, a responsabilidade de corrigir essa distorção teria de ser delegada a outro módulo em etapas posteriores.

Em relação à documentação, no *website* é possível encontrar uma extensa documentação do *framework* (FFMPEG, 2016b). A documentação se divide tanto para as ferramentas fornecidas assim como para a API das bibliotecas, todas muito bem detalhadas.

Quanto ao suporte à transmissão do vídeo, fator importante na seleção da tecnologia, o FFmpeg torna trivial a transmissão do vídeo adquirido por protocolos conhecidos de transporte de dados. O *framework* possibilita a transmissão por protocolos que implementam o RTP como o RTSP (SCHULZRINNE; RAO; LANPHIER, 1998) e o RTMP (THORNBURGH, 2014) e a protocolos proprietários como o HLS, além de poder configurar a transmissão usando tunelamento HTTP e também ajustar parâmetros como taxa de transmissão e número de *threads*. Além disso, também é possível transmitir o vídeo para um outro servidor e delegar a este a tarefa de retransmitir a outros clientes.

Comparando as duas tecnologias pesquisadas através da Tabela 1, optou-se pela utilização do FFmpeg, principalmente devido ao suporte que fornece para transmissão do vídeo. Na próxima seção serão discutidos os diferentes protocolos de comunicação considerados para o projeto, os problemas encontrados e a solução definida. A seguir é apresentada uma tabela com diferentes critérios de comparação entre o OpenCV e o FFmpeg.

Tabela 1 – Comparação entre as ferramentas de aquisição FFmpeg e OpenCV.

	OpenCV	**FFmpeg**
Multiplataforma	x	x
Multilinguagem	x	-
Ajuste de parâmetros do vídeo	-	x
Facilidade de geração do vídeo	-	x
Documentação	x	x
Correção de distorção	x	-
Transmissão do vídeo	-	x

Fonte: Autores

3.2.2 Transmissão

Após a aquisição do vídeo pela câmera utilizando o *framework* FFmpeg, a próxima tarefa no estudo de viabilidade é a pesquisa de tecnologias para transmissão do vídeo capturado para o dispositivo móvel. As principais fontes de pesquisa foram as RFCs dos protocolos considerados, apoiando-se nas documentações do *framework* e das APIs das tecnologias candidatas para a reprodução do vídeo no *smartphone*, de modo a garantir a compatibilidade entre as soluções e a integração entre os módulos do sistema.

Como já mencionado, o *framework* FFmpeg fornece suporte à transmissão dos vídeos capturados através dos protocolos mais utilizados para *streaming*, entre eles RTSP, HLS e RTMP. Neste módulo foram analisadas duas formas de transmissão: a primeira,

utilizada em uma versão preliminar do projeto, foi realizando a transmissão do vídeo utilizando o protocolo RTSP e a segunda forma estudada foi utilizando o protocolo HLS.

3.2.2.1 RTSP

RTSP (*Real Time Streaming Protocol*) (SCHULZRINNE; RAO; LANPHIER, 1998) (TANENBAUM, 2007) é um protocolo responsável por estabelecer e controlar *streams* de áudio ou vídeo de forma sincronizada e em tempo-real. O RTSP não estabelece conexões. Por atuar na camada de Aplicação no modelo TCP/IP, o RTSP pode utilizar quaisquer protocolos de transporte, sendo os mais comuns TCP (POSTEL, 1981) e UDP (POSTEL, 1980), como ilustra a Figura 14.

O transporte por TCP é o mais comum, por ser capaz de passar por *firewalls* utilizando tunelamento por HTTP com mais facilidade do que o protocolo UDP e por realizar controle de fluxo e de erros. Além disso, o protocolo TCP tenta armazenar em um *buffer* local o máximo de pacotes que conseguir no mínimo de tempo possível. Isso possibilita ao usuário assistir ao vídeo enquanto a aplicação descarrega a mídia (TANENBAUM, 2007). O uso de UDP é mais comum para cenários em que é realizado *multicast* de mídias. Ambos, porém, podem ser utilizados no projeto para a prova de conceito.

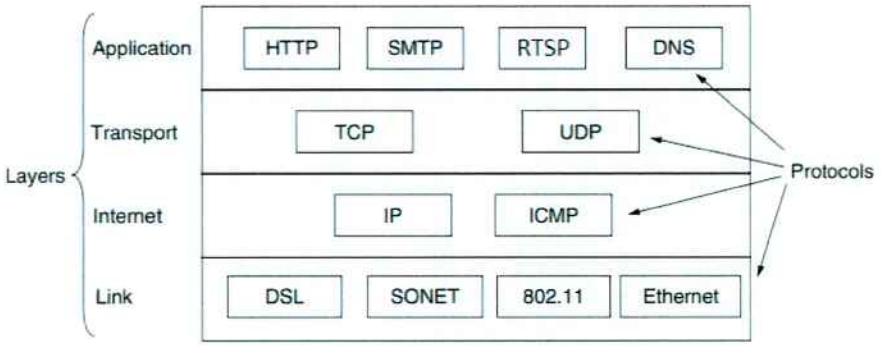

Figura 14 – Modelo de referência TCP/IP.

Fonte: (TANENBAUM, 2007) (Adaptado)

3.2.2.2 HTTP Live Streaming

HTTP (*Hyper Text Transfer Protocol*) (FIELDING; RESCHKE, 2014) é um protocolo que atua na camada de aplicação do modelo TCP/IP. Por possuir caráter genérico, o HTTP é utilizado para diversas tarefas além da distribuição de hipermídia, sendo facilmente estendido para outras aplicações.

O HTTP normalmente é implementado utilizando o protocolo TCP da camada de transporte. Dada sua característica genérica e seu crescente uso na Internet, este protocolo evoluiu para atuar como um protocolo de transporte para diferentes aplicações que utilizam um sistema de requisição-resposta (TANENBAUM, 2007), como por exemplo a transmissão de arquivos, aplicação utilizada no projeto.

Em 2009, foi desenvolvido o HLS (HTTP Live Streaming) (PANTOS; MAY, 2016) (Apple Inc., 2016) pela Apple Inc., um protocolo de comunicação multimídia que utiliza o HTTP como protocolo de transporte. O HLS utiliza um arquivo que contém informações sobre a lista de reprodução e metadados da mídia sendo transmitida. A lista de reprodução é composta de segmentos de pequenos vídeos. identificados por um nome definido no arquivo de metadados. O arquivo metadados, que possui a extensão m3u8, é o primeiro arquivo a ser requisitado pelo cliente, para que este identifique os nomes dos arquivos de vídeo e sua localização no servidor e possa então fazer a requisição destes. A Figura 15 apresenta um exemplo de um arquivo m3u8. O *framework* FFmpeg dá suporte à gravação de vídeos utilizando o protocolo HLS, o que facilitou para o teste de viabilidade.

Para a reprodução dos vídeos gravados, utilizou-se um servidor HTTP que armazena os vídeos numa máquina local e que posteriormente são solicitados pela aplicação no *smartphone*. Tal solução foi possibilitada pelo amplo suporte ao protocolo encontrado em diversas plataformas, incluindo na API disponível para a plataforma Android. O vídeo é gravado e armazenado diretamente em um diretório pertecente ao servidor HTTP, já estando acessível para requisição.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:9
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:8.333322,
out0.ts
#EXTINF:3.000000,
out1.ts
#EXTINF:3.000000,
out2.ts
#EXTINF:3.000000,
out3.ts
#EXT-X-ENDLIST
```

Figura 15 – Exemplo de um arquivo de metadados m3u8.

Fonte: Autores

3.2.3 Reprodução

Nesta seção é descrito o estudo realizado para reprodução de vídeos no *smartphone*. Os métodos estudados utilizam a plataforma Android, porém as bibliotecas estudadas também oferecem suporte para a plataforma iOS de *smartphones*, de forma que podem ser implementados para os dispositivos da Apple de forma análoga.

Foram estudados dois possíveis métodos para a reprodução dos vídeos, porém ambos utilizam, em última instância, a mesma biblioteca final disponibilizada pela SDK do Android. Um dos métodos estudados foi utilizando a conhecida plataforma de desenvolvimento de games Unity, que fornece um amplo suporte para desenvolvimento em ambientes *mobile*. A segunda opção foi utilizando o ambiente de desenvolvimento Android Studio, que utiliza a linguagem Java. Os dois ambientes de desenvolvimento possuem suporte para desenvolvimento de aplicativos para o Google Cardboard através da SDK disponibilizada pela Google. A SDK oferece APIs de desenvolvimento para Android, iOS e Unity, porém as classes disponibilizadas e funcionalidades diferem entre eles, sendo a mais completa para Android.

A seguir é descritos o ambiente de desenvolvimento utilizado, assim como é dada uma visão geral da SDK para o Google Cardboard e uma breve descrição sobre este *gadget*.

3.2.3.1 Android

Android é um sistema operacional para dispositivos móveis, sendo o SO mais utilizado para *smartphones* no mundo. O desenvolvimento de aplicativos para Android utiliza a SDK oficial e é auxiliado por uma vasta documentação disponibilizada *online* (Android, 2016c) (Android, 2016a), detalhando as diversas funcionalidades oferecidas pela plataforma.

Para o projeto, escolheu-se o Android no lugar de outros sistemas operacionais devido à ampla difusão de dispositivos que o utilizam, além de não ser necessário um ambiente restrito de desenvolvimento para a criação de aplicativos. O desenvolvimento de aplicativos utiliza a IDE Android Studio (Android, 2016b), baseado na IDE Eclipse e que oferece uma integração simples para execução e debug de código em emuladores ou dispositivos reais.

3.2.3.2 Google Cardboard

O Google Cardboard (Google Inc., 2016a) é um acessório que permite experimentar a tecnologia de realidade virtual de forma simples e acessível usando um *smartphone*. Utilizando materiais baratos como papelão para o corpo do acessório e acrílico para as lentes, o Google Cardboard pode ser adquirido facilmente em comparação a óculos de realidade virtual mais sofisticados. A Google fornece em seu *website* arquivos PDF contendo

as especificações técnicas para a construção do Cardboard, incluindo dimensões e materiais utilizados, como ilustrado na Figura 16 e Figura 17.

Após montado (Figura 18), basta inserir no Cardboard um *smartphone* que possui giroscópio e executar um dos muitos aplicativos desenvolvidos para realidade virtual. A seção a seguir descreve o suporte ao desenvolvimento de aplicativos para o Cardboard.

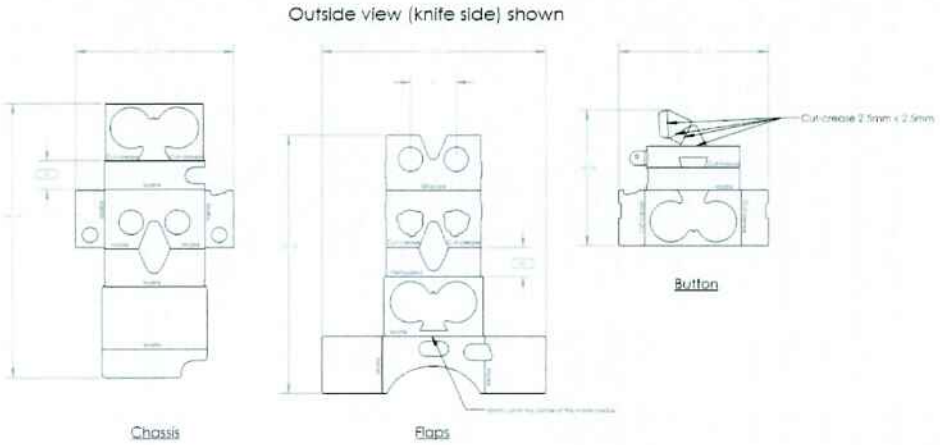

Figura 16 – Especificação técnica para o corpo do Cardboard.

Fonte: (Google Inc., 2016a)

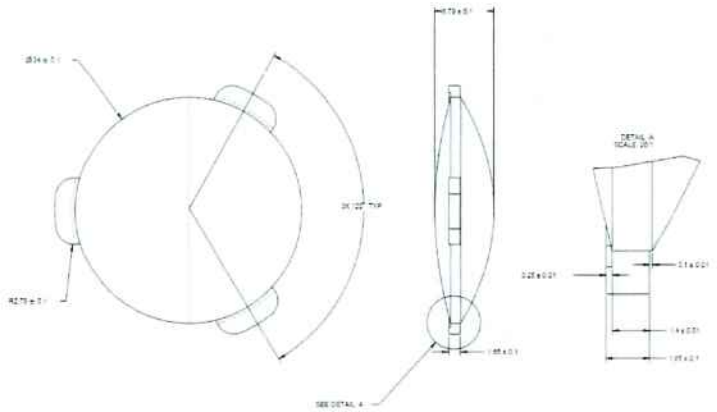

Figura 17 – Especificação técnica para a lente do Cardboard.

Fonte: (Google Inc., 2016a)

Figura 18 – Google Cardboard montado.

Fonte: (Google Inc., 2016a)

3.2.3.3 Cardboard SDK

O Cardboard SDK (Google Inc., 2016b) é uma biblioteca para desenvolvimento de aplicativos para as plataformas iOS e Android que utilizam realidade virtual. A SDK simplifica tarefas de desenvolvimento comuns no contexto de realidade virtual para que o desenvolvedor precise lidar apenas com a experiência de imersão ao usuário final. Tais tarefas envolvem correção de distorção de lentes, *head tracking*, interação com o usuário etc.

Uma das funcionalidades fornecidas pela SDK é a reprodução de vídeos panorâmicos, sejam locais ou remotos. No contexto deste projeto, tal funcionalidade desempenhou um fator chave no desenvolvimento, dada a dificuldade que seria enfrentada na implementação completa utilizando a SDK base do Android.

A SDK possui uma classe chamada `CardboardView`, em que são realizadas todas as operações de renderização de imagem para a tela do dispositivo. Para o projeto, foi integrada a esta classe a opção de configurar de qual servidor seria adquirido o vídeo e demais parâmetros de reprodução descritos na seção de implementação.

A reprodução de vídeos é realizada renderizando cada *frame* do vídeo em uma textura. Essa textura contendo os *frames* do vídeo é então projetada em uma superfície, em que a forma da superfície determina a deformação do vídeo ao ser projetado para o usuário.

Parte II

Desenvolvimento

4 Introdução

O projeto proposto é composto de dois atores principais: o *rickshaw*, que porta uma câmera e transmite o vídeo gravado do ambiente ao seu redor em tempo real para o segundo ator, o usuário que irá visualizar o vídeo utilizando o Google Cardboard. De forma simplificada, esse caso de uso pode ser interpretado como uma arquitetura cliente e servidor, em que o *rickshaw* atua como servidor e o usuário que irá visualizar o vídeo atua como cliente. Dessa forma, o desenvolvimento foi dividido em dois módulos: o aplicativo sendo executado no *smartphone* do cliente e irá possibilitar a visualização do vídeo, e o código servidor responsável pela gravação e transmissão do vídeo.

O processo de desenvolvimento do projeto utilizou a metodologia em Cascata, descrita em (PRESSMAN, 2009). A escolha dessa metodologia em particular foi devida ao projeto apresentar, desde o início, requisitos bem definidos, além de possuir uma arquitetura simples.

5 Especificação

Seguindo o Modelo em Cascata, o processo de desenvolvimento do projeto foi realizado de forma sistemática e sequencial, através das cinco fases definidas no modelo. A fase inicial corresponde ao levantamento de requisitos funcionais e não-funcionais do projeto, seguindo as fases de planejamento, modelagem, construção e finalmente implantação do sistema. A seguir são descritas todas as etapas de desenvolvimento do projeto.

5.1 Requisitos

Nesta seção são listados os requisitos funcionais e não-funcionais do projeto. Os requisitos foram levantados a partir das funcionalidades esperadas do projeto e serão divididos em dois módulos: cliente e servidor. O módulo cliente será responsável por receber o vídeo disponível no servidor e sua reprodução no *smartphone*. O módulo servidor será responsável por enviar aos clientes o vídeo adquirido através da câmera, assim como possibilitar o ajuste de parâmetros de mídia. Requisitos funcionais serão identificados por REF. e não-funcionais serão identificados por RNF, cada requisito recebendo uma identificação numérica única.

5.1.1 Aplicação Cliente

A aplicação cliente é um aplicativo para a plataforma Android, que será responsável pela reprodução do vídeo enviado pelo servidor. Abaixo são apresentadas as características desse módulo com os seus respectivos requisitos.

5.1.1.1 Reprodução por *Streaming*

5.1.1.1.1 Descrição

O dispositivo deve ser capaz de receber um vídeo por *streaming* e reproduzí-lo em Realidade Virtual ao usuário. Os requisitos dessa funcionalidade definem suporte a protocolos de comunicação e opções de reprodução.

5.1.1.1.2 Requisitos

RNF1 - Suporte a protocolos de *streaming*: O aplicativo deve suportar os protocolos comumente usados para transmissão de vídeos por *streaming*. Os mais utilizados e escolhidos como prioridades para o aplicativo são os protocolos RTSP e HLS.

RNF2 - Reprodução em Realidade Virtual: O aplicativo irá reproduzir o vídeo utilizando uma interface que possibilite a visualização em Realidade Virtual através do uso do Google Cardboard.

RNF3 - Reprodução com Estereoscopia: A reprodução do vídeo terá suporte à visão estereoscópica.

REF1 - Definição do Servidor: O usuário deve ser capaz de definir de qual servidor receber um vídeo através de um campo de texto para inserção do endereço desse servidor.

REF2 - Controle de Parâmetros de Reprodução: O usuário deve ser capaz de configurar parâmetros de reprodução do vídeo. Os seguintes parâmetros podem ser configurados:

- Superfície de projeção do vídeo
- Dimensões da superfície

5.1.2 Servidor

O módulo servidor é responsável por receber o vídeo da câmera e fornecê-lo à aplicação cliente. Como prova de conceito, o servidor é executado na mesma máquina que recebe o vídeo da câmera, de forma que o vídeo fica armazenado num diretório local para transmissão. Abaixo são apresentadas as características do servidor e seus respectivos requisitos.

5.1.2.1 Aquisição do vídeo

5.1.2.1.1 Descrição

O servidor irá adquirir o vídeo através da câmera USB instalada e armazená-lo num diretório local para transmissão. A aquisição do vídeo será realizada utilizando o *framework* FFmpeg, responsável pela definição dos parâmetros de aquisição.

5.1.2.1.2 Requisitos

RNF4 - Armazenamento do vídeo: O vídeo deverá ser armazenado num diretório local da máquina servidor para ser transmitido posteriormente.

REF3 - Configuração de parâmetros de aquisição: Deverá ser possível a configuração dos parâmetros de aquisição do vídeo pela câmera. Os parâmetros serão os disponibilizados pelo *framework*, sendo os mais utilizados:

- Resolução do vídeo

- *Codec*
- Formato de imagem
- Tamanho do segmento do vídeo
- Gravação monoscópica ou estereoscópica

5.1.2.2 Transmissão

5.1.2.2.1 Descrição

A transmissão do vídeo será realizada através de um protocolo de *streaming* e disponibilizado utilizando um servidor HTTP. Os requisitos para transmissão são descritos a seguir.

5.1.2.2.2 Requisitos

RNF5 - Servidor HTTP: Na máquina local um servidor HTTP deve estar em execução para que os vídeos sejam disponibilizados para o módulo cliente. O servidor também deve estar configurado de forma a possibilitar o acesso aos arquivos.

RNF6 - Transmissão por HLS: A transmissão do vídeo será realizada através do protocolo HLS. Dessa forma, deve estar disponível um arquivo contendo metadados do vídeo para que este seja devidamente acessado pelo módulo cliente.

RNF7 - *Multicasting*: O servidor deve ser capaz de transmissão em *multicasting*. Requisições concorrentes serão tratadas pelo servidor HTTP.

5.2 Planejamento

5.2.1 Estrutura Analítica de Projeto

Para determinar as atividades a serem realizadas durante a execução do projeto, foi construída a Estrutura Analítica de Projeto (EAP) (IEEE, 2005). A Figura 19 apresenta a EAP para este projeto de formatura.

A EAP apresenta todas as atividades realizadas no projeto. A primeira parte do projeto consistiu em estudos teóricos sobre estereoscopia e suas aplicações práticas. Em seguida foi realizado um estudo sobre câmeras para determinar o melhor modelo para uso no projeto. Durante o estudo de viabilidade, foram realizados estudos sobre transmissão em tempo-real e reprodução de vídeos utilizando o Carboard. O desenvolvimento da documentação, que inclui a monografia e relatórios parciais, foi realizado durante todo o projeto.

Figura 19 – Estrutura Analítica do Projeto.

Fonte: Autores

5.3 Modelagem

Nesta etapa é definida a arquitetura do sistema a partir dos requisitos levantados na primeira etapa de desenvolvimento. Durante o estudo de viabilidade já foi possível obter uma versão primitiva da arquitetura, que foi desenvolvida e alterada conforme novos requisitos foram sendo adicionados ao projeto. A arquitetura final implementa o modelo de arquitetura em camadas, sendo dividida em três módulos e organizados em três camadas.

A Camada de Apresentação é responsável pela interface gráfica que interage com o usuário e implementada pelo módulo Renderer no aplicativo. A Camada de Negócio, responsável pela aquisição do vídeo do servidor e renderização para posterior visualização em realidade virtual, também é implementada pelo aplicativo no componente MediaPlayer e pelo *Rickshaw* através do módulo Media Server. A Camada de Dados, em que é realizada a gravação do vídeo pela Câmera e seu armazenamento no Servidor, é implementada pelo módulo Recorder. A Figura 20 a seguir ilustra as relações entre os módulos.

Para a Prova de Conceito do projeto a aquisição e armazenamento do vídeo são implementados pelo mesmo módulo. A câmera utilizada na aquisição é conectada a um *notebook* através de uma porta USB e o vídeo gravado é imediatamente armazenado em um servidor HTTP local sendo executado no computador. Tal abordagem foi adotada devido a limitações de aquisição de vídeo por um dispositivo móvel que fosse compatível com a reprodução futura utilizando Realidade Virtual.

Utilizando a arquitetura proposta como referência, a sequência de operações realizadas entre a gravação e reprodução do vídeo pode ser visualizada na Figura 21 abaixo e será utilizada posteriormente na fase de testes.

O módulo Rickshaw será responsável pela aquisição, codificação e armazenamento do vídeo, enquanto aguarda requisições do módulo cliente. Este por sua vez realiza as

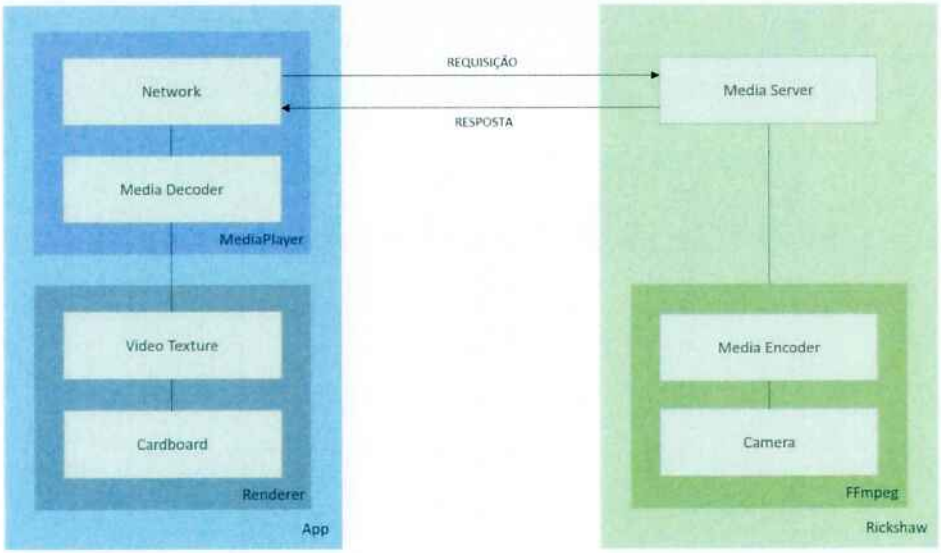

Figura 20 – Arquitetura do sistema.

Fonte: Autores

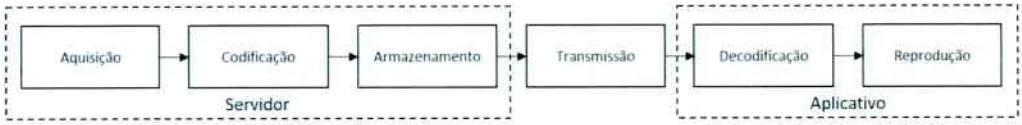

Figura 21 – Sequência de operações realizadas entre a gravação e reprodução do vídeo.

Fonte: Autores

operações de decodificação do vídeo transmitido e finalmente sua reprodução.

5.3.1 *Rickshaw*

O módulo *Rickshaw* é o módulo resposável pela aquisição da imagem através da câmera. Esse módulo também é responsável pelo envio do vídeo ao servidor. O método de envio depende do tipo de câmera utilizada, e no contexto do projeto é utilizada uma conexão USB entre a câmera e o Servidor. A câmera utilizada é uma câmera USB com lente *fisheye*, utilizada comumemente para monitoramento. A câmera possibilita aquisição de imagem com resolução máxima de 1920x1080 e com taxa de aquisição de 30 *frames* por segundo. É possível utilizar outras tecnologias de aquisição. como câmeras *wireless* ou câmeras de dispositivos móveis, sendo necessário apenas implementar o método de envio do vídeo para o servidor. Esse módulo representa fisicamente a pessoa portando a câmera enquanto anda e enviando o vídeo para o servidor.

A aquisição do vídeo é controlada utilizando o *framework* FFmpeg, utilizando os

parâmetros já mencionados. A configuração de tais parâmetros é realizada através de linhas de comando durante a execução da aplicação.

5.3.2 Servidor

O módulo servidor é responsável por receber o vídeo enviado pela câmera, armazená-lo e então enviá-lo para os clientes que o requisitarem. O servidor disponibiliza os vídeos para os clientes utilizando algum dos protocolos utilizados para *streaming* já apresentados, com a ressalva de tal protocolo ser suportado pelo módulo cliente. Dado o requisito RNF6, o servidor irá utilizar o protocolo HLS, disponibilizando os vídeos através de um servidor HTTP.

5.3.3 Aplicativo

O módulo aplicativo está localizado no dispositivo móvel, implementando os requisitos RNF1, RNF2, RNF3, REF1 e REF2. A principal função do aplicativo é receber o vídeo armazenado no módulo Servidor e reproduzí-lo utilizando a tecnologia de realidade virtual. A reprodução do vídeo é controlada por diversos parâmetros, como especificado nos requisitos do módulo.

A Figura 22 a seguir apresenta o diagrama das classes implementadas no aplicativo e suas relações. As classes `VrViewActivity` e `VRActivity` dependem, em última instância, da classe base `Activity` fornecida pela plataforma Android e não apresentada na figura por brevidade. A classe `VrViewRenderer` implementa a funcionalidade de renderização do vídeo em realidade virtual, e é utilizada pela classe `VrViewActivity`. A classe `MainActivity`, que também herda da classe base `Activity`, é responsável pela interface gráfica com o usuário para a configuração dos parâmetros de reprodução, utilizando, para isto, elementos gráficos como botões e demais recursos através da classe `View`, também fornecida pela plataforma.

Além disso, foram utilizadas bibliotecas de terceiros para suporte ao desenvolvimento do aplicativo, detalhadas na seção seguinte.

5.4 Construção

5.4.1 Aplicativo

O aplicativo foi desenvolvido utilizando-se o ambiente de desenvolvimento Android Studio, em plataforma Windows. Na codificação, foram utilizadas bibliotecas para auxiliar no desenvolvimento do módulo de realidade virtual, responsável pela reprodução do vídeo. Para os testes durante e depois do desenvolvimento, foi utilizado um *smartphone* com o

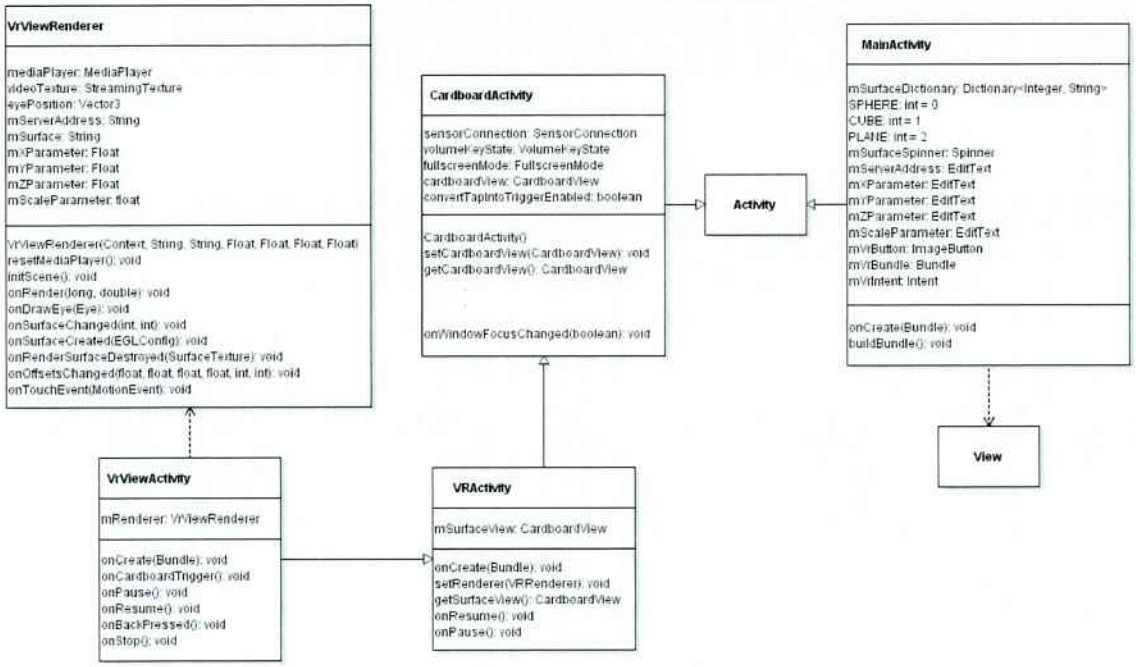

Figura 22 – Diagrama de classes do aplicativo cliente.

Fonte: Autores

sistema Android versão 21.0. As ferramentas utilizadas no desenvolvido são descritas a seguir.

5.4.1.1 Android Studio

Android Studio é a IDE (Integrated Development Environment) oficial para o desenvolvimento de aplicativos para a plataforma Android. Disponível para os sistemas operacionais Windows, Linux e Mac OS, o Android Studio oferece uma ampla gama de ferramentas para auxiliar no desenvolvimento.

Além da possibilidade de utilizar um emulador para execução do aplicativo, é possível a execução em dispositivos físicos utilizando a interface USB fornecida pela IDE. O Android Studio também possibilita a utilização e integração de bibliotecas externas de forma simples e descomplicada através de repositórios, de forma a garantir que o desenvolvedor direcione toda a atenção para o desenvolvimento. A Figura 23 apresenta uma das telas do ambiente de desenvolvimento.

Um problema comumente encontrado no desenvolvimento de aplicativos para a plataforma Android é em relação à grande variedade de dispositivos que a utilizam, sendo necessário adaptar o aplicativo de forma que seja executado em diferentes versões do

Android e para diferentes tamanhos e resoluções de telas. O Android Studio simplifica tal tarefa exigindo que o desenvolvedor precise apenas fornecer os recursos gráficos em tamanhos padronizados. As tarefas de redimensionamento, posicionamento e seleção dos elementos gráficos baseado no tamanho e resolução da tela fica a cargo da plataforma em tempo de execução.

A IDE também fornece diversos exemplos e modelos de aplicativos para auxiliar o desenvolvedor inexperiente, em conjunto com os guias oficiais de desenvolvimento fornecidos no site oficial da plataforma Android.

As seções a seguir descrevem as bibliotecas e classes utilizadas no desenvolvimento do aplicativo.

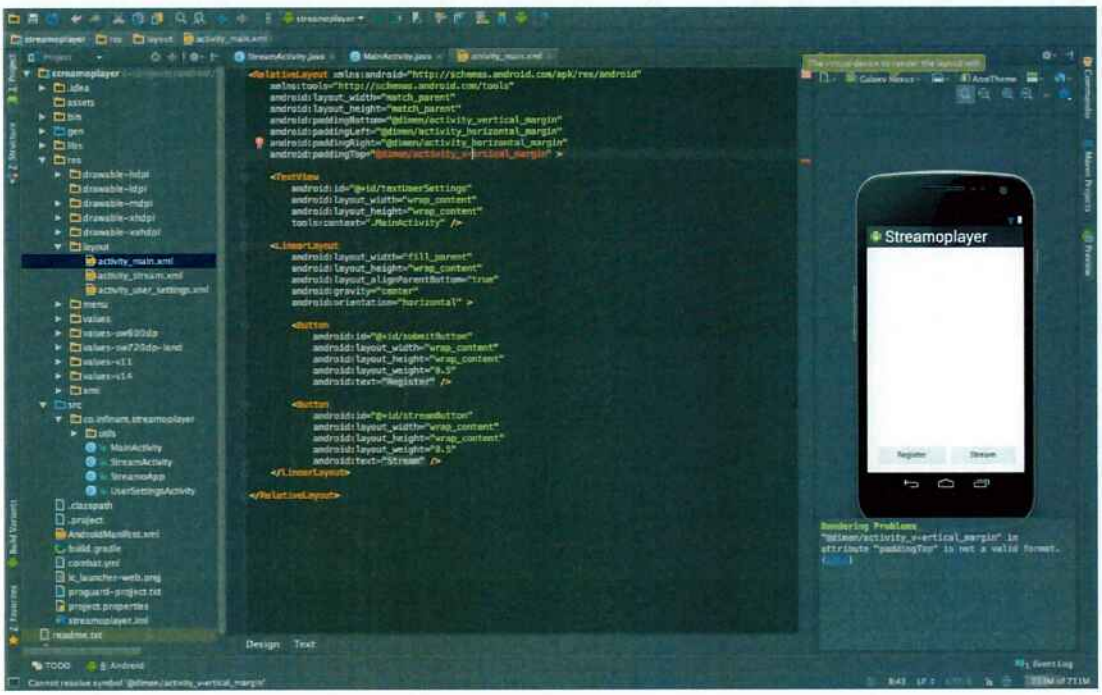

Figura 23 – Tela do ambiente de desenvolvimento Android Studio.

Fonte: Autores

5.4.1.2 Classe `MediaPlayer`

A classe `MediaPlayer` (Android, 2016d) é disponibilizada pela API da plataforma Android e oferece funcionalidades de controle de reprodução de áudio e vídeo. A classe suporta a reprodução de mídia de diversas fontes, sendo as mais utilizadas arquivos de mídia armazenadas localmente no *smartphone*, em servidores HTTP ou servidores que disponibilizam vídeo por *streaming* através dos protocolos RTSP, RTMP e HLS.

O diagrama de estados na Figura 24 apresenta o ciclo de vida de uma instância da classe `MediaPlayer`. Para que seja possível reproduzir um vídeo utilizando a classe `MediaPlayer`, é necesário definir a fonte da mídia através do método `setDataSource`. Em seguida, é necessário chamar o método `prepareAsync`, responsável por inicializar o carregamento da mídia e da criação do *buffer* de reprodução. Uma vez realizadas essas operações, é possível então iniciar a reprodução do vídeo através do método `start`.

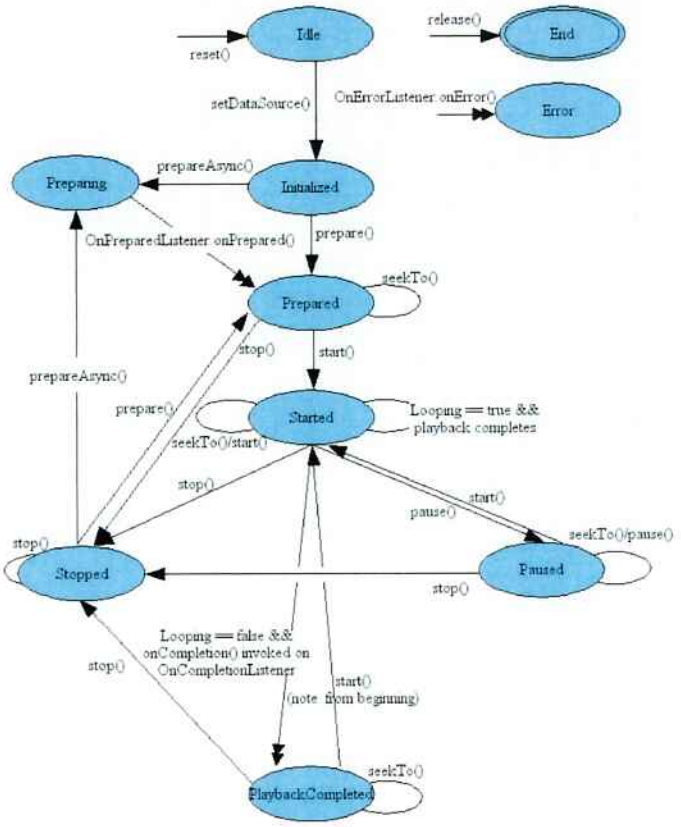

Figura 24 – Ciclo de vida de uma instância da classe `MediaPlayer`.

Fonte: (Android, 2016d)

Para que seja possível a visualização do vídeo sendo reproduzido, é necessário projetar o vídeo em uma superfície utilizando uma textura. A biblioteca Rajawali, descrita a seguir, foi utilizada para a projeção do vídeo em superfícies.

5.4.1.3 Rajawali

Rajawali (Rajawali, 2016) é uma biblioteca *open-source* para a plataforma Android baseada em OpenGL e que oferece funcionalidades de renderização de vídeos. A biblioteca permitiu a integração entre as classes `MediaPlayer` e `CarboardView`, para que fosse possível a reprodução do vídeo em realidade virtual. A integração é realizada na classe

`VrViewRenderer`, em que o vídeo carregado pela instância da classe `MediaPlayer` é utilizado como uma textura. Essa textura é então projetada em uma superfície para que seja possível utilizá-la na visualização em realidade virtual. É possível projetar a textura em qualquer espécie de superfície, porém, só foram obtidos bons resultados utilizando a esfera e o cubo como superfícies.

Por ser baseada na difundida *engine* OpenGL, utilizada amplamente em aplicações de renderização 3D, a biblioteca já implementa as operações de manipulação de superfícies e objetos tridimensionais, assim como a utilização de texturas e materiais para aqueles. A biblioteca é atualizada frequentemente e acompanha de perto as atualizações na SDK para o Google Cardboard, garantindo assim suporte contínuo para futuras funcionalidades.

5.4.2 Servidor HTTP

O servidor HTTP utiliza a distribuição Apache (The Apache Software Foundation, 2016) para armazenar e disponibilizar os arquivos de vídeo aos clientes. A solução de utilizar o servidor Apache foi devida à simplicidade de instalação e configuração, bastando executar a aplicação para que o servidor seja iniciado e para que seja possível realizar as requisições do lado cliente.

Os arquivos de vídeo ficam armazenados no diretório raiz do servidor, sendo necessário apenas conhecer o nome do arquivo para que seja feita a requisição. Como era de interesse apenas disponibilizar os arquivos para o cliente, não foram implementadas funções adicionais referentes a segurança, disponibilidade ou desempenho.

5.4.3 Aquisição

A aquisição do vídeo é realizada através de uma câmera USB com lente *fisheye* de 180°. A escolha da câmera foi feita com o objetivo do observador poder visualizar em totalidade o ambiente sendo filmado, utilizando a agência da cabeça como modo de interação com o meio. É possível, porém, também utilizar outras câmeras, com a ressalva de que a imagem gerada pela aquisição terá um menor campo de visão, o que pode comprometer a sensação de imersão do usuário. A seguir são descritos os modos de aquisição.

5.4.3.1 Aquisição Monoscópica de Imagem

A aquisição monoscópica de imagem é feita através do *framework* FFmpeg, executado na mesma máquina hospedando o servidor HTTP. Para que seja possível reproduzir o vídeo no aplicativo é necessária a configuração correta de certos parâmetros de aquisição. A seguir são apresentados os parâmetros utilizados na gravação.

- Input
  É necessário especificar o dispositivo de gravação pelo qual será realizada a aquisição de imagem. Para isso é usado o argumento `-i video=<nome_dispositivo>`. O nome da câmera utilizada no projeto é "HD USB Camera".

- *Codec*
  O *codec* utilizado na gravação do vídeo deve ser especificado pelo argumento `-vcodec`. O *codec* sendo usado é o *libx264*.

- Formato dos Pixels
  O formato dos pixels determina o tipo de codificação das cores de cada *frame* e é especificado pelo argumento `-pix_fmt`. O formato utilizado é o *yuv420p*.

- Resolução
  A resolução do vídeo é configurada pelo parâmetro `-s`. Para melhorar o desempenho na transmissão, foi utilizada a resolução HD, de 1280x720.

- Alocação de memória
  É possível configurar o tamanho máximo de memória que pode ser utilizado pelo *buffer* de gravação para *frames* em tempo-real e evitar, assim, perda de *frames*. O argumento que controla tal parâmetro é `-rtbufsize`, sendo utilizado o valor de 1500M.

- Otimização
  Para otimizar a aquisição de vídeo, é possível ajustar parâmetros de baixo nível através de opções pré-configuradas e determinadas para situações específicas, através dos argumentos `-tune` e `-preset`. Para o contexto do projeto, que utiliza transmissão em tempo-real, é utilizada a opção `zerolatency` para o argumento `-tune`, que garante rápida codificação do vídeo e baixa latência. Para o argumento `-preset`, que controla a compressão do vídeo, foi utilizada a opção `ultrafast`, que prioriza a velocidade sobre a qualidade na compressão.

- Saída
  O arquivo de saída, assim como seu formato, são os últimos argumentos a serem especificados na linha de comando. A extensão do arquivo especificada é utilizada pelo FFmpeg para determinar o formato do arquivo. Para a transmissão utilizando HLS, é necessário apenas especificar um arquivo de saída com a extensão .m3u8.

A linha de comando final, com todas as opções acima descritas, fica como descrito a seguir:

```
ffmpeg -f dshow -i video="HD USB Camera" -rtbufsize 1500M -vcodec libx264
-pix_fmt yuv420p -s 1280x720 -tune zerolatency -preset ultrafast foo.m3u8
```

5.4.3.2 Aquisição Estereoscópica de Imagem

Para a aquisição estereoscópica de imagem, é necessária a configuração de todos os parâmetros discutidos na aquisição monoscópica, além da configuração de parâmetros adicionais. A aquisição deve ser realizada utilizando-se duas câmeras que irão simular o olho humano, separadas por uma distância adequada para que seja possível aplicar o efeito de estereoscopia. As imagens adquiridas das duas câmeras são processadas pelo FFmpeg e combinadas em um único vídeo de saída, em que são dispostas lado a lado. A Figura 25 apresenta um exemplo do vídeo de saída, porém foi utilizado um vídeo já gravado com uma única câmera. Para o caso em que é utilizado estereoscopia cada câmera iria fornecer uma imagem e estas duas seriam combinadas. A seguir são apresentados os parâmetros adicionais que devem ser configurados para este modo de aquisição.

Figura 25 – Ciclo de vida de uma instância da classe `MediaPlayer`.

Fonte: Autores

- Filtros Complexos
  Este parâmetro deve ser utilizado quando é a realizada a aquisição de múltiplas fontes, como ocorre na estereoscopia. Além disso, esse parâmetro possibilita a utilização de filtros especiais que podem ser aplicados às imagens adquiridas. O argumento que indica a utilização dessa funcionalidade é `-filter_complex`, seguido dos filtros a serem utilizados entre aspas e separados por ponto-e-vírgula. A seguir são descritos os filtros utilizados para a aplicação do efeito estereoscópico.

- Deslocamento
  O filtro *pad* possibilita o deslocamento do vídeo pela tela. Esse filtro foi utilizado para que as imagens adquiridas das duas câmera fossem colocadas lado a lado. O filtro desloca as imagens nos eixos x e y a partir de valores definidos pelo usuário. Para que as duas imagens fiquem lado a lado é necessário deslocar uma delas no sentido positivo do eixo x o valor correspondente à largura do vídeo. O filtro é aplicado passando o argumento `pad=<dx>:<dy>`, em que `dx` e `dy` são os deslocamentos horizontal e vertical, respectivamente.

- Sobreposição
  O filtro *overlay* é necessário para que possa ser realizada a fusão dos dois vídeos, ajustando a resolução do vídeo de saída para que os dois sejam visualizados lado a lado. Os argumentos que devem ser passados para o filtro são as posições em x e y do vídeo a ser sobreposto. Após ser realizado o deslocamento de um dos vídeos para a direita, é passada sua posição para o filtro *overlay*, que corresponde à largura do vídeo à esquerda. O filtro é aplicado através do argumento `overlay=<x=0>:<y=0>`.

A linha de comando final, incluindo os parâmetros adicionais para a aquisição estereoscópica, fica como descrito a seguir. Como a aquisição é feita a partir de duas câmeras, é necessário especificar duas fontes de imagens através do argumento `-i`. Os textos entre colchetes na seção dos filtros representam marcadores para que todos os filtros possam ser combinados e aplicados um após o outro, em sequência.

```
ffmpeg -f dshow -i video="HD USB Camera 1" -i video="HD USB Camera 2"
-filter_complex "[0:v:0]pad=iw*2:ih[bg]; [bg][1:v:0]overlay=w"
-rtbufsize 1500M -vcodec libx264 -pix_fmt yuv420p -s 1280x720
-tune zerolatency -preset ultrafast foo.m3u8
```

5.5 Plano de Testes

Os testes finais são divididos em dois módulos, testes objetivos e testes subjetivos. Os testes objetivos consideram os requisitos do projeto e abordam medições de latência na transmissão dos vídeos, resolução e consumo de recursos computacionais na gravação e reprodução. Os testes subjetivos são realizados por meio de usuários que utilizam o sistema e descrevem a sensação de imersão ao visualizar o vídeo pelo Google Cardboard.

5.5.1 Testes Objetivos

Os testes objetivos medem a latência na transmissão do vídeo entre o servidor e o aplicativo e o consumo de recursos físicos na gravação e reprodução dos vídeos, ambos em função da resolução utilizada na reprodução. Como os fatores medidos dependem do ambiente em que o sistema está sendo executado, é importante destacar as configurações dos equipamentos utilizados, descritos a seguir. A máquina utilizada como servidor nos testes é um notebook com processador Intel(R) Celeron(R) de 2.2 GHz e arquitetura 32 bits, 3 GB de memória RAM e executando Windows 7. O *smartphone* executando o aplicativo é o modelo H442f da LG, com 1 GB de memória RAM e executando Android versão 5.0.1. Finalmente, a câmera utilizada na gravação dos vídeos é uma câmera USB com lente *fisheye*, modelo FHD01M da marca ELP.

5.5.1.1 Latência

Utilizando a sequência de operações apresentada na Figura 21, a latência na transmissão do vídeo seria idealmente definida como o intervalo de tempo entre o início e término da transmissão do vídeo entre o cliente e o servidor, como ilustrado na Figura 26. Porém, para uma medição precisa desse intervalo de tempo, seria necessária a modificação e recompilação da plataforma Android a partir de seu código fonte. Dada a inviabilidade da tarefa e considerando como nulos os intervalos de tempo nas etapas de decodificação e reprodução, a latência na transmissão é definida como o intervalo de tempo entre o instante em que a requisição por um *frame* chega ao servidor e o instante em que este *frame* é visualizado na tela do *smartphone*, compreendendo as operações de transmissão, decodificação e reprodução do vídeo, como ilustrado na Figura 27.

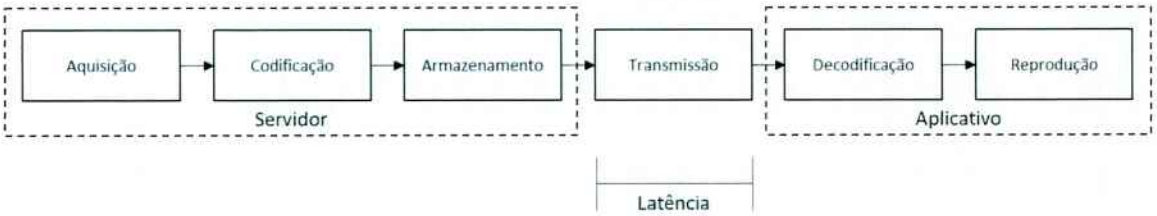

Figura 26 – Latência ideal

Fonte: Autores

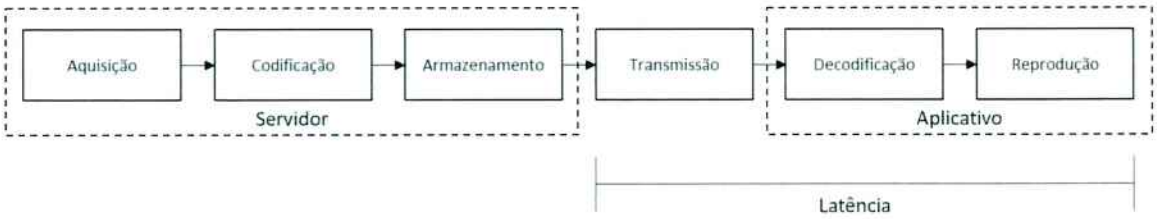

Figura 27 – Latência aproximada

Fonte: Autores

A latência é medida pelo aplicativo utilizando uma *thread* independente para garantir que as medições não sejam afetadas por outras operações. Para isso, foi utilizado o método `onPrepareAsync` da classe `MediaPlayer`. Esse método envia uma requisição para uma URL especificada e prepara a mídia recebida para reprodução de forma assíncrona. Quando a mídia está pronta para ser reproduzida, a classe `MediaPlayer` chama um *callback* definido em código. É possível, portanto, obter a latência de transmissão através do intervalo de tempo entre a chamada do método `onPrepareAsync` e a chamada do *callback* correspondente. A Tabela 2 abaixo apresenta as medições obtidas para a latência em função da resolução e do tipo de aquisição. Devido a inconsistências nos tempos observados para aquisição estereoscópica, tais dados não são apresentados aqui.

Tabela 2 – Testes de latência em função da resolução do vídeo.

Aquisição Monoscópica	
Resolução (px)	**Latência (s)**
1920 x 1080 (Full HD)	20.53
1280 x 720 (HD)	9.71
640 x 360 (SD)	0.66
320 x 180 (SD)	0.48

Fonte: Autores

5.5.1.2 Recursos Físicos

O consumo de recursos computacionais é medido no módulo servidor, dada a maior exigência de tais recursos pelas tarefas desempenhadas no módulo. Tais testes foram selecionados dada a possibilidade de implementação futura de tais módulos em um *smartphone*, que dispõe de limitações de recursos. A medição será realizada utilizando a aplicação ProcessExplorer, distribuída pela Microsoft. O ProcessExplorer fornece informações detalhadas de um determinado processo quanto ao consumo de memória, processador e operações de I/O. Para que todas as operações realizadas no módulo servidor sejam analisadas, é necessário monitorar dois processos: o servidor HTTP e a aplicação FFmpeg. De forma análoga aos testes realizados para latência, os testes desta seção serão realizados em função da resolução do vídeo e do tipo de aquisição. As tabelas a seguir (Tabela 3 e Tabela 4) apresentam os valores obtidos nos testes de recursos para os modos de aquisição monoscópico e estereoscópico.

Tabela 3 – Recursos computacionais consumidos em função da resolução para aquisição monoscópica.

Aquisição Monoscópica		
Resolução (px)	**Memória (MB)**	**CPU (%)**
1920 x 1080 (Full HD)	100	75
1280 x 720 (HD)	81	80
640 x 360 (SD)	70	62.5
320 x 180 (SD)	67	50

Fonte: Autores

5.5.2 Testes Subjetivos

Os testes subjetivos medem a sensação de imersão que os usuários do sistema têm ao visualizar o vídeo através do Google Cardboard. Para isso, é utilizada uma escala que varia de 0 a 5 em que os usuários avaliam o grau de imersão, sendo 1 o caso em que o usuário não se sente imerso no vídeo e 5 o caso em que ele sente total imersão. Para garantir a uniformidade dos testes, será utilizado o mesmo vídeo para todos os usuários que testarem o sistema. Além disso, serão utilizadas diferentes resoluções, com a expectativa

Tabela 4 – Recursos computacionais consumidos em função da resolução para aquisição estereoscópica.

Aquisição Estereoscópica		
Resolução (px)	**Memória (MB)**	**CPU (%)**
1920 x 1080 (Full HD)	150	70
1280 x 720 (HD)	130	70
640 x 360 (SD)	120	70
320 x 180 (SD)	120	70

Fonte: Autores

de que vídeos com melhores definições de imagem irão proporcionar um maior grau de imersão.

Um total de 10 pessoas foram entrevistadas, sendo apresentado, para cada uma, três vídeos do mesmo ambiente, porém com resoluções diferentes. Os valores apresentados na Tabela 5 indicam as notas fornecidas por cada pessoa em relação a cada um dos vídeos.

Tabela 5 – Testes subjetivos da sensação de imersão em função da resolução do vídeo.

Resolução (px)	1	2	3	4	5	6	7	8	9	10
1920 x 1080 (Full HD)	5	4	3	4	4	5	4	3	4	5
1280 x 720 (HD)	4	3	3	2	4	4	4	3	2	3
320 x 180 (SD)	3	1	3	1	2	3	3	2	2	1

Fonte: Autores

Através das notas obtidas é possível calcular a média do grau de imersão para cada uma das resoluções. Para a mais alta resolução, 1920 x 1080 (Full HD) *pixels*, o grau de imersão é próximo do máximo valor possível, com uma média de 4.1. À medida em que a resolução do vídeo diminui, também ocorre uma diminuição do grau de imersão obtido pelo usuário. A menor média obtida, de 2.1, corresponde à menor das resoluções testadas, ou seja, de 320 x 180 (SD) *pixels*.

6 Discussão

Fundamentando-se nos requisitos do projeto listados na seção 5.1 e nos resultados dos testes realizados e descritos na seção 5.5, é realizada uma breve discussão sobre os resultados alcançados para o projeto.

6.1 Sistema em Tempo-Real

Uma das principais características do projeto é a reprodução de um vídeo enquanto este é gravado remotamente, de forma que o observador possa visualizar em tempo-real o ambiente sendo gravado. A métrica que determina tal característica é a latência observada durante a transmissão do vídeo entre o observador e o *rickshaw* e que foi analisada durante a fase de testes.

Quando é utilizado o modo de aquisição monoscópico, a Tabela 2 apresenta valores que se adequam ao esperado, de modo que vídeos com maiores resoluções apresentam um maior tempo de transmissão. Para vídeos com resolução de 1920x1080 pixels, a maior resolução possível fornecida pela câmera, é notável o grande atraso durante a transmissão do vídeo e, portanto, não é o mais adequado para uma aplicação em tempo-real. A resolução de 1280x720 também apresenta um valor elevado no atraso, porém pode ser utilizada quando a qualidade do vídeo é priorizada sobre o tempo de transmissão. As duas últimas resoluções apresentam baixa definição, porém o atraso na transmissão pode ser considerado imperceptível e devem ser utilizadas para ambientes com banda de dados limitada.

Para o modo de aquisição estereoscópica, porém, os valores obtidos para a latência aparentam não apresentar relação com a resolução utilizada, o que contraria a hipótese levantada. Os valores de latência obtidos pelas duas maiores resoluções são praticamente indistinguíveis e para as duas menores resoluções ficaram abaixo da precisão fornecida na análise. Porém, sob melhor análise dos vídeos gravados, verificou-se que a qualidade destes foi drasticamente reduzida na etapa de aquisição e mantida uniforme para todas as resoluções testadas. Tal transformação se deve ao *framework* FFmpeg e supõe-se que é realizada devido aos filtros aplicados, que utilizam um alto nível de processamento e, portanto, causariam demasiado *overhead* caso utilizados com as resoluções nativas.

6.2 Recursos

O baixo consumo de recursos físicos não foi explicitamente definido como um requisito, porém, para trabalhos futuros em que a migração para ambientes móveis for

estudada, tal fator é indispensável para sua viabilidade.

Os valores obtidos para a aquisição monoscópica também estão dentro do esperado, como apresentado na Tabela 3. Vídeos com maiores resoluções apresentam maior consumo de memória na aquisição, sendo o máximo valor obtido de 100 MB para a resolução de 1920x1080. Para o processamento os valores obtidos também apresentam relação com a resolução. A discrepância obtida para as resoluções de 1920x1080 e 1280x720 pode ser atribuída a picos de processamento durante a gravação.

Para a aquisição estereoscópica, os valores de memória também apresentam forte relação com a resolução dos vídeos (Tabela 4). O processamento, porém, apresentou-se uniforme para todas as resoluções testadas.

Levando-se em consideração de que tais vídeos foram gravados utilizando-se um computador portátil, os valores de memória não se apresentam muito elevados. Para o processamento, porém, a demanda é relativamente alta, mesmo para as menores resoluções. Dada a limitação de recursos computacionais em dispositivos móveis, mesmo o baixo consumo de memória deve ser levado em consideração em uma possível migração para tal ambiente. Um dispositivo mediano apresenta 1 GB de memória, e o consumo que a gravação utilizaria é relativamente alto para um único processo. Da mesma forma, deve ser considerado o processamento demandado pela aplicação, buscando formas de otimização para projetos futuros.

6.3 Imersão

Outra característica importante para o projeto é em relação à imersão, ou seja, à possibilidade do usuário final se sentir imerso num local remoto através de sua visualização utilizando a tecnologia de realidade virtual. Os testes realizados sobre tal característica, apesar de subjetivos, refletem bem a sensação do usuário ao utilizar o projeto.

Em relação aos resultados dos testes, os valores obtidos para a sensação de imersão experimentada pelos usuários apresentam relação com a resolução do vídeo reproduzido, como indicado na Tabela 5. Na resolução mais alta, apesar do grande atraso observado na reprodução, a sensação de imersão se apresentou mais alta. Passando pelas resoluções de 1280x720 e 320x180, é notável a diminuição do grau de imersão experimentado devido à menor resolução das imagens. Na menor resolução, apesar do pequeno atraso na transmissão, a imagem se apresenta borrada e pixelada, fator crucial para a menor imersão do usuário.

7 Conclusão

Dado o objetivo proposto de desenvolvimento de um ambiente imersivo, em que um usuário à distância possa realizar um passeio em realidade virtual e em tempo-real utilizando a agência da cabeça, o presente trabalho atende às expectativas levantadas. Utilizando diversas tecnologias, o trabalho propõe uma infraestrutura que soluciona o problema apresentado através do levantamento e implementação de requisitos necessários para concretizar a solução.

O projeto possuía desde o início um forte caráter multidisciplinar, envolvendo múltiplas áreas de computação como evidenciado na Parte I, tais como redes de computadores, novas mídias digitais, aquisição e manipulação de imagem e desenvolvimento de aplicações para dispositivos móveis. Tal característica necessitou de uma extensa e produtiva fase de pesquisa, fortemente auxiliada pelos orientadores, e que se mostrou indispensável para o efetivo desenvolvimento do projeto.

A fase de aquisição do projeto, que utiliza processamento e manipulação de imagem, possibilitou o conhecimento de uma área completamente nova para os alunos. A pesquisa e desenvolvimento forneceu contato com ferramentas amplamente utilizadas pela comunidade de computação gráfica e que foram essenciais para viabilizar o desenvolvimento do projeto.

A fase de transmissão foi fundamentada nas RFCs relativas aos protocolos de comunicação utilizados para aplicações em tempo-real e proporcionaram o conhecimento necessário para o desenvolvimento ou utilização de ferramentas adequadas que implementam tais protocolos. A pesquisa de tal fase também foi apoiada pelo conhecimento adquirido na disciplina Redes de Computadores, guiando suas fases iniciais. Tal fase proporcionou a integração entre todos os módulos do projeto, demandando, portanto, especial atenção.

Em relação à fase de reprodução do vídeo, o auxílio fornecido pelos orientadores se mostrou de suma importância para finalizar o desenvolvimento do projeto e envolve a área de novas mídias digitais. Os conceitos estudados durante a pesquisa envolvendo imersão e realidade virtual foram amplamente guiados pelos orientadores e fundamentaram a escolha das tecnologias e bibliotecas utilizadas nessa fase.

Finalmente, os conhecimentos adquiridos durante o curso relativos à especificação, modularização e desenvolvimento de um projeto de engenharia se mostraram indispensáveis para o projeto. Os processos ensinados durante o curso garantiram uma abordagem lógica e eficiente nas etapas de pesquisa e desenvolvimento do trabalho, de forma que serviu como espinha dorsal sobre a qual foi estruturada toda a solução para o problema proposto.

Referências

Android. *Android API Reference*. 2016. Acessado em 08 nov 2016. Disponível em: <https://developer.android.com/reference/packages.html>. Citado na página 30.

Android. *Android Studio*. 2016. Acessado em 08 nov 2016. Disponível em: <https://developer.android.com/studio/index.html>. Citado na página 30.

Android. *Introduction to Android*. 2016. Acessado em 08 nov 2016. Disponível em: <https://developer.android.com/guide/index.html>. Citado na página 30.

Android. *Media Player*. 2016. Acessado em 10 nov 2016. Disponível em: <https://developer.android.com/reference/android/media/MediaPlayer.html>. Citado 2 vezes nas páginas 42 e 43.

Apple Inc. *HTTP Live Streaming*. 2016. Acessado em 08 nov 2016. Disponível em: <https://developer.apple.com/streaming/>. Citado na página 29.

ARAUJO, R. B. de. *Especificação e análise de um sistema distribuído de realidade virtual*. Tese (Doutorado) — Escola Politécnica da Universidade de São Paulo, 1996. Citado 3 vezes nas páginas 19, 20 e 21.

FFMPEG. *FFmpeg*. 2016. Acessado em 08 nov 2016. Disponível em: <http://ffmpeg.org/>. Citado na página 26.

FFMPEG. *FFmpeg Documentation*. 2016. Disponível em: <http://ffmpeg.org/documentation.html>. Citado na página 27.

FIELDING, R.; RESCHKE, J. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. IETF, 2014. RFC 7230 (Proposed Standard). (Request for Comments, 7230). Disponível em: <http://www.ietf.org/rfc/rfc7230.txt>. Citado na página 28.

Google Inc. *Google Cardboard*. 2016. Acessado em 08 nov 2016. Disponível em: <https://vr.google.com/cardboard/>. Citado 3 vezes nas páginas 30, 31 e 32.

Google Inc. *Google VR SDK*. 2016. Acessado em 08 nov 2016. Disponível em: <https://developers.google.com/vr/cardboard/overview>. Citado na página 32.

IEEE. IEEE Standard for Application and Management of the Systems Engineering Process. *IEEE Std. 1220-2005*, 2005. Citado na página 37.

JACOBSON, L. *Realidade Virtual em Casa*. [S.l.]: Editora Berkeley, 1994. Citado na página 21.

JOHNSON, L. et al. *NMC Horizon Report: 2016 Higher Education Edition*. [S.l.], 2016. Citado na página 11.

LATTA, J.; OBERG, D. A conceptual virtual reality model. *IEEE Computer Graphics and Applications*, Institute of Electrical and Electronics Engineers (IEEE), v. 14, n. 1, p. 23–29, jan 1994. Disponível em: <http://dx.doi.org/10.1109/38.250915>. Citado na página 20.

MACHADO, L. dos S. *Conceitos Básicos da Realidade Virtual.* [S.l.], 1995. [INPE-5975-PUD/0-25]. Citado na página 21.

NETTO, A. V.; MACHADO, L. dos S.; OLIVEIRA, M. C. F. de. *Realidade Virtual: Definições, Dispositivos e Aplicações.* [S.l.], 2002. 33 páginas. Citado 2 vezes nas páginas 20 e 21.

OpenCV. *Camera Calibration with OpenCV.* 2016. Acessado em 08 nov 2016. Disponível em: <http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html>. Citado na página 26.

OpenCV. *Open Source Computer Vision.* 2016. Acessado em 08 nov 2016. Disponível em: <http://opencv.org/>. Citado na página 25.

OpenCV. *The OpenCV Reference Manual.* [S.l.], 2016. Release 2.4.13.1. Disponível em: <http://docs.opencv.org/2.4/opencv2refman.pdf>. Citado na página 26.

PANTOS, R.; MAY, W. *HTTP Live Streaming.* IETF, 2016. Draft-pantos-http-live-streaming-20. (Internet-Draft). Disponível em: <https://tools.ietf.org/html/draft-pantos-http-live-streaming-20>. Citado na página 29.

PIMENTEL, K. *Virtual reality : through the new looking glass.* New York: Intel/McGraw-Hill, 1995. ISBN 0070501688. Citado 3 vezes nas páginas 21, 22 e 23.

POSTEL, J. *User Datagram Protocol.* IETF, 1980. RFC 768 (INTERNET STANDARD). (Request for Comments, 768). Disponível em: <http://www.ietf.org/rfc/rfc768.txt>. Citado na página 28.

POSTEL, J. *Transmission Control Protocol.* IETF, 1981. RFC 793 (INTERNET STANDARD). (Request for Comments, 793). Updated by RFCs 1122, 3168, 6093, 6528. Disponível em: <http://www.ietf.org/rfc/rfc793.txt>. Citado na página 28.

PRESSMAN, R. S. *Software Engineering: A Practitioner's Approach.* 7. ed. [S.l.]: McGraw-Hill Education, 2009. ISBN 0073375977. Citado 2 vezes nas páginas 24 e 34.

Rajawali. *Rajawali: Android OpenGL ES 2.0/3.0 Engine.* 2016. Acessado em 10 nov 2016. Disponível em: <https://github.com/Rajawali/Rajawali>. Citado na página 43.

ROBERTSON, G.; CARD, S.; MACKINLAY, J. Three views of virtual reality: nonimmersive virtual reality. *Computer*, Institute of Electrical and Electronics Engineers (IEEE), v. 26, n. 2, p. 81, feb 1993. Disponível em: <http://dx.doi.org/10.1109/2.192002>. Citado 2 vezes nas páginas 12 e 19.

SCHULZRINNE, H. et al. *RTP: A Transport Protocol for Real-Time Applications.* IETF, 2003. RFC 3550 (INTERNET STANDARD). (Request for Comments, 3550). Updated by RFCs 5506, 5761, 6051, 6222, 7022, 7160, 7164. Disponível em: <http://www.ietf.org/rfc/rfc3550.txt>. Citado na página 26.

SCHULZRINNE, H.; RAO, A.; LANPHIER, R. *Real Time Streaming Protocol (RTSP).* IETF, 1998. RFC 2326 (Proposed Standard). (Request for Comments, 2326). Disponível em: <http://www.ietf.org/rfc/rfc2326.txt>. Citado 2 vezes nas páginas 27 e 28.

TANENBAUM, A. S. *Computer Networks.* 4. ed. [S.l.]: Prentice Hall, 2007. Citado 2 vezes nas páginas 28 e 29.

The Apache Software Foundation. *Apache HTTP Server Project*. 2016. Acessado em 10 nov 2016. Disponível em: <http://httpd.apache.org/>. Citado na página 44.

THORNBURGH, M. *Adobe's RTMFP Profile for Flash Communication*. IETF, 2014. RFC 7425 (Informational). (Request for Comments, 7425). Disponível em: <http://www.ietf.org/rfc/rfc7425.txt>. Citado na página 27.

TORI, R.; KIRNER, C.; SISCOUTTO, R. *Fundamentos e Tecnologia de Realidade Virtual e Aumentada*. [S.l.]: Editora SBC, 2006. Citado 7 vezes nas páginas 14, 15, 16, 17, 18, 19 e 21.

WIKIPEDIA. *Rickshaw*. 2016. Acessado em 04 nov 2016. Disponível em: <https://en.wikipedia.org/wiki/Rickshaw>. Citado na página 10.