DOUGLAS GALETTI RIBEIRO

TELECONFERÊNCIA 3D EM REDE IP UTILIZANDO VISÃO ESTEREOSCÓPICA

Santo André

Dezembro de 2014

DOUGLAS GALETTI RIBEIRO

TELECONFERÊNCIA 3D EM REDE IP UTILIZANDO VISÃO ESTEREOSCÓPICA

Relatório de Trabalho de Graduação

Relatório apresentado ao Curso de Graduação em Engenharia de Informação da Universidade Federal do ABC, como requisito para aprovação na disciplina Trabalho de Graduação 3.

Orientador:
Prof. Dr. Celso Setsuo Kurashima

UNIVERSIDADE FEDERAL DO ABC

Santo André
Dezembro de 2014

Ficha Catalográfica

Ribeiro, Douglas G.
Teleconferência 3D em Rede IP Utilizando Visão Estereoscópica
44 páginas
Relatório de Trabalho de Graduação em Engenharia de Informação.
Universidade Federal do ABC
Santo André, SP: UFABC, 2014.

Aos meus pais Geasir Ribeiro e Dulcineia Galetti Ribeiro
pelo amor, empenho, dedicação e suporte constante.

AGRADECIMENTOS

À minha noiva Betânia Santos da Silva, pelo apoio, confiança e carinho durante toda a graduação.

À Dario Ribeiro e Justa Trajano por todo incentivo e amor prestado.

Ao orientador Prof. Doutor Celso Setsuo Kurashima, pelo acompanhamento competente e dedicado na elaboração deste trabalho.

Aos amigos da UFABC e do Laboratório de TV Digital da UFABC Josivan Pereira da Silva, João Vitor de Carvalho, Adenilson José Araujo Tomé, Regiane Yuuko Hyodo, Vinícius Ormenesse e Gregory Oliveira.

À Universidade Federal do ABC pela disponibilização de recursos laboratoriais para o desenvolvimento deste trabalho.

A todos os que direta ou indiretamente contribuíram para a realização desta pesquisa.

O verdadeiro teste da vida são nos momentos de pressão, cansaço, tentação, medo e dificuldade;É aí que você mostra se realmente é grande ou não.
(Jeffrey R. Holland)

Resumo

Este projeto propõe um sistema de videoconferência 3D em tempo real através de redes IP de banda larga. O sistema consiste da captura de imagens por meio de duas câmeras em estéreo sincronizadas entre si, e posicionadas lado a lado de frente com a pessoa que irá se comunicar com o lado remoto. O par de imagens são codificadas e transmitidas por rede IP para o receptor. Programas para realizar a captura, transmissão e visualização do vídeo 3D e audio desta vídeo conferência foram desenvolvidos no projeto. A renderização de vídeo 3D é realizada por meio de uma combinação de visão computacional e métodos gráficos. O usuário pode visualizar em tempo real a nuvem de pontos gerada. O processo não necessita de nenhum hardware especializado, apenas computadores pessoais com placas gráficas e webcams reduzindo o custo total do equipamento. Para um par de imagens de 640x480 pixels e um sinal de voz amostrado com 8 bits a 16 kHz, a video conferência 3D foi realizada com banda total de 4 Mbps, e a taxa de quadros da visualização foi de 1 a 3 quadros por segundo. Concluímos que o sistema foi satisfatoriamente implementado quanto à captura, codificação e transmissão dos sinais e necessita de melhorias na renderização e visualização.

Palavras-chave: video conferência , teleconferência, 3D, multiview , codificação , multimídia

Abstract

This project proposes a system for 3D real-time videoconferencing over IP broadband networks. The system consists of image capture using two estéreo cameras synchronized with each other and positioned side by side in front of the person who will communicate with the remote side Te image pair is encoded and transmitted over an IP network to the receiver. A set of programs were implemented for the capturing, transmission and the visualization of 3D video and audio of this video conference system. The rendering of 3D video is performed using a combination of computer vision and graphic methods. The process requires no specialized hardware, only personal computers with graphics cards and webcams reducing the total cost of the equipment. For a couple of 640x480 pictures and a 8 bits voice signal at 16 kHz, the 3D video conferencing was performed using a bandwidth of 4 Mbps and the display frame rate was 1 to 3 frames per second. We conclude that the system was successfully implemented as the capture, coding and transmission of signals and requires some improvements in rendering and visualization.

Keyword: video conference , teleconference, 3D, multiview , codification , multimedia

Lista de Figuras

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Qualquer nova invenção ou tecnologia deve provar a sua utilidade , custo-efetividade e resolver um problema que justifique sua existência. Quando a videoconferência foi introduzida pela primeira vez, em uma grande feira Mundial em 1964 , realizada em Nova York, o sistema tinha uma qualidade que ninguém poderia sonhar que tomaria o lugar do telefone padrão (INERN, 2014). Passado os anos, a década de noventa viu o avanço e desenvolvimento de sistemas de videoconferência devido a muitos fatores, incluindo os avanços técnicos em *Internet Protocol* (*IP*) e também tecnologias de compressão de vídeo mais eficientes foram desenvolvidos que permitiria videoconferência baseada em computador. Não demorou muito tempo e a videoconferência começou a ser utilizada pelas massas através de serviços gratuitos e de software, tais como o *NetMeeting, MSN Messenger e Yahoo Messenger*[1]*, Skype*[2] e outros. Embora vídeo com distoções e imagens pouco nítidas, a tecnologia começou lentamente a ser adotada pelo mercado consumidor (NEFSIS, 2014).

Ainda na década de noventa uma combinação de especificações e normas abriu o caminho para a videoconferência como uma aplicação confiável no mundo dos negócios. A ITU (International Telecommunication Union Telecommunication Standardization sector) estabeleceu o padrão H.263 de compressão de vídeo , o que reduziu a transmissão de banda para comunicações de baixas taxa de bits. Outras normas já estavam em desenvolvimento, como o H.323 para aplicações multimídia baseadas em pacotes. Tanto H.263 e H.323, e foram revistos e atualizados em 1998. The Moving Picture Experts Group desenvolveu MPEG-4 como um padrão ISO para conteúdo multimídia e, embora não diretamente relacionado com a videoconferência como uma aplicação desktop, todas as normas mencionadas trabalharam em conjunto para avançar ainda mais o conceito de interoperabilidade em relação ao conteúdo dos dados e sua transmissão (NEFSIS, 2014).

A videoconferência foi demonstrada pela primeira vez em 1968, mas ganhou destaque após a "Guerra do Terror" depois que repórteres de TV utilizaram um satélite portátil e um videofone para transmissão ao vivo do Afeganistão em Outubro de 2001. Foi a primeira vez que a tecnologia de videoconferência foi usada para falar em tempo real com alguém na zona de guerra através de som e vídeo. Pela primeira vez , a guerra foi transmitida em tempo real para pessoas ao redor do mundo, em vez de ser por transmissão gravada (NEFSIS, 2014).

O acesso à Internet de alta velocidade tornou-se amplamente disponível a um custo razoável. Ao mesmo tempo, o custo da captação de vídeo e de tecnologia de exibição também diminuiu . O público em geral foi capaz de comprar câmeras para o computador, o custo dos computadores diminuiu e o acesso à Internet de banda larga estava disponível em quase todas as regiões dos EUA. Todos esses fatores, incluindo a disponibilidade de software livre a partir dos principais provedores de serviços de mensagens instantâneas, se combinaram para tornar a videoconferência ainda mais acessível para o mercado consumidor .

O ensino superior começou a abraçar os benefícios da videoconferência em meados de 2003. Escolas de todo o mundo começaram a integrar videoconferência em seus programas de ensino à distância. Como a

[1] https://br.messenger.yahoo.com

[2] www.skype.com/pt-br

qualidade de transmissão de vídeo aumentou, a videoconferência tornou-se muito mais popular nas escolas. Administradores e professores perceberam os imensos benefícios de interação em tempo real entre professores e alunos, como realização de aulas e apresentações. *VBrick* começou a fornecer vários sistemas MPEG- 4 de vídeo para as universidades nos EUA e foi rapidamente ganhando importância e popularidade como um meio eficaz de se comunicar e interagir em tempo real para as organizações empresariais de todo o país (INERN, 2014) . Em 2004 empresas de videoconferência continuaram refinando suas aplicações e ajustando-as para obter usabilidade e desempenho mais confiáveis. Em março de 2004 , uma plataforma de vídeo conferência livre baseado em Linux, compatível com H.323 e *NetMeeting*, foi lançada (NEFSIS, 2014).

Com o avanço das tecnologias de software, a velocidade do processador dos computadores e banda larga com Internet, tornou-se possível utilizar inteiramente servidores e computadores convencionais para realizar videoconferência de qualidade (REALPLAYER, 2014), isto é, transmissão em tempo real de imagens com definição, som com poucos ruídos e atraso na comunicação entre participantes reduzido. Contudo, os sistemas baseados em hardware estão disponíveis hoje. Sistemas de videoconferência dedicados geralmente tem todos os componentes necessários em um único equipamento (REALPLAYER, 2014).

A indústria de videoconferência trouxe algumas atualizações recentes notáveis como chamadas de vídeo através de redes sociais. A videoconferência 3D, no entanto, pode fornecer novos elementos as chamadas de vídeo como a captura de gestos do usuário como o olhar, gestos do corpo, boca etc. O que acontecer na videoconferência 3D vai pavimentar o caminho para video chat 3D (REALPLAYER, 2014).

A videoconferência vai continuar a evoluir até se tornar uma parte integrante da vida profissional e pessoal. Como a tecnologia tem sofrido novas transformações, sem dúvida, vai se tornar mais acessível e eventualmente, onipresente, utilizada em grande escala (NEFSIS, 2014) (REALPLAYER, 2014).

Diversos programas de videoconferência foram desenvolvidos ao longo dos últimos anos como *Skype, Hangouts*, entre outros. Embora sejam sistemas que cumpram seus objetivos, não estão sob regime de software livre e seus codificadores não podem ser estudados e modificados de maneira a favorecer um nova implementação ou um estudo acadêmico sobre os mesmos.

O presente projeto visa a criação e implementação de um sistema de videoconferência utilizando softwares livres a partir do sistema desenvolvido em pesquisas anteriores (RIBEIRO; KURASHIMA, 2012). Também maior imersividade será adicionada as chamadas por vídeo, de maneira tal que os indivíduos participantes sintam maior proximidade e possam interagir entre si. Assim, uma videoconferência 3D será implementada com base em estudos anteriores (KURASHIMA *et al*, 2002) e utilizando técnicas que capturem imagens de diversos ângulos de um participante, envie esses dados a outro computador pela rede local e construa um objeto 3D do mesmo com softwares de renderização gráfica (RUIGANG *et al*, 2007).

2. OBJETIVOS

O projeto deste trabalho de graduação de Engenharia de Informação tem como objetivo o desenvolvimento de um protótipo de videoconferência 3D em tempo real através da rede IP de banda larga. Algumas metas específicas deverão ser alcançadas para implementação do sistema:

- Estudo da transmissão de dados multimídia e múltiplos videos sincronizados utilizando a API GStreamer de maneira a garantir um desempenho geral satisfatório do sistema;
- Implementação de renderização do vídeo em objeto 3D utilizando um par de câmeras estéreo e técnicas de computação gráfica para exibição do modelo tri-dimensional.

Ao fim do projeto, espera-se um sistema de videoconferência 3D funcional e com desempenho satisfatório.

3. ESTADO DA ARTE

Avanços em tecnologia de rede, câmeras e monitores tornaram possível uma nova gama de aplicações para comunicação 3D, como TV 3D *freeview point*, videoconferência imersiva, etc. Em geral, nessas aplicações são utilizadas múltiplas câmeras afim de obter a mesma imagem, mas com diferentes pontos de observação. Essas imagens são transmitidas de um ponto a outro em tempo real ou não, porém, existem fatores que contribuem direta ou indiretamente a qualidade de execução do sistema e muito tem sido estudado a respeito.

Em vídeos *Multiview Video* - vídeos com múltiplos pontos de observação - a compressão é essencial para prover comunicação 3D de qualidade e diversos estudos sobre codificação preditiva para vídeos *multiview* tem sido feitos, principalmente devido a grande redundância entre vídeos com diferentes pontos de observação. Uma pesquisa realizada pela Microsoft (FLORENCIO, 2008) aborda este assunto e definem pontos importantes sobre o problema, mostrando um algoritmo simples que consegue reduzir a taxa de bits em até duas vezes. O algoritmo desenvolvido estima a posição do observador para computar a contribuição de cada cada pixel na imagem sintetizada, fazendo a codificação de cada macro-bloco de cada vista de câmera com qualidade proporcional à probabilidade de que o pixel será utilizado na imagem sintética.

De forma geral, o participante remoto envia seu atual ponto de observação e informação sobre movimentação (site A). O ponto de observação é então renderizado sendo feito o controle de compressão adaptativa para, dessa forma, maximizar a eficiência de codificação e minimizar a largura de banda utilizada (site B) como visto na Figura 3.1.

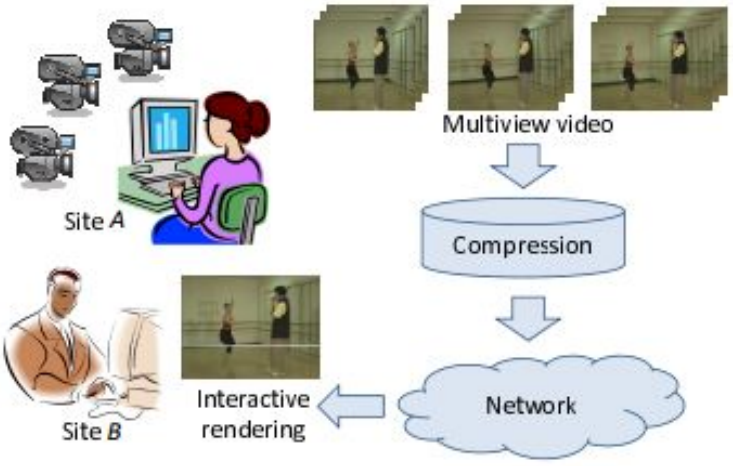

Figura 3.1: Cenário de tele conferência imersivo (FLORENCIO; ZHANG, 2008).

Os equipamentos utilizados em uma videoconferência são muito importantes para que o sistema funcione perfeitamente e um dos maiores desafios na implementação de um sistema de videoconferência é a complexidade computacional empregadada, principalmente em um sistema 3D e imersivo, pois é gerado um modelo 3D a partir dos vídeos capturados. Pesquisadores da Alemanha (FELDMANN *et al,* 2010) estudaram o assunto e propuseram uma forma de alta performance eficiente para adquirir os quadros utilizados por um sistema de videoconferência 3D, levando em consideração o uso de GPU ou de CPU com vários núcleos de processamento. Conseguiram utilizar algoritmos complexos, capturando e transmitindo dados de 16 câmeras HD em tempo real como visto na Figura 3.2.

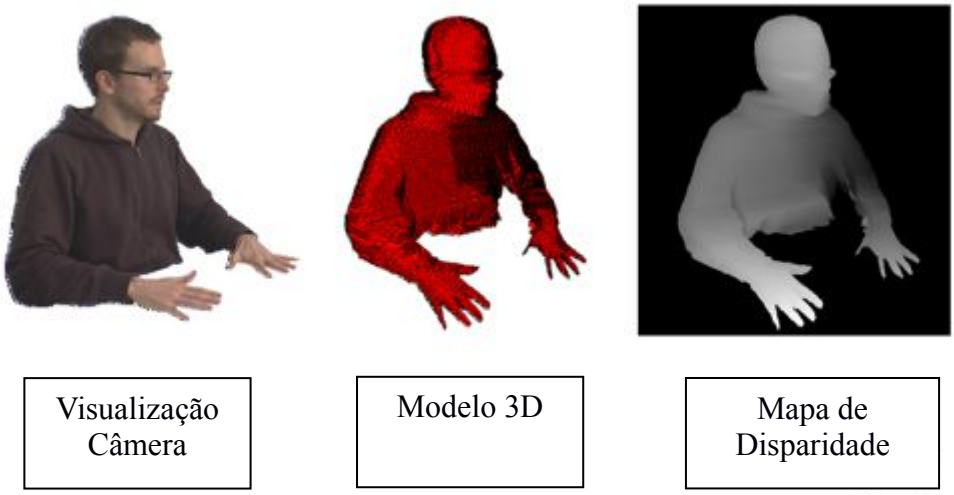

Figura 3.2: Resultado da captura de imagens de 16 câmeras e posterior processamento (FELDMANN; WAIZENEGGER; ATZPADIN; SCHREER, 2010).

Outro grupo de pesquisadores procuraram criar uma cópia virtual dos participantes e inseri-los em um ambiente tridimensional (WEIK *et al,* 1997). Basicamente, os modelos foram previamente criados sendo compostos de duas formas básicas: o esqueleto e uma espécie de superfícies triângulares para movimentação.

Capturadas imagens em estéreo, estas são adaptadas ao esqueleto do modelo e as texturas são aplicadas. Assim, ao final do processo é gerado um modelo flexível de um participante e um plano de fundo virtual pode ser inserido para maior imersividade como visto na Figura 3.3.

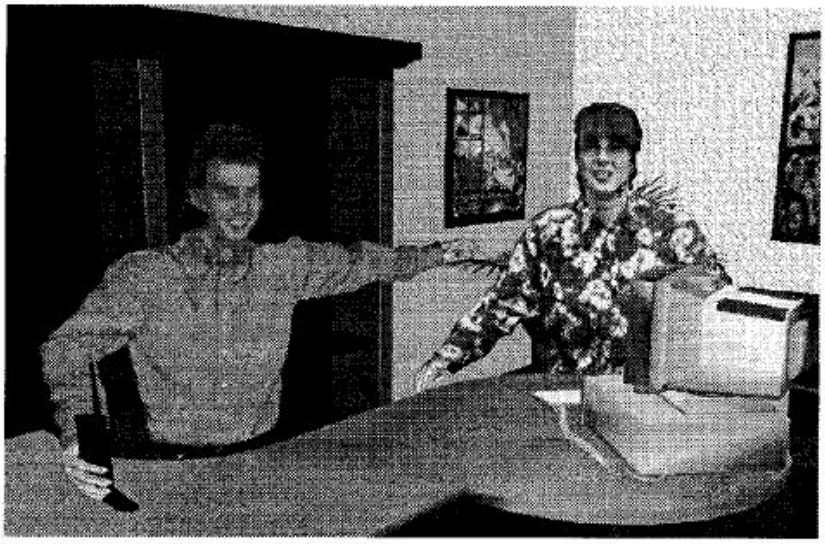

Figura 3.3: Ambiente virtual 3D com modelos tridimensionais de cada participante (WEIK; WINGBERMUEHLE; KOPERNIK, 1997).

Em outro projeto foi construído modelos faciais 3D de forma automática. Eram capturadas uma fotos frontais e laterais de uma pessoa com dimensões 512x480 pixels assegurando que o tamanho da face em cada imagem eram iguais. O sistema implementado era capaz de identificar o fundo da imagem, cabelo e face do participante. Com um template de modelo, 50 pontos eram inseridos de forma a definir as regiões dos olhos, nariz, boca, queixo etc. Essas imagens então são dispostas nas regiões especificadas (AKIMOTO *et al*, 1993) na Figura 3.4.

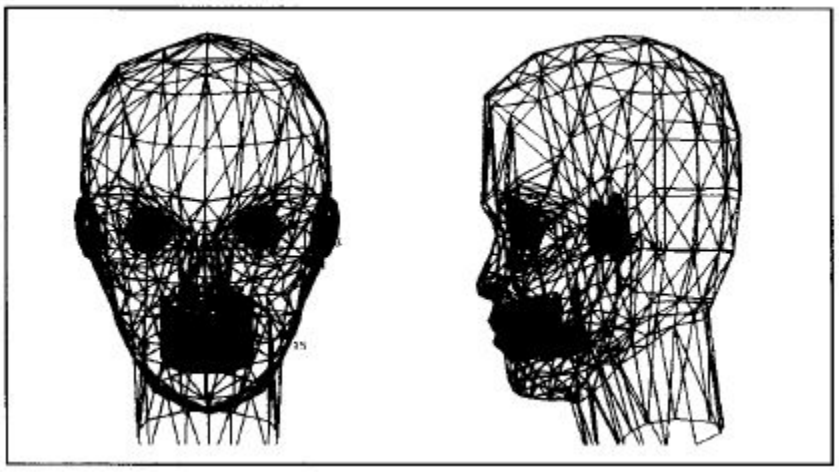

Figura 3.4: Modelo facial 3D resultante de modificação do modelo (AKIMOTO; SUENAGA; WALLACE, 1993).

Outros métodos de construção de objetos 3D foram estudados, um deles de grande importância para este projeto. Um grupo de pesquisadores da Universidade de Hannover (NIEM; WINGBERMUHLE, 1997) implementaram um sistema de construção automática de objetos 3D utilizando uma câmera de celular. O sistema implementado permite que que cada foto de diferentes ângulos possam ser calibradas individualmente obtendo o objeto e seu padrão simultaneamente. Dessas imagens um modelo 3D pode ser

estimado através das silhuetas das imagens como visto na Figura 3.5. O sistema consegue gerar objetos com boa aproximação, além de reduzir custos, pois não há necessidade de equipamentos específicos.

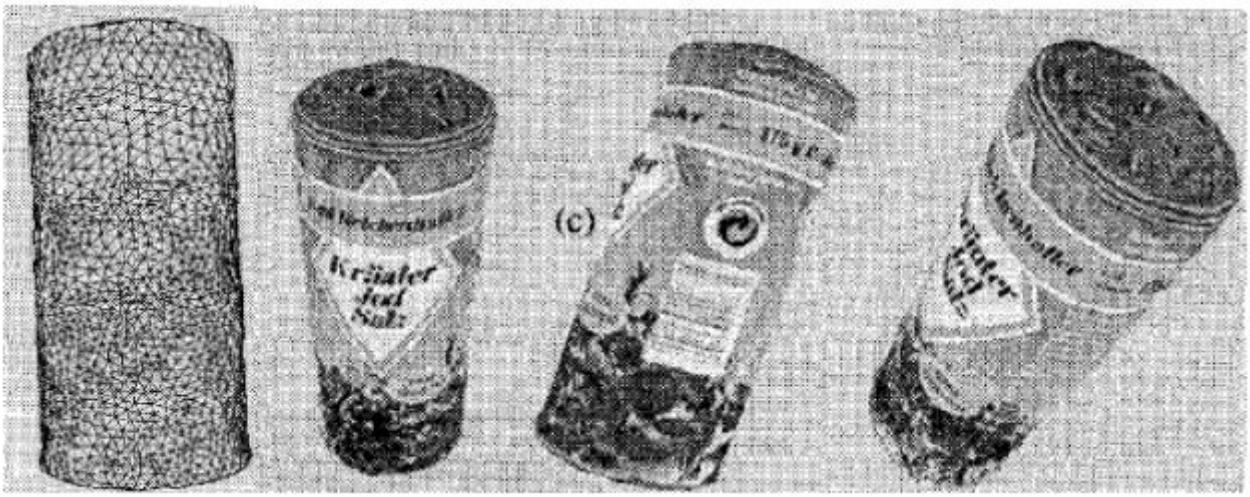

Figura 3.5: Modelo sintetizado sem textura e com textura em diferentes ângulos (NIEM; WINGBERMUHLE, 1997).

4. ESPECIFICAÇÃO DOS REQUISITOS DO PROJETO

Um dos principais requisitos do projeto é a implementação com menor custo possível e livre de quaisquer restrições devido a direitos autorais e patentes. Assim, todo o sistema foi desenvolvido em regime de código livre (GNU). Foram utilizados o sistema operacional Ubuntu 14.04 LTS e o codificador de voz em regime de software livre Speex (Speex, 2014), em container de extensão .ogg. A linguagens de programação para implementação do sistema foram Python e Shell (Linux).

Foram utilizadas duas webcams convencionais para obtenção das imagens. A resolução máxima de cada uma era de 640x480 pixels a 30 quadros por segundo. Diversos testes foram realizados com o intuito de estabelecer uma distância apropriada entre as câmeras e a qualidade obtida na obtenção do modelo tri-dimensional. Na obtenção de voz, foi utilizado um microfone convencional com taxa de bits de 15 kbps e taxa de amostragem de 16 kHz. Ainda na fase de testes, o programa Meshlab (Meshlab, 2014) foi utilizado para visualizar os modelos gerados.

Por se tratar de uma videoconferência, toda informação foi transmitida a cada um dos participantes utilizando o protocolo de rede UDP por ser mais eficiente em transmissão em tempo real. A banda utilizada para a transmissão foi a disponível nas dependências da UFABC. Para a construção do modelo 3D de um dos participantes, foi implementado um programa que recebe as imagens enviadas pela rede e as processa de forma a gerar um objeto tri-dimensional e exibir ao usuário. Os programas e dispositivos utilizados, assim como suas caracterísicas são caracterizados na Tabela 4.1.

Tabela 4.1: Principais dispositivos e software utilizados.

Dispositivo / Software	Uso	Características
Webcam	Captura de imagens	640x480, 30 FPS
Microfone genérico	Captura de Voz	Codificação a 15 kbps, 16 kHz
Gstreamer 1.0	Codificação Imagem e Voz	Codificador Speex, protocolo UDP
Protocolo UDP	Transmissão de dados	Mais eficiente que TCP
Rede IP Local Banda Larga	Transmissão de Dados	110 Mbps Download, 97 Mbps Upload
Meshlab	Construção Objeto 3D	Utiliza imagens/vídeo para construir objetos 3D

Todo processo de codificação, transmissão, recepção e decodificação de vídeo e voz foram realizados utilizando a API GStreamer presente nos repositórios do Ubuntu. Esta possui uma série de recursos multimídia ja implementados e podem ser utilizados como em um diagrama de blocos, sendo necessário o conhecimento de algumas informações de configuração.

5. ALTERNATIVAS E SOLUÇÕES

Diferentes soluções foram abordadas para atingir a meta do projeto. Dentre elas destaca-se a obtenção da nuvem de pontos utilizando algoritmos como Bundler e PMVS, assim como utilizar hardwares específicos e suas bibliotecas como o Microsoft Kinect. A Tabela 5.1 apresenta o cronograma de atividades realizadas no período de implementação do projeto.

Tabela 5.1: Principais atividades durantente o trabalho de graduação

Atividade	Mês (a partir do ínicio das atividades)											
	1	2	3	4	5	6	7	8	9	10	11	12
Revisão Bibliográfica	x	x	x	x	x	x	x	x				
Experimentos com Bundler				x	x	x	x					
Experimentos com Kinect						x	x					
Experimentos com câmeras stereo							x	x	x	x	x	x
Video conferência experimental											x	x
Análise de desempenho											x	x
Elaboração do relatório			x	x			x	x	x	x	x	x

5.1 Bundler e PMVS

A primeira abordagem de gerar um modelo tri-dimensional de face humana utilizando diversas câmeras foi obter por processamento de determinados algoritmos uma nuvem de pontos e convertê-la em um objeto tri-dimensional. Entretanto, algoritmos que realizam este tipo de processo são muito complexos e é requerido computadores de alto desempenho pois processos deste tipo realizam geração de mapas de disparidade, calculam coordenadas da imagem para mundo real entre outros.

Utilizando o algoritmo e programas Bundler e PMVS (FURUKAWA; 2010), foi possível, a partir de diferentes fotos capturadas de diferentes ângulos, gerar um modelo tri-dimensional de um objeto ou pessoa. Os resultados obtidos foram satisfatórios para objetos, mas não para face humana (ver seção 9).

O algoritmo do Bundler compara as diferentes imagens capturadas e gera uma nuvem de pontos esparsa apenas com os pontos em comum das imagens e que traçam as características principais do objeto como a forma dele. Depois de definido esses pontos, o algoritmo PMVS utiliza estes pontos e os analisa comparando pixel a pixel e definindo posição nos eixos XYZ e as cores no padrão RGB. Caso não encontre correlação entre pixels, nenhuma informação é armazenada e a nuvem de pontos é gerada com falta de informações, deixando o modelo tri-dimensional obtido com a face do usuário incompleta.

Observando os resultados obtidos ao fim do processo, para objetos o algoritmo do Bundler consegue gerar uma nuvem de pontos com mais informação porque consegue definir diferentes parâmetros para cada pixel. Por exemplo, em uma camisa listrada o algoritmo tem facilidade em definir quais pixels estão na mesma vizinhança, associando-os. Em texturas ruins ou contínuas como, por exemplo, um fundo branco ou mesmo regiões da face de uma pessoa, informações não são obtidas porque o algoritmo não consegue correlacionar os respectivos pixels, pois sua vizinhança apresenta pixels muito semelhantes.

Em objetos, a nuvem de pontos gerada é satisfatória, com grande número de pontos; entretanto, para faces, muitos pontos são perdidos devido a não correlação, causando perda de informações a aparição de regiões incompletas como visto nas Figura 5.1, Figura 5.2 e Figura 5.3.

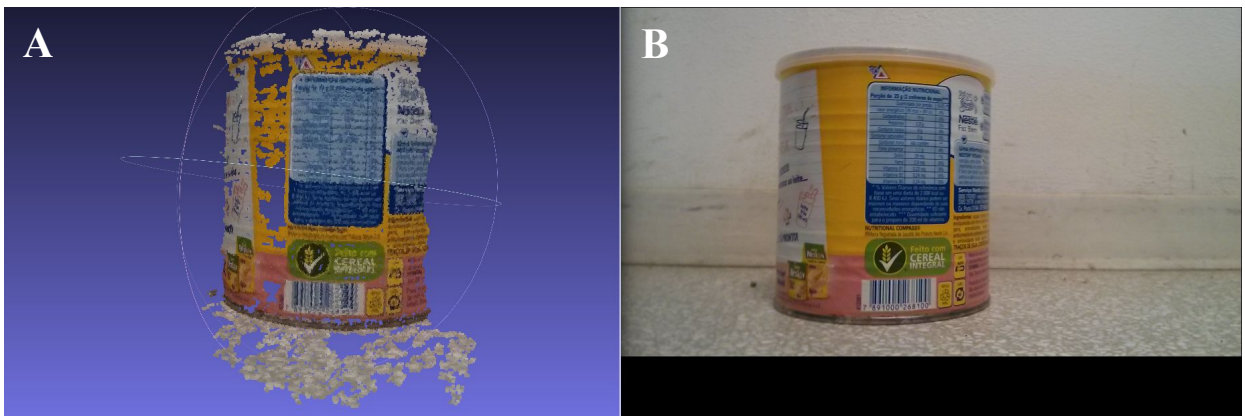

Figura 5.1: (A) Nuvem de pontos gerada a partir de 10 fotos tiradas de uma lata de Neston (B) compreendendo180 graus frontal.

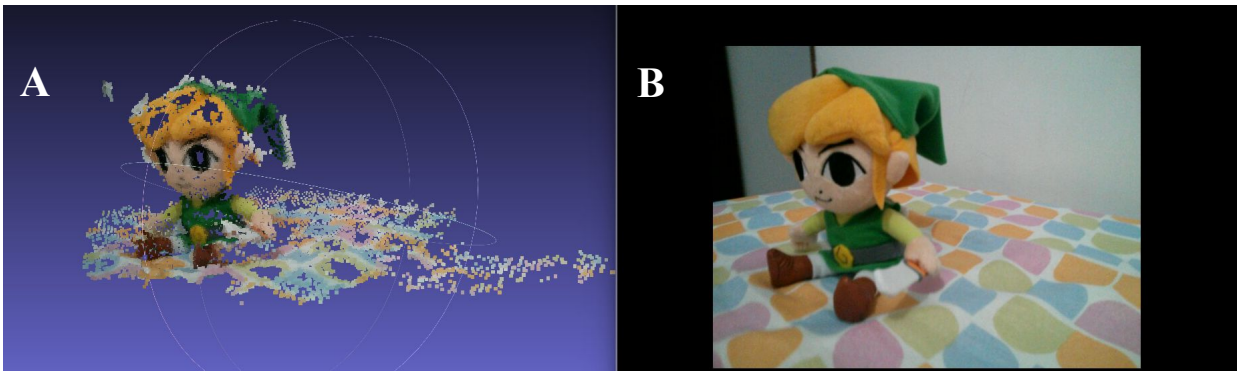

Figura 5.2: (A) Nuvem de pontos gerada a partir de 15 fotos tiradas do boneco Link (B) compreendendo 180 graus frontal. Fonte: o autor.

Os resultados obtidos foram satisfatórios para objetos, porém, para o escopo do projeto que é a construção de uma face humana tri-dimensional, o objetivo não foi alcançado. Outro motivo importante é que a obtenção da nuvem de pontos demorava entre um a 3 minutos dependendo do número de fotos utilizadas, o que invalida um dos objetivos do projeto que é a comunicação (envio da nuvem de pontos e atualização dela) em tempo real.

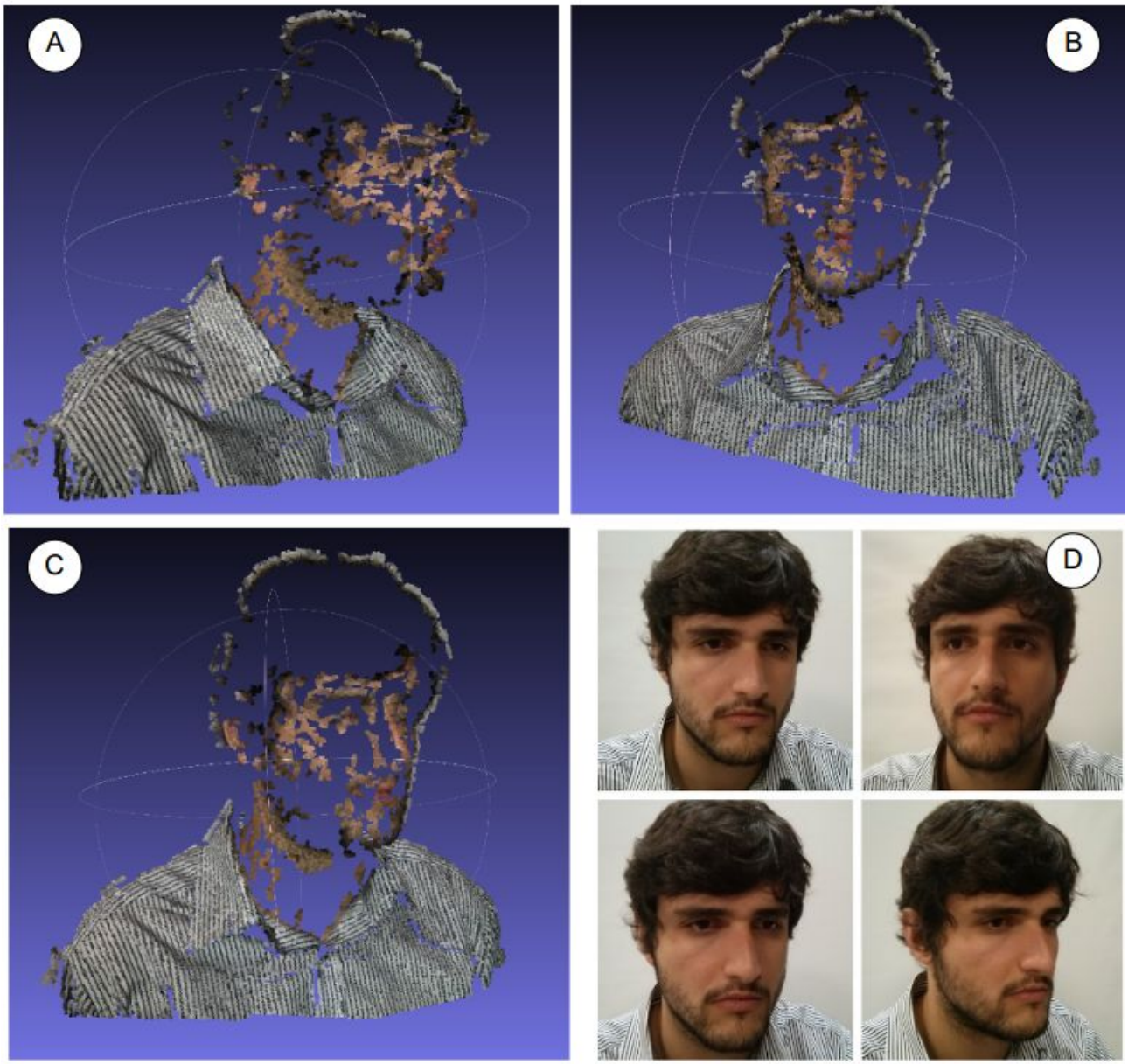

Figura 5.3: (A) Visão lateral direita , (B) visão frontal e (C) visão da lateral direita e parte da lateral esquerda do participante. (D) Corresponde a algumas das 25 fotos capturadas do participante. A qualidade da camisa ilustra a dificuldade em obter pontos para face, mas a facilidade em objetos com texturas. Fonte: o autor.

5.2 Microsoft Kinect

Outra abordagem para obter a nuvem de pontos foi utilizar um hardware específico. A câmera, que possui detecção de vídeo em RGB, combinando vermelho, verde e azul para formar as imagens captadas em uma resolução de 640x480 a 30 quadros por segundo (FPS). Há sensores de profundidade 3D que usam em conjunto um projetor infravermelho e um sensor CMOS monocromático para projetar o ambiente em 3D e perceber as suas modificações (APC, 2011). Assim, a nuvem de pontos é gerada com maior precisão do que se utilizada câmeras tradicionais como no exemplo anterior com o Bundler na Figura 5.4.

A nuvem de pontos, devido a precisão em sua obtenção possui poucos pontos considerados ruídos, ou seja, pontos que não fazem parte do rosto ou estão com coordenadas erradas. Por esta razão é possível usar a

nuvem de pontos de forma a troná-la um objeto tri-dimensional. Este é visualizado utilizando a ferramenta MeshLab. Entre outros tipos de extensões de arquivo, a ferramenta utiliza o arquivo de extensão *.ply* da nuvem de pontos gerado ao final do processo anterior e renderiza utilizando triangulação de Poisson, criando um modelo 3D da face do usuário como visto na Figura 5.5.

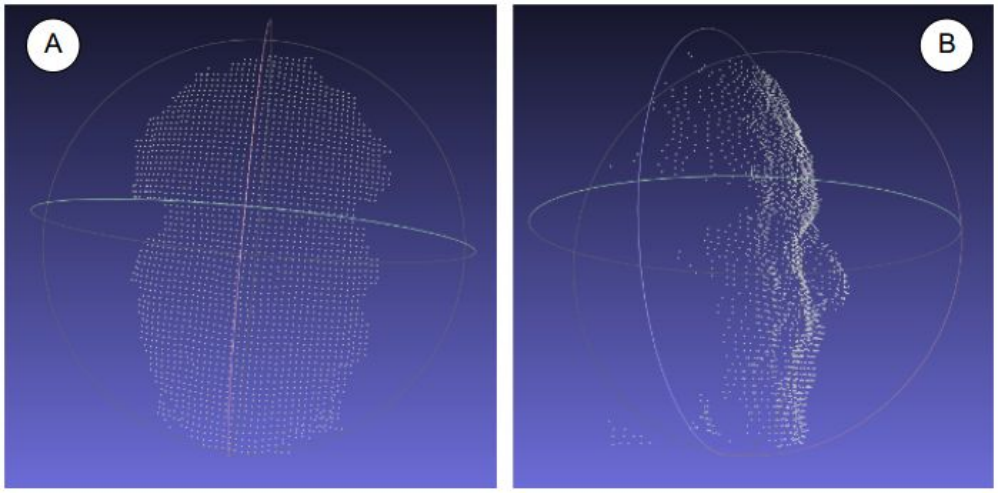

Figura 5.4: (A) Visão frontal e (B) visão lateral direita da nuvem de pontos normatizados. Fonte: o autor.

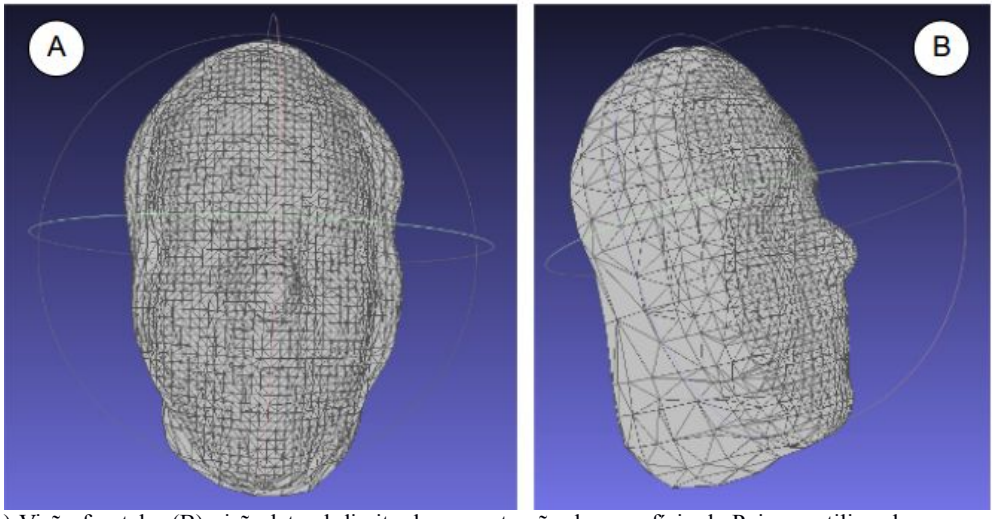

Figura 5.5: (A) Visão frontal e (B) visão lateral direita da reconstrução de superfície de Poisson utilizando as nuvens de pontos. A face do participante foi gerada. Fonte: o autor.

A reconstrução de superfície de Poisson é uma abordagem que expressa a reconstrução da superfície como a solução para uma equação de Poisson. Este método utiliza uma solução implícita para aproximar a superfície e, em seguida, extrair a isosuperfície usando uma adaptação do algorítimo de Marching Cubes (LORENSEN *at al*, 1987). Uma vez que as soluções precisas são necessárias apenas perto da superfície, uma estrutura octree é aplicada para aproximar os dados brutos e, escolhendo a máxima profundidade octree, pode-se criar superfícies de diferentes LOD - Level of Detail como visto na Figura 5.6.

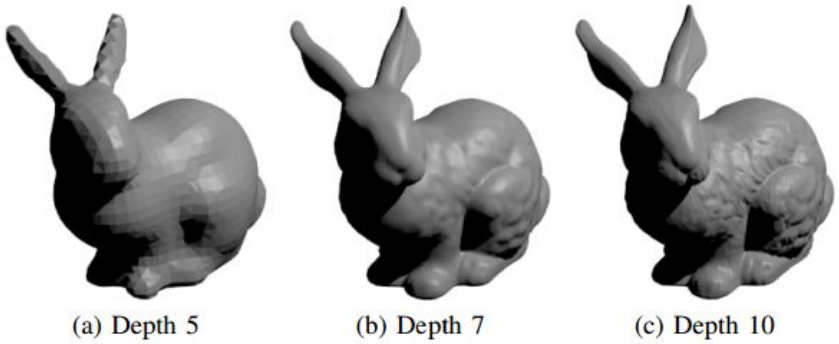

Figura 5.6: Coelho reconstruído com o método de Poisson com três resoluções distintas. (A) corresponde a resolução com depth igual a 5 ; (B) com depth igual a 7; (C) com depth igual a 10. (PIAZENTIN *et all*, 2012).

Em uma videoconferência, alto nível de detalhe é importante, pois pode construir o objeto (face do usuário) de forma mais detalhada e fiel. Em compensação, se o nível de detalhe for menor, a renderização do objeto é mais rápida e contribui para sua transmissão em aplicações baseadas em rede.

Outra tentativa foi utilizar apenas a nuvem de pontos. Em linguagem de programação Java foi implementado um visualizador simples que permitia a interação (rotação, zoom) da nuvem de pontos e removia o fundo, gerando a nuvem de pontos apenas da face do usuário como pode ser visto na Figura 5.7.

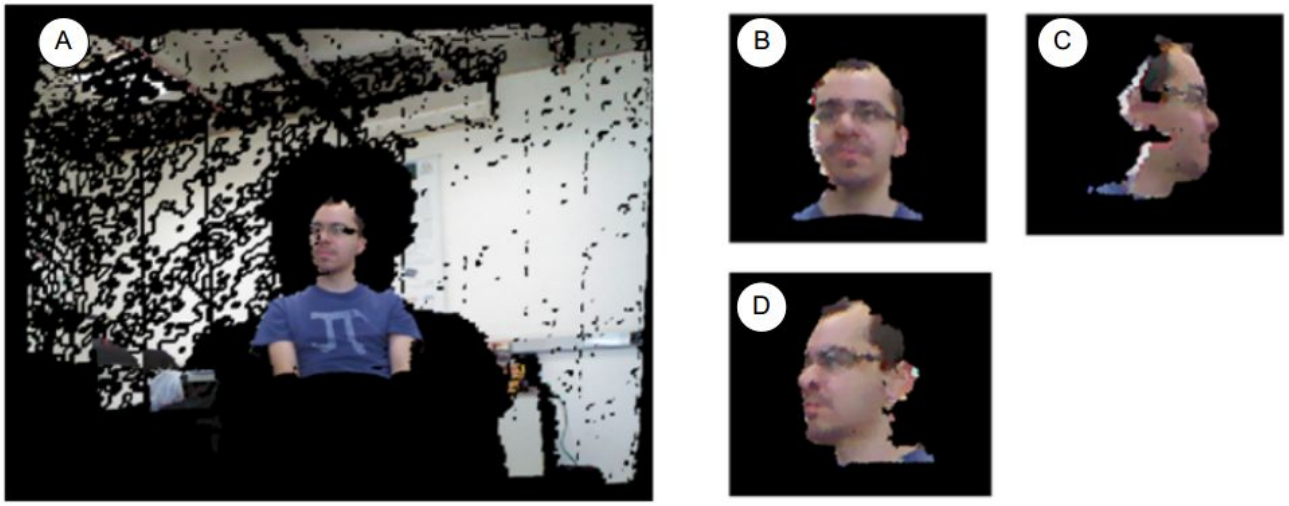

Figura 5.7: (A) Nuvem de pontos gerada pelo Kinect é processada de forma a gerar uma versão tri-dimensional do participante. É feita a detecção da face de forma que o fundo todo é deletado como visto na (B) visão frontal, (C) visão lateral direita e (D) visão lateral esquerda do participante. Fonte: o autor.

A nuvem de pontos e o objeto tri-dimensional foram obtidos de forma satisfatória, porém invalidam uma das propostas do projeto que é o não uso de equipamentos e dispositivos específicos. Neste caso, a construção tri-dimensional da face da pessoa só pode ser obtida se utilizado o dispositivo Kinect.

5.3 Comparativo Bundler + PMVS e Kinect

Os algoritmos Bundler e PMVS conseguem construir uma nuvem de pontos a partir de diferentes imagens. Os resultados obtidos para objetos foram satisfatórios, porém, para rostos os resultados foram ruins, pois a face não foi contruída e informações perdidas.

Utilizando o Kinect foi possível obter a nuvem de pontos da face, porém apenas frontalmente. Os resultados obtidos foram satisfatórios.

Comparadas as duas tecnologias, o Bundler + PMVS obteve um resultado tri-dimensional mais amplo, ou seja, a construção de um objeto 3D completo, porém, sem a face da pessoa. O Kinect apresentou a construção de uma nuvem de pontos da face completa, mas apenas da face frontal. Não gerou um objeto 3D completo como no Bundler + PMVS.

Devido aos resultados obtidos a solução adotada pelo projeto foi a utilização de câmeras estéreo que atuariam semelhante ao processo do Kinect. Detalhes a respeito do funcionamento do projeto estão na seção 7 e subseções.

6. ANÁLISE E VIABILIDADE DO PROJETO

O projeto tem como base resultados obtidos em projeto de iniciação científica realizados anteriormente, tendo sido estudados os codificadores de vídeo e voz, Theora e Speex respectivamente, assim como a API GStreamer para codificação e envio de informações. Ao final do último projeto foi desenvolvido um programa para videoconferência em Java na plataforma Linux. Portanto, informações relevantes foram pesquisadas sobre o assunto (RIBEIRO *et al*, 2012) e foram aproveitadas na implementação deste projeto.

6.1 Análise Técnica

As webcams foram instaladas e configuradas no ambiente Ubuntu 14.04 LTS. Elas foram presas a um suporte em alumínio acoplado a um tripé de forma a deixá-las horizontalmente no mesmo nível. As lentes foram ajustadas manualmente e as imagens calibradas por um software implementado em Python. As fotos obtidas foram enviadas utilizando protocolo UDP pela rede local a um outro computador que realizou a construção do objeto tri-dimensional utilizando programas também implementados.

Com a implementação da câmera e da captura de voz completa, a nuvem de pontos gerada pela análise das fotos obtidas pelas webcams são transmitidas ao outro usuário. As imagens registrarão a face do usuário que terá sua face construída sob um objeto 3D

A implementação do sistema depende de uma série de fatores e riscos subsequentes que podem determinar a viabilidade do projeto. Pelo menos dois fatores importantes devem ser considerados:

- A configuração e sincronização das webcams em ambiente Linux pode ser inviável devido a uma possível falta de drivers e suporte das câmeras utilizadas.

- Gerar um objeto 3D a partir de imagens é uma tecnologia de difícil implementação
- Os resultados obtidos podem não ser satisfatórios o suficiente para uma video conferência em tempo real.

Riscos menores também estão envolvidos como:

- Banda larga disponível insuficiente para bom funcionamento do sistema;
- Computadores utilizados para transmissão com hardware inferior ao mínimo necessário para codificação e decodificação dos dados;

6.2 Análise Econômica

A implementação do sistema de videoconferência 3D neste projeto tem seus custos reduzidos, pois os equipamentos necessários estão disponíveis em um dos laboratórios de Engenharia de Informação da UFABC. Não haverá gastos com computadores, cabos, monitores etc.

7. DESCRIÇÃO SISTEMICA DO PROJETO

O sistema implementado deve executar uma video conferência 3D. Um modelo tri-dimensional do participante deve ser criado e exibido ao outro usuário de forma que este usuário possa interagir com o modelo (zoom, rotação, diferentes ângulos de visão etc). Para que o sistema realize a tarefa, alguns passos são necessários: captura e transmissão de imagens, nuvem de pontos, sinal de voz; e geração e renderização remoto do modelo tri-dimensional.

As subseções a seguir detalham cada etapa do processo de criação da face tri-dimensional do usuário e transmissão pela rede IP local.

7.1 Captura e Transmissão de Voz

A voz do usuário é capturada por um microfone convencional, codificado em formato Speex e enviado ao computador do usuário de destino (em rede local) pelo protocolo UDP utilizando a API GStreamer. O outro computador recebe os dados, decodifica e emite o som através de alto falantes ou fone de ouvido conforme ilustrado na Figura 7.1.

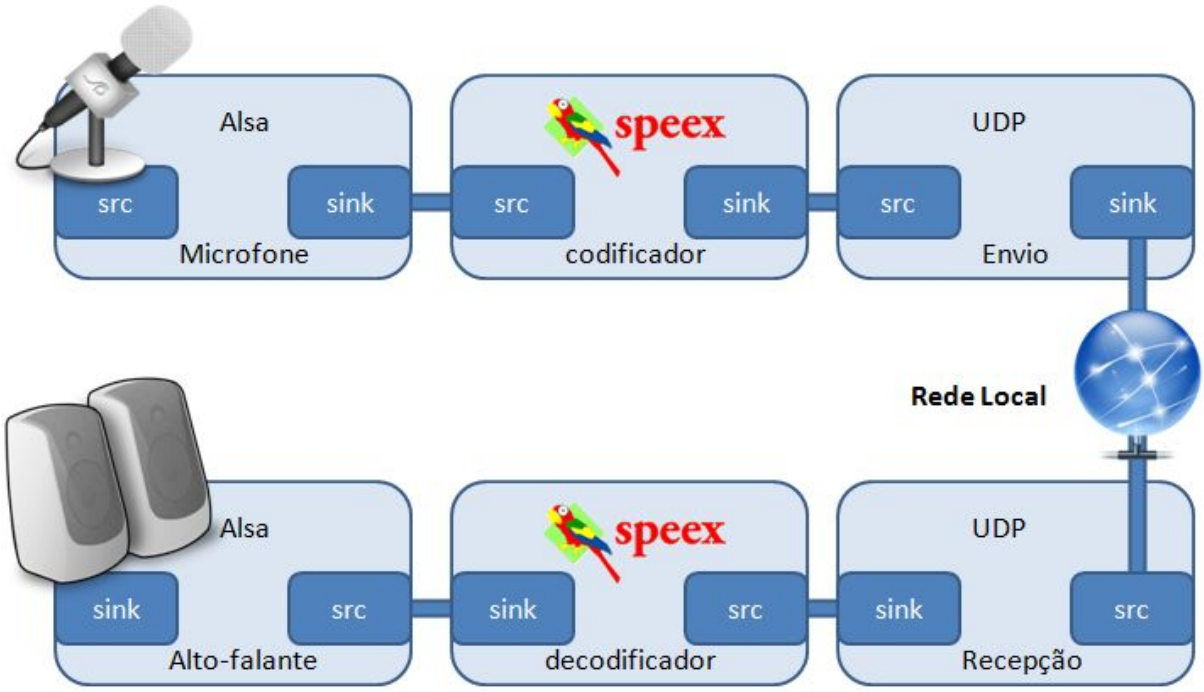

Figura 7.1: Ilustração funcionamento basico do sistema de voz. Voz capturada pelo microfone é codificada em format Speex e é enviado através do protocolo UDP para outro computador em rede local. Fonte: o autor.

7.2 Captura de Imagens e Calibração

As webcams são dispostas uma ao lado da outra a uma distância de 12 cm, pois apresentou os melhores resultados obtidos nos testes realizados (ver seção 8). Elas são ajustadas manualmente de forma que a altura, foco e ângulo das câmeras sejam semelhantes. O objetivo é capturar as imagens em estéreo, ou seja, a visualização de um mesmo foco por dois mecanismos de captação de imagens conforme Figura 7.2. Em seres humanos, diz-se que a imagem percebida pelo cérebro resulta da combinação de duas imagens captadas uma em cada olho (FILHO *et al*, 2008).

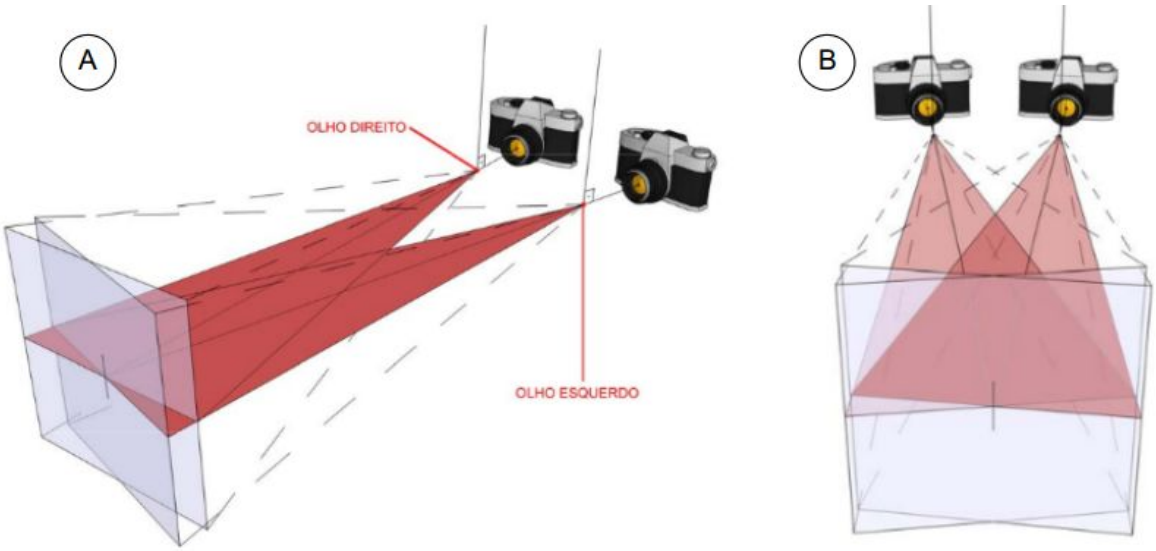

Figura 7.2: Mecanismos para captação de imagens com focos visuais coincidentes. (A) corresponde ao mecanismo visto da lateral e (B) o mesmo mecanismo visto de frente . Esquema análogo aos olhos humanos (FILHO, 2008).

As duas imagens são combinadas por um processo de retificação e obtem a posição dos pixels da imagem da esquerda com o pixel correspondente na imagem da direita. Com este método é possível calcular a distância do pixel a câmera. A profundidade é traduzida em um mapa de disparidade em que pontos são mostrados em escala de cor cinza, sendo cores próximas a branca correspondentes a pixels próximos as câmeras e cores escuras tendendo a preto, pixels longe das câmeras.

Para que o mapa de profundidade gerado a partir das imagens capturadas seja satisfatório, é necessário que as câmeras sejam ajustadas, porém quando em visão estéreo é necessário saber a relação espacial entre as duas câmeras afim de que sejam corrigidas as distorções das lentes. O processo pode ser automatizado utilizando o algoritmo de Zhang mostrando a ambas câmeras um tabuleiro de xadrez, pois o algoritmo de calibração detecta os vértices de cada quadrado em cada imagem capturada de modo a fazer a calibração posteriormente. Com a disparidade entre os pontos observados em cada câmera é possível calcular a matriz de rotação e translação que transforma o sistema de coordenadas da câmera da esquerda para o sistema de coordenadas da câmera da direita (DRÖPPELMANN *et al*, 2010).

O processo de calibração de câmeras foi realizado com as bibliotecas do OpenCV para Python que possuem funções especialmente para câmeras em estéreo. A biblioteca possui uma função para calibrar as câmeras, usando uma lista de coordenadas de pontos na imagem da esquerda e as coordenadas dos mesmos pontos na imagem da direita, resultando em coordenadas que os pontos realmente estão no mundo real.

Os pontos utilizados são as interseções de um tabuleiro de xadrez como visto na Figura 7.3. Esses pontos são reconhecidos pelas câmeras e uma lista de coordenadas é retornada. Esta lista gerada corresponde a uma série de arquivos com coeficientes de distorção de cada câmera para corrigir a distorção das lentes e matrizes de translação/rotação relacionando a câmera da esquerda com a da direita.

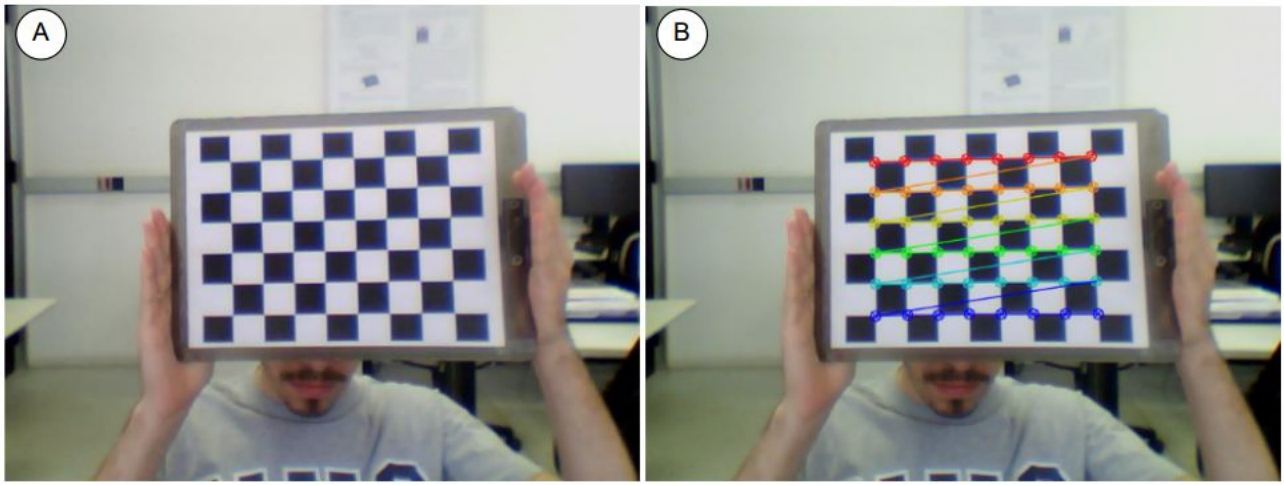

Figura 7.3: (A) Imagem capturada pela câmera e (B) pontos do tabuleiro de xadrez detectados pelo OpenCV. Fonte: o autor.

Depois de calibradas, as câmeras capturam fotos sincronizadas a cada um segundo. As imagens são capturadas em formato *.jpeg* e transmitidas utilizando o API Python Bottle para o computador de destino como ilustrado na Figura 7.4. Os arquivos de calibração são enviados ao destino uma única vez, onde serão utilizados para gerar a nuvem de pontos.

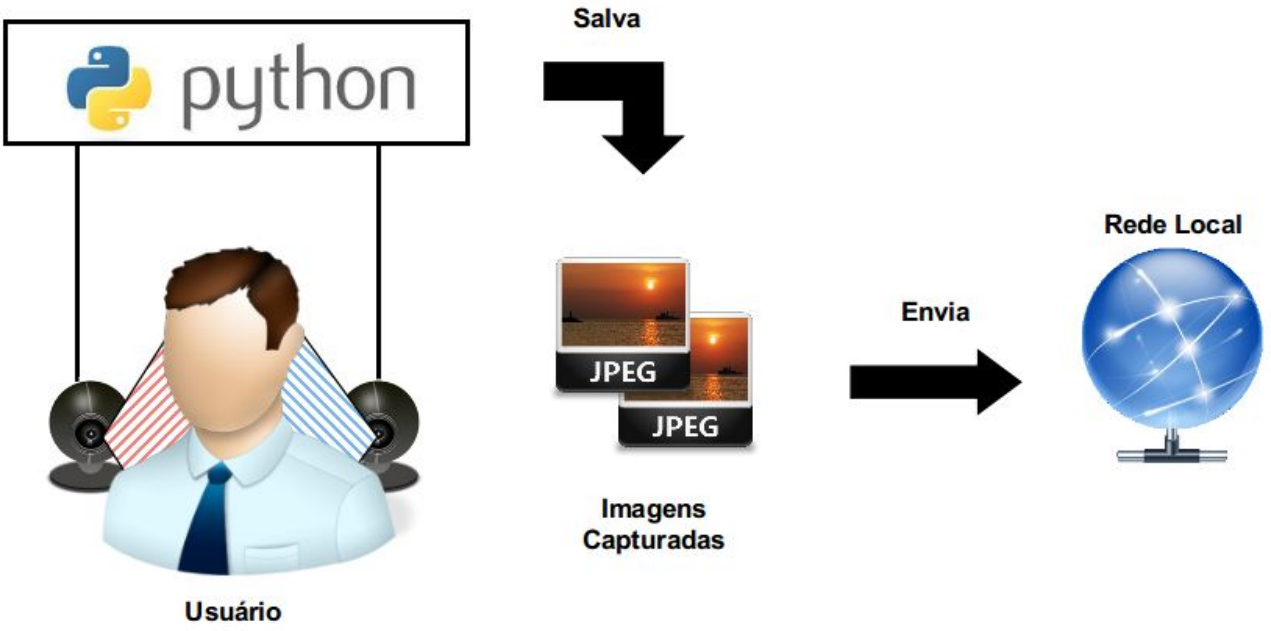

Figura 7.4: Ilustração disposição das cameras. Cada cor representa o campo de visão de cada dispositivo (esquerda, direita) em relação ao usuário que é fotografado. Cada webcam salva uma imagem em formato jpg que é enviada ao computador do usuário de destino utilizando a rede interna local. Fonte: o autor.

7.3 Nuvem de Pontos

Uma nuvem de pontos é uma coleção de pontos em um sistema de coordenadas. Em um sistema de coordenadas tri-dimensional esses pontos são definidos por X, Y e Z e usados para representar a superfície externa de um objeto.

Em sistemas com câmeras eestéreo para gerar a nuvem de pontos é necessário identificar os pontos correspondentes na imagem da esquerda e da direita. A geometria das câmeras possibilita restringir a busca por esses pontos em uma única linha dimensional na imagem da direita também conhecida como linha epipolar. Assim, a busca de um determinado ponto existente na imagem da esquerda pode ser encontrado na imagem da direita procurando na linha epipolar correspondente, não sendo preciso percorrer toda a linha o que acelera o processo de obtenção dos pontos correspondentes e a criação do mapa de disparidade.

Para encontrar os pontos correspondentes em cada imagem foi utilizado um algoritmo que usa apenas a informação local ao redor de cada pixel. As regiões de cada imagem conseguem ser mapeadas quando a região é distinta de outras regiões das imagens, por exemplo, regiões com texturas repetitivas ou ruins em geral não são obtidas no processo de fazer a correspondência entre pontos (ENSENSO, 2012). Nos experimentos realizados neste projeto, lugares com sombras ou muita luz (textura repetitiva) informações foram perdidas e não puderam ser inseridas na nuvem de pontos.

Encontrados os pontos correspondentes, é gerado um arquivo de extensão *.ply* que contém um cabeçalho de configuração que indica as variáveis presentes no arquivo e cada linha possui valores correspondentes aos eixos XYZ e cores no padrão RGB de cada ponto encontrado. Na Figura 7.5 o cabeçalho indica o formato da

nuvem de pontos, o número de vértices e quais propriedades estão incluídas. Ao fim do cabeçalho, cada coluna corresponde ao eixo X, eixo Y, eixo Z, e o Red, Green e Blue, respectivamente.

```
ply
format ascii 1.0
element vertex 187199
property float x
property float y
property float z
property uchar red
property uchar green
property uchar blue
end_header
-11.588039 12.757475 -27.215946 184 188 191
-11.496689 12.715232 -27.125828 186 190 193
-11.368421 12.631579 -26.947369 186 193 197
-11.315789 12.631579 -26.947369 185 192 196
-11.226230 12.590164 -26.859016 184 191 195
-11.210526 12.631579 -26.947369 183 190 194
-12.875000 15.000000 -32.000000 188 197 198
-12.812500 15.000000 -32.000000 189 198 199
```

Figura 7.5: Exemplo de arquivo de nuvem de pontos.

7.4 SISTEMA COMPLETO

O sistema deve funcionar de forma que uma videoconferência 3D em tempo real seja possível. Assim, cabeça e rosto dos usuários devem ser transmitidos e modelados garantindo desempenho mínimo satisfatório, ou seja, o modelo tri-dimensional deve ser fiel a face do usuário e a visualização em tempo real com atraso que não prejudique a conversa entre os participantes. O processamento de dados e a renderização consomem recursos consideráveis do computador e dependem principalmente da nuvem de pontos. Quanto maior a nuvem, mais tempo é necessário para construir o objeto tri-dimensional. O sistema de videoconferência 3D foi implementado conforme mostrado na Figura 7.6. e descrito na seção 4.1.

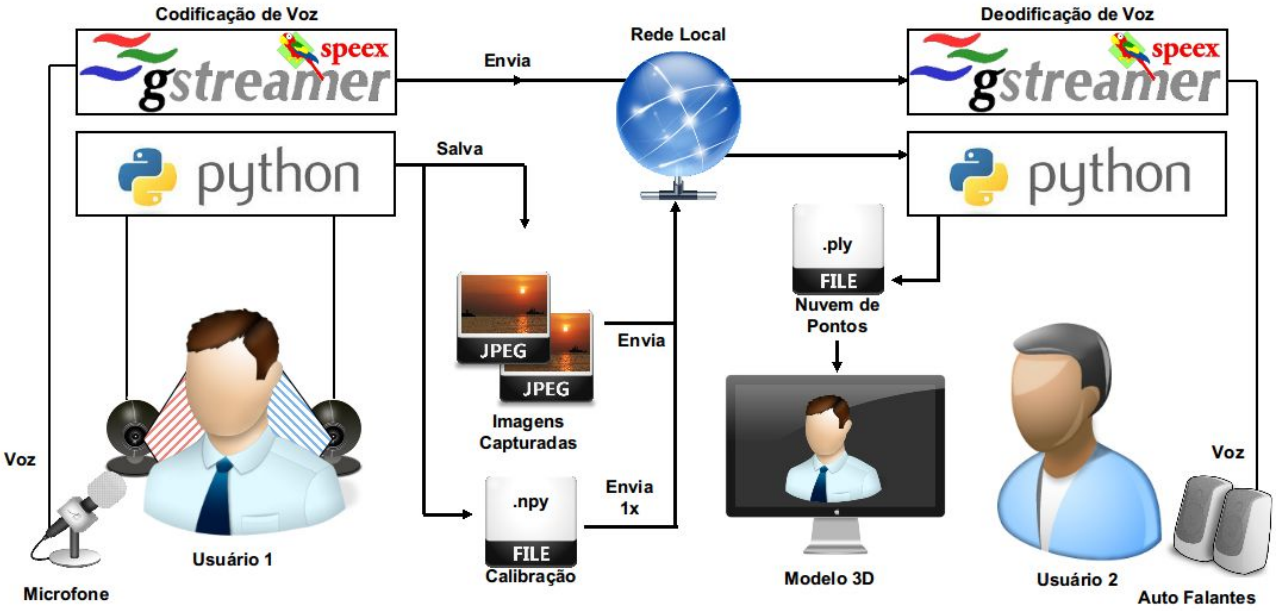

Figura 7.6: Sistema de videoconferência com chamada do usuário 1. Para a chamada do usuário 2, o mesmo processo ocorre, mas do usuário 2 para o usuário 1. Fonte: o autor.

7.4.1 Voz

1. Conforme a apresentado na Figura 7.6 , a voz do usuário 1 é capturada por um microfone, codificada no formato Speex utilizando GStreamer e transmitida para o computador do usuário 2 pela rede local utilizando o protocolo UDP.

2. A voz transmitida ao usuário 2 é decodificada pelo GStreamer e enviada aos auto-falantes, permitindo que este usuário possa ouvir o usuário 1.

7.4.2 Imagem

1. O usuário 1 fica de frente para as duas webcams. Utilizando o programa implementado em Python, as fotos são tiradas continuamente a um intervalo de um segundo, convertidas em formato *.jpeg* e enviadas pela rede local utilizando o protocolo UDP para o computador do usuário 2. Os arquivos de calibração são enviados uma única vez sempre que uma video conferência é inicializada.

2. O usuário 2 recebe as fotos que são processadas utilizando o programa implementado, gerando um mapa de disparidade. Esse mapa é usado com os arquivos de calibração e um arquivo de nuvem de pontos em formato *.ply* é gerado.

3. O arquivo é aberto por um programa e atualizado a cada um segundo, de forma que o usuário 2 possa ver e interagir com o modelo tri-dimensional.

8. PROTÓTIPO FINAL

Nesta seção é descrito os aspectos gerais do projeto divididos nas seguintes subseções: (i) descrição geral do produto, (ii) descrição funcional, (iii) manual operacional e, (iv) informações de suporte técnico.

Para simplificar o funcionamento do protótipo final, as descrições a seguir exemplificam o processo unidirecional do sistema implementado, isto é, modelo tri-dimensional do usuário 1 sendo visualizado pelo usuário 2. O sentido inverso também ocorre da mesma maneira.

8.1 Descrição Geral do Protótipo

O protótipo desenvolvido é constituído de duas webcams convencionais da marca dispostas em um apoio de alumínio preso a um tri-pé, um microfone e dois computadores com sistema operacional Ubuntu 14.04 LTS. Bibliotecas de programação devem ser instalados (ver anexos) para o funcionamento do sistema de videoconferência 3D implementado.

Um dos computadores terá as webcams ligadas a ele e atuará como servidor, o usuário 1 da Figura 7.6, responsável pelo envio de voz, calibração, captura e envio de imagens, assim como o envio único do arquivo de calibração das câmeras. O outro computador atuará como cliente, o usuário 2 da Figura 7.6. Este recebe as imagens e o arquivo de calibração, faz o processamento do mapa de disparidade recebido e gera uma nuvem de pontos e mostra o modelo tri-dimensional ao usuário. Também recebe os dados de voz transmitidos, decodifica e emite ao usuário.

Com base na Figura 7.6 da seção anterior, os programas automatizam o processo de envio de dados e voz entre computadores e realizam tarefas como construção do modelo tri-dimensional. O esquema pode ser visto conforme a Figura 8.1.

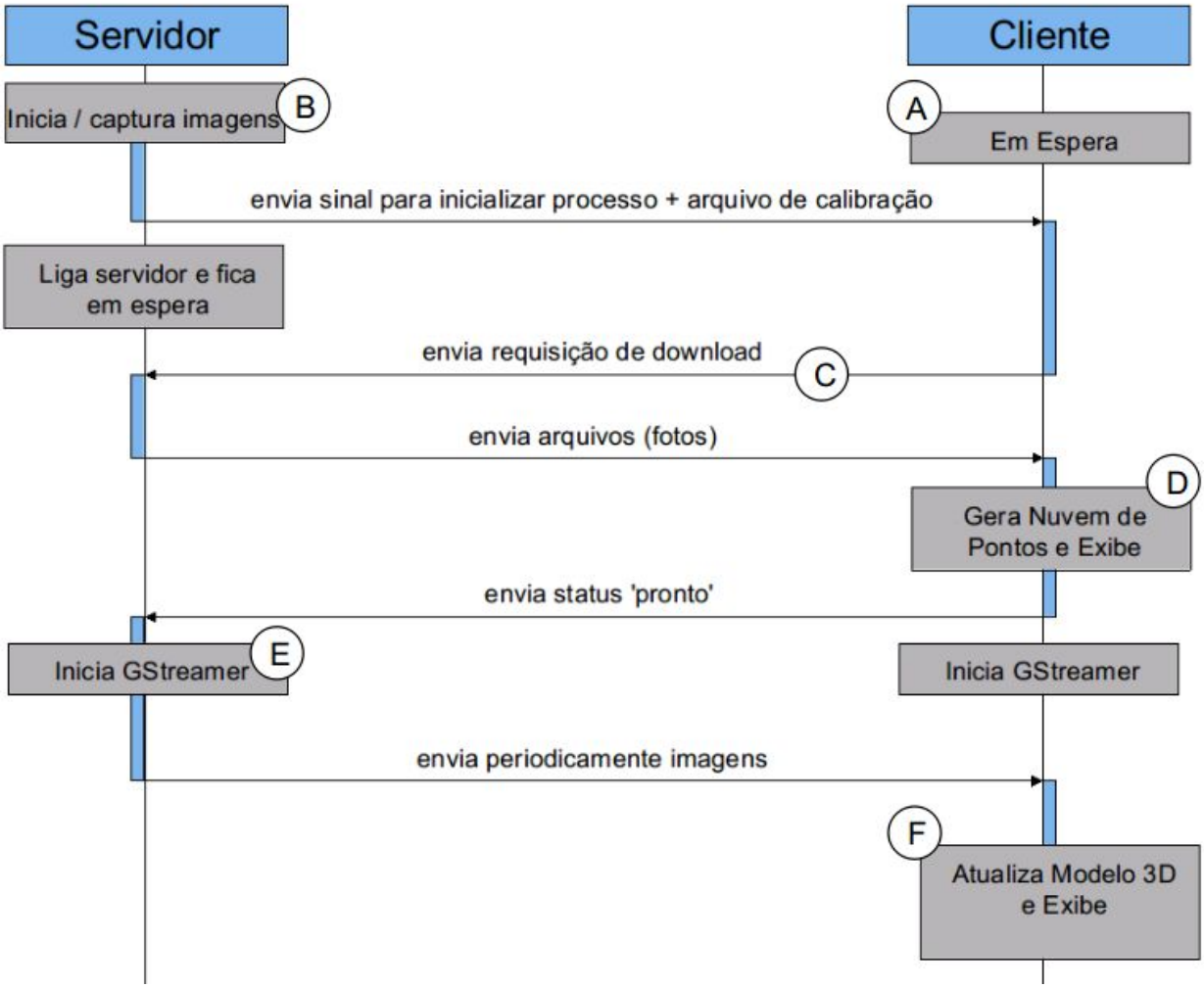

Figura 8.1: Esquema de comunicação entre cliente e servidor e envio de informações entre eles. Fonte: o autor.

A comunicação é realizada da seguinte forma:

A - O script do cliente deve ser o primeiro a ser inicializado. O script cliente fica em espera até que o script do servidor seja inicializado.

B - O servidor, se necessário, realiza a calibração das câmeras e salva os resultados em uma pasta. Este envia um sinal ao cliente e inicia a captura de imagens das duas webcams.

C - O cliente, ao receber o sinal, faz o download do arquivo comprimido uma única vez durante todo o processo e, a cada um segundo, faz o download das duas imagens.

D - Estas são processadas com os arquivos de calibração e geram um mapa de disparidade. A partir deste mapa, é gerado um arquivo de nuvem de pontos em formato *.ply* .

E - Em segundo plano, inicia-se a comunicação de voz entre os participantes da teleconferência utilizando o codificador Speex e a API GStreamer.

F - Este é aberto e processado por um visualizador, tornando possível que o usuário do lado cliente (usuário 2) possa ver e interagir com o modelo tri-dimensional gerado.

O processo de envio de imagens pelo lado servidor e processamento delas pelo lado cliente é contínuo e

realizado repetidas vezes até que a video conferência seja finalizada.

8.1.1 Webcams

As cameras foram dispostas frontalmente ao usuário conforme Figura 8.2. A distância entre câmeras foi variada de 8 a 20 cm com o objetivo de avaliar a qualidade dos modelos tri-dimensionais gerados e definir a distância adequada entre as webcams. As câmeras foram reguladas quanto a altura, ângulo e foco. Uma foto de cada câmera é capturada, codificada em formato de imagem *.jpeg* e em seguida salva em um diretório previamente especificado para posterior uso. Cada câmera tem sua foto salva.

Figura 8.2: Webcams utilizadas para captura de imagens em estéreo. Fonte: o autor.

8.1.2 Envio de arquivos pela Rede Local

Após as fotos e os arquivos de calibração serem salvos em um diretório específico, um aplicativo desenvolvido em linguagem de programação Python envia esses arquivos para o computador do lado cliente quando este faz download. É utilizada a rede local para isso. O script funciona conforme os passos a seguir:

- Cliente é executado na máquina do usuário 2. Servidor é executado na máquina do usuário 1;
- Cliente aguarda resposta do usuário 1. Uma vez recebida, o lado servidor cria um servidor temporário local da máquina. O lado cliente, acessa esse conteúdo e cria uma cópia das fotos e dos arquivos de calibração. O arquivo de calibração é obtido uma única vez.
- Ao finalizar a video conferência, o lado cliente envia uma mensagem continuamente para que o lado servidor desligue o servidor e finalize a execução. O lado cliente faz o mesmo.

8.1.3 Envio de voz pela Rede Local

Quando o processo de envio de imagens está completo, o lado servidor inicia a captura e codificação de voz utilizando o codificador Speex. A voz é codificada e enviada utilizando o API GStreamer que se

encarrega de processar os dados e utilizar o protocolo UDP para envio. Quando o servidor de voz está pronto, este envia um sinal ao lado cliente. Utilizando o GStreamer este recebe e decodifica os dados e emite a voz ao usuário utilizando auto falantes.

9. RESULTADOS DOS ENSAIOS EXECUTADOS

As subseções a seguir descrevem (i) os procedimentos experimentais; (ii) resultados obtidos; (iii) análise de desempenho e (iv) conclusão.

9.1 Procedimentos Experimentais

Dois computadores com sistema operacional Ubuntu 14.04 LTS 64-bits foram utilizados. Em cada máquina foram instalados os aplicativos e dependências do sistema conforme (ver anexos). Diferentes experimentos foram realizados. O primeiro variando a distância entre as câmeras. O segundo melhorando aspectos do cenário e iluminação. Ambos foram realizados com o objetivo de avaliar a qualidade da nuvem de pontos gerada segundo a iluminação do local e a influencia do espaçamento entre as câmeras na geração do mapa de disparidade.

9.1.1 Distância entre Câmeras

As imagens capturadas pelas câmeras são em estéreo, câmera da direita e câmera da esquerda. Como mencionado na seção 7.3 as câmeras possuem informações em comum e outras não. A distância entre elas permite obter mais informações distintas e que podem ser usadas para modelar diferentes ângulos do objeto tri-dimensional; entretanto, se as informações em comum não forem suficientes ou insatisfatórias, a obtenção do mapa de disparidade pode ser ruim e obtenção de valores relativos aos eixos XYZ serão imprecisas ou erradas. Para obter uma distância cuja qualidade do mapa de disparidade fosse adequada, alguns experimentos foram realizados variando a distância entre as duas câmeras em 8, 12, 16 e 20 centímetros.

9.1.2 Iluminação e Cenário

Com o valor de distância entre câmeras obtido, foi verificado os diferentes efeitos da iluminação do ambiente sobre as fotos capturadas e a nuvem de pontos gerada. A luz, principalmente, cosntitui um fator de importância na obtenção dos pontos, influenciando diretamente no brilho da imagem, assim como possíveis sombras e reflexos na face do usuário. Diferentes iluminações podem causar diferentes resultados. Foram feitos três experimentos variando a iluminação do local, primeiro com iluminação ambiente, com muita luz e com luzes frontais e laterais.

Nos testes foram improvisados ambientes de forma a favorecer ou não a iluminação do local. Alguns desses testes foram realizados em um quarto (Figura 9.1) e outro no laboratório NAVI3 da UFABC.

Figura 9.1: Improviso de ambiente. Melhora na iluminação utilizando luzes, reflexão da luz com paredes e entornos em cor branca etc. Fonte: o autor. (A) usuário de frente a câmera e pequena iluminação lateral; (B) uso de pano branco para refletir a luz do ambiente; (C) imagens capturadas por cada câmera.

9.1.3 Video Conferência 3D Experimental

Realizados os testes anteriores, foi realizada uma video conferência 3D experimental com câmeras a uma distância de 12 cm entre si (ver subseções abaixo), e luz adequada (frontal e lateral). Como há apenas um par de webcams, apenas um dos computadores enviou imagens em estéreo. O outro enviou um streaming de vídeo com uma única webcam utilizando o codificador de vídeo Theora e GStreamer.

9.2 Resultados Obtidos

Testes foram realizados e os resultados obtidos podem ser verificados nas subseções a seguir.

9.2.1 Distância entre Câmeras

Diferentes distâncias entre as webcams foram verificadas. As nuvens de pontos foram visualizadas utilizando o programa Meshlab. Alguns dos resultados são conferidos a seguir.

9.2.1.1 Distância de 20 cm

Foram feitos alguns disparos e imagens foram capturadas como visto na Figura 9.2.

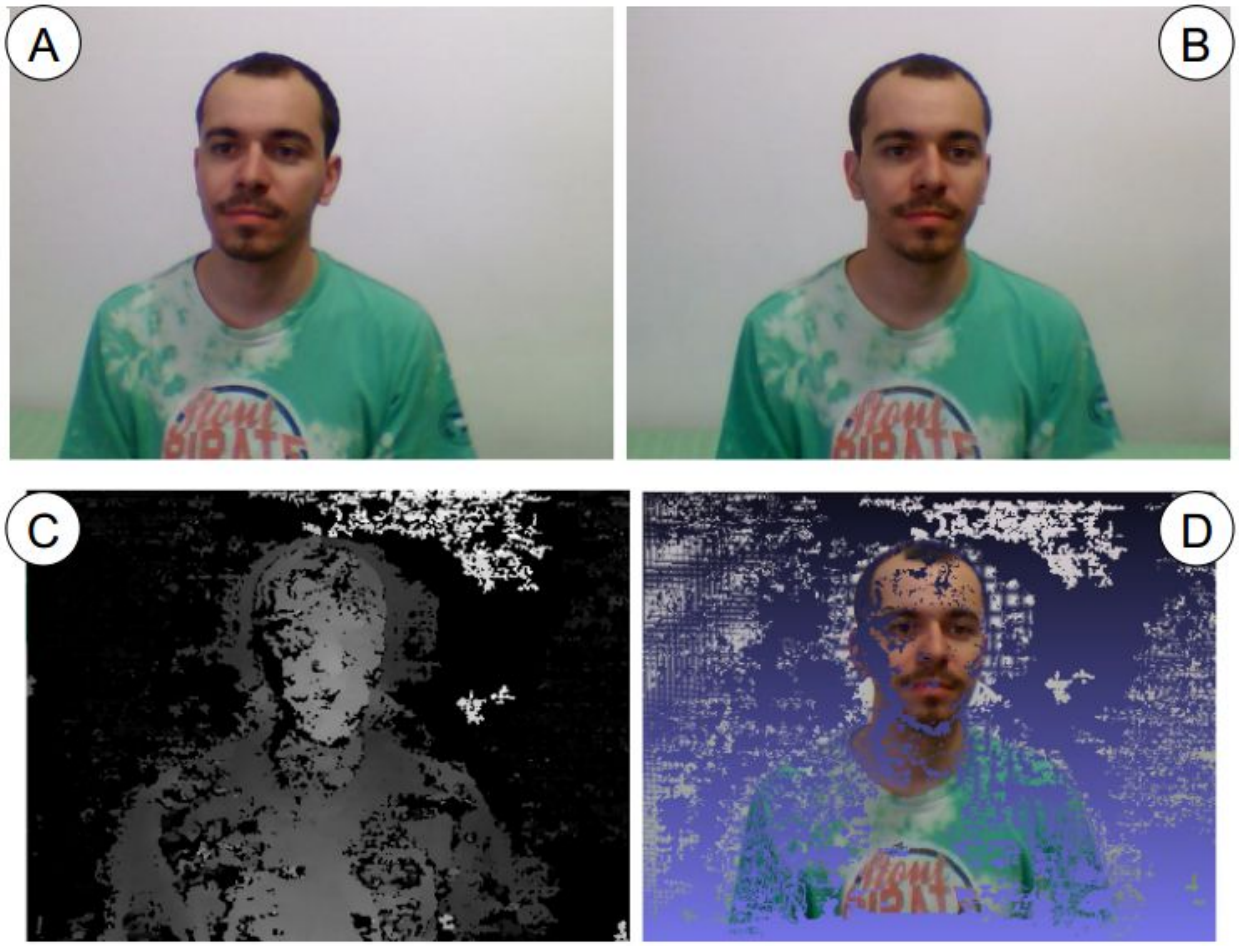

Figura 9.2: (A) Foto capturada pela câmera da esquerda e (B) foto capturada pela câmera da direita com 20 cm de espaçamento entre as cameras; (C) Mapa de disparidade gerado a partir das fotos e (D) visualização da nuvem de pontos. Fonte: o autor.

É possível observar que o mapa de disparidade apresenta diversos pontos escuros. Esses pontos são perda de informação, devido principalmente a não correlação dos pixels de interceção das duas imagens. Se a correlação não ocorre, nenhuma informação é salva. Considerando este aspecto, diversos pontos da face do usuário foram perdidos.

9.2.1.2 Distância de 16 cm

Foram feitos alguns disparos e imagens foram capturadas como visto na Figura 9.3.

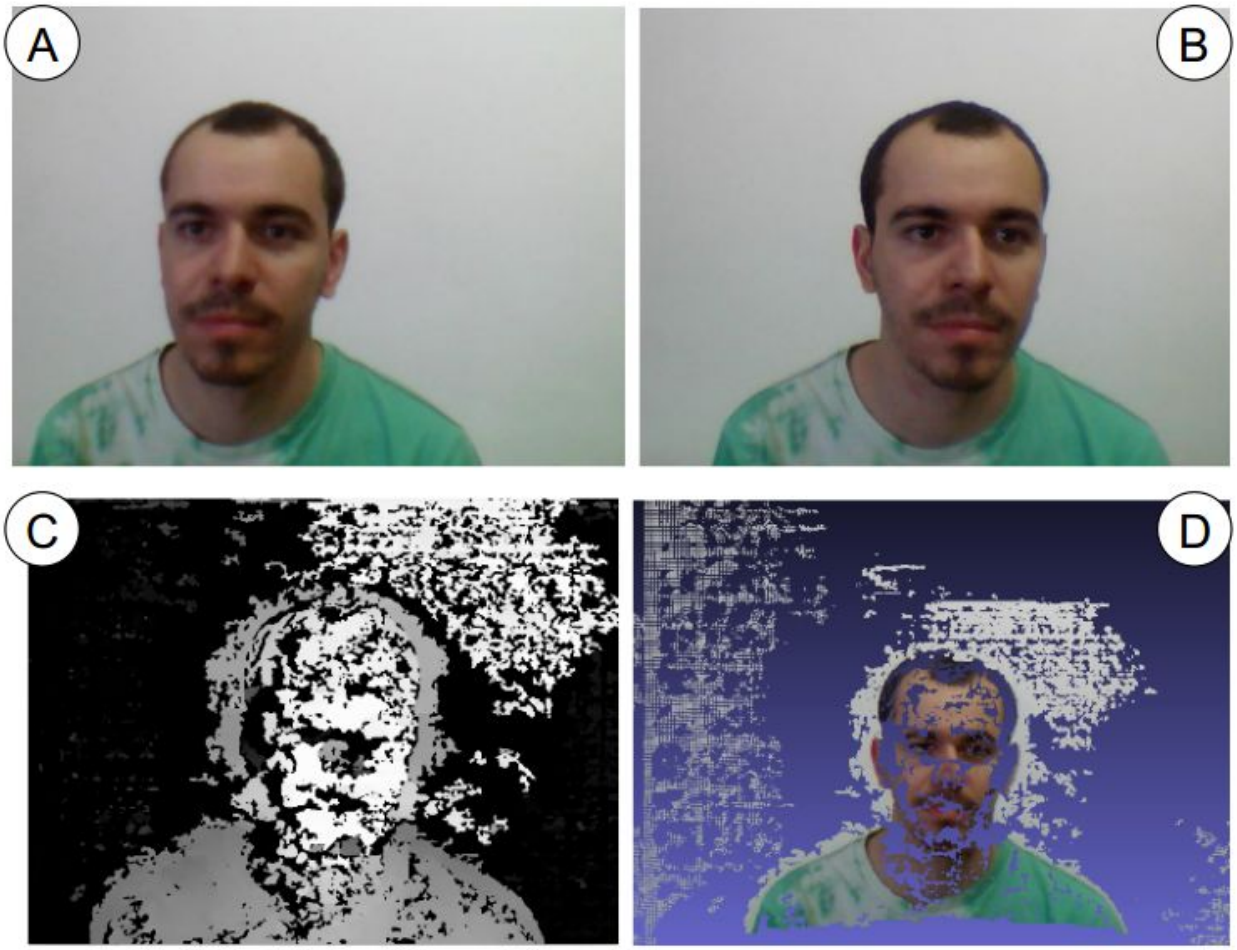

Figura 9.3: (A) Foto capturada pela câmera da esquerda e (B) foto capturada pela câmera da direita com 16 cm de espaçamento entre as cameras; (C) Mapa de disparidade gerado a partir das fotos e (D) visualização da nuvem de pontos. Fonte: o autor.

De todos os testes realizados, a nuvem de pontos gerada foi falha. Destaque a um dos testes em que uma das fotos estava sem foco. Devido não somente a distância entre câmeras, mas também ao foco ruim, a correlação entre os pixels de cada foto foi menor ainda e menos informações foram obtidas resultando em menos pontos na nuvem de pontos.

9.2.1.3 Distância de 12 cm

Foram feitos alguns disparos e imagens foram capturadas como visto na Figura 9.4.

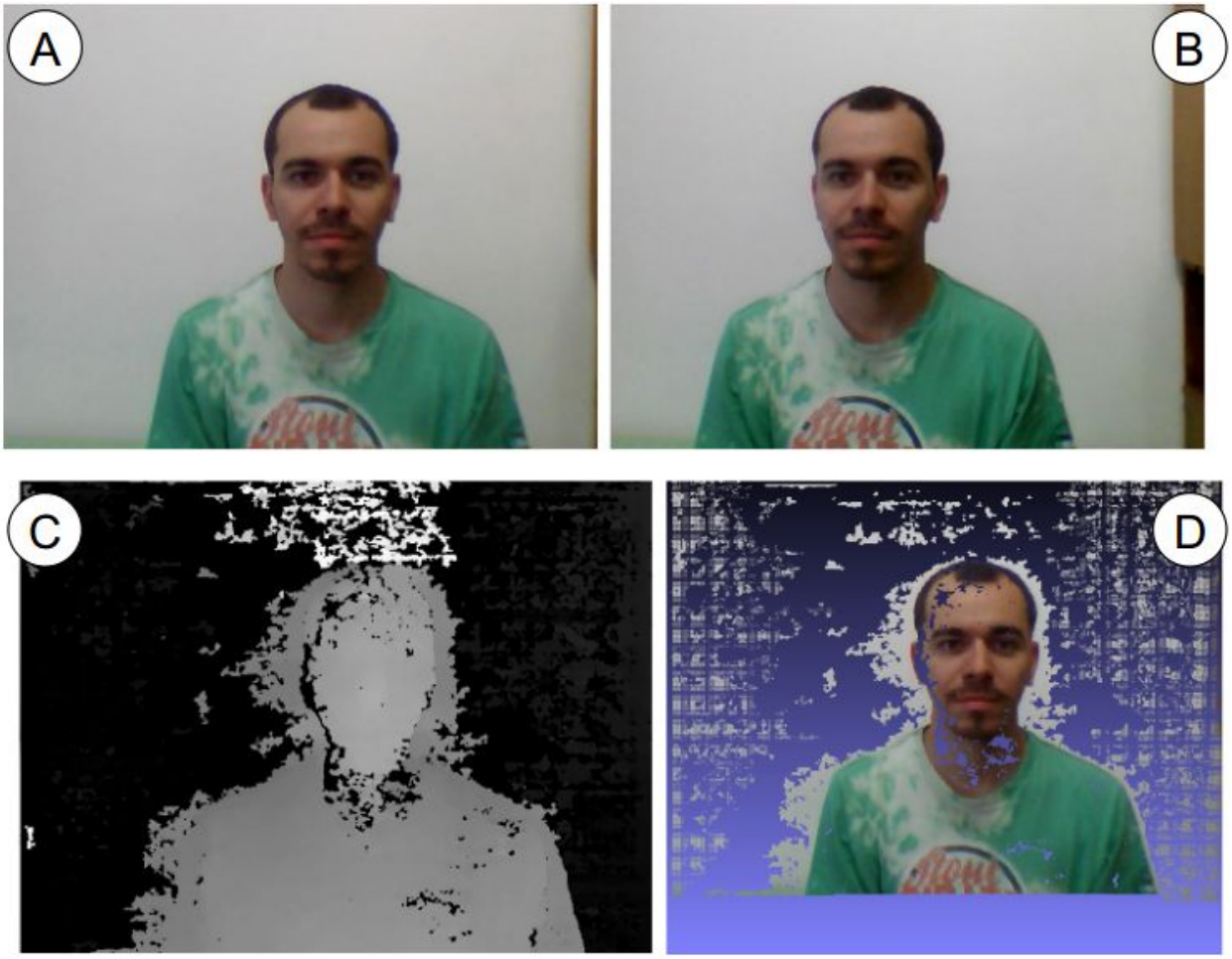

Figura 9.4: (A) Foto capturada pela câmera da esquerda e (B) foto capturada pela câmera da direita com 12 cm de espaçamento entre as cameras; (C) Mapa de disparidade gerado a partir das fotos e (D) visualização da nuvem de pontos. Fonte: o autor.

De todos os testes realizados referentes a distância entre câmeras, doze centímetros foi o que resultou nas melhores nuvens de pontos geradas. O mapa de disparidade é quase contínuo, o que significa que houve correlações entre pixels e suas vizinhanças. Como a perda de informação foi pequena, a nuvem de pontos gerada apresentou resultado satisfatório sendo que os ‘buracos’ ocorrerão principalmente em áreas de sombra (pescoço) ou brilho intenso (testa).

9.2.1.4 Distância de 8 cm

Foram feitos alguns disparos e imagens foram capturadas como visto na Figura 9.5.

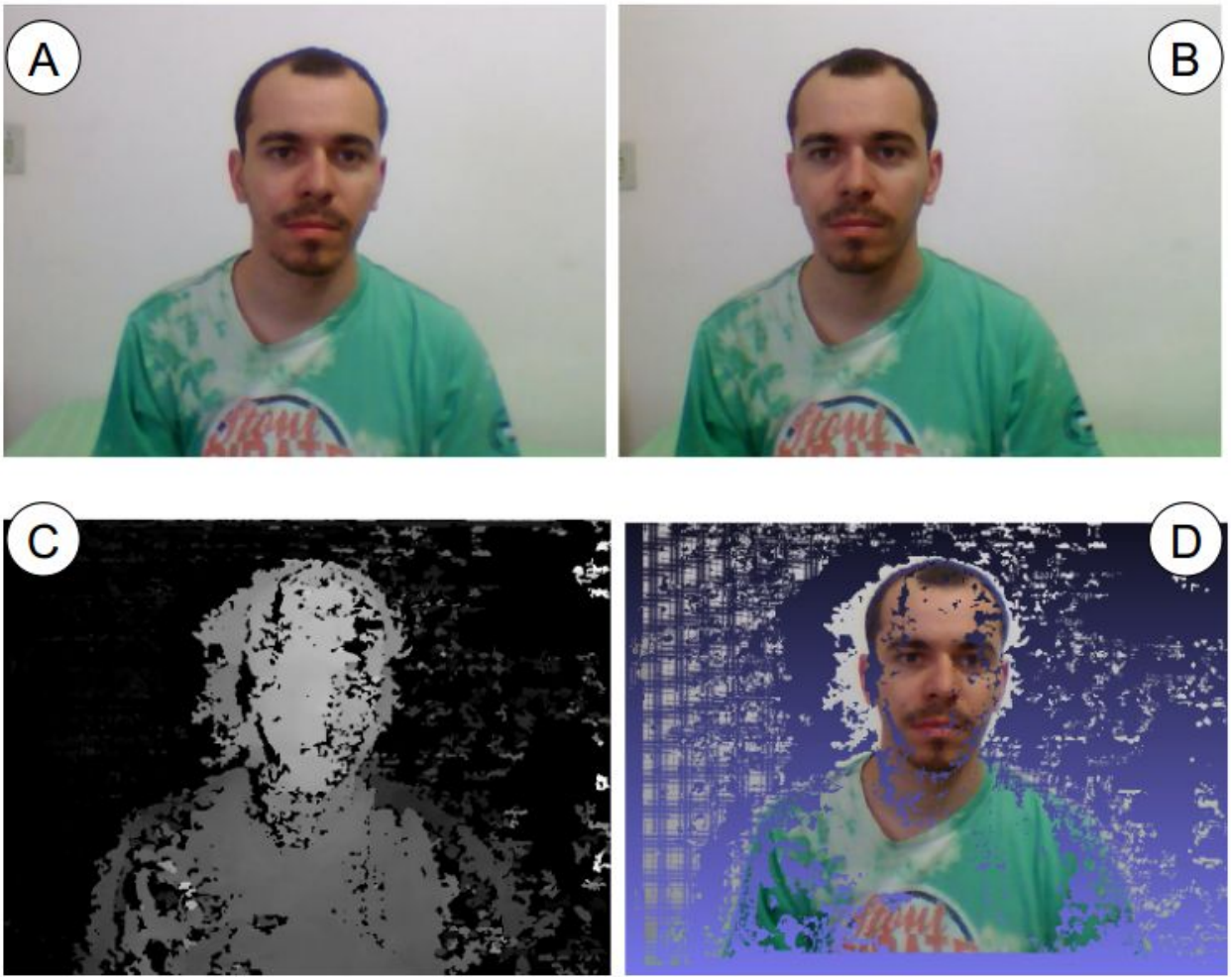

Figura 9.5: (A) Foto capturada pela câmera da esquerda e (B) foto capturada pela câmera da direita com 8 cm de espaçamento entre as cameras; (C) Mapa de disparidade gerado a partir das fotos e (D) visualização da nuvem de pontos. Fonte: o autor.

Com distância entre câmeras de apenas oito centímetros, a nuvem de pontos gerada foi similar a de doze centímetros, entretanto, a visualização tri-dimensional foi afetada devido a pequena diferença de ângulo das duas imagens capturadas. O efeito tri-dimensional é menos perceptível.

9.2.2 Iluminação

Alguns testes variando a iluminação do ambiente foram realizadas. Os resultados são apresentados a seguir.

9.2.2.1 Luz Ambiente

Foram feitos alguns disparos e imagens foram capturadas como visto na Figura 9.6. Foi utilizada apenas a luz do local, sendo que as duas câmeras foram colocadas exatamente abaixo da lâmpada.

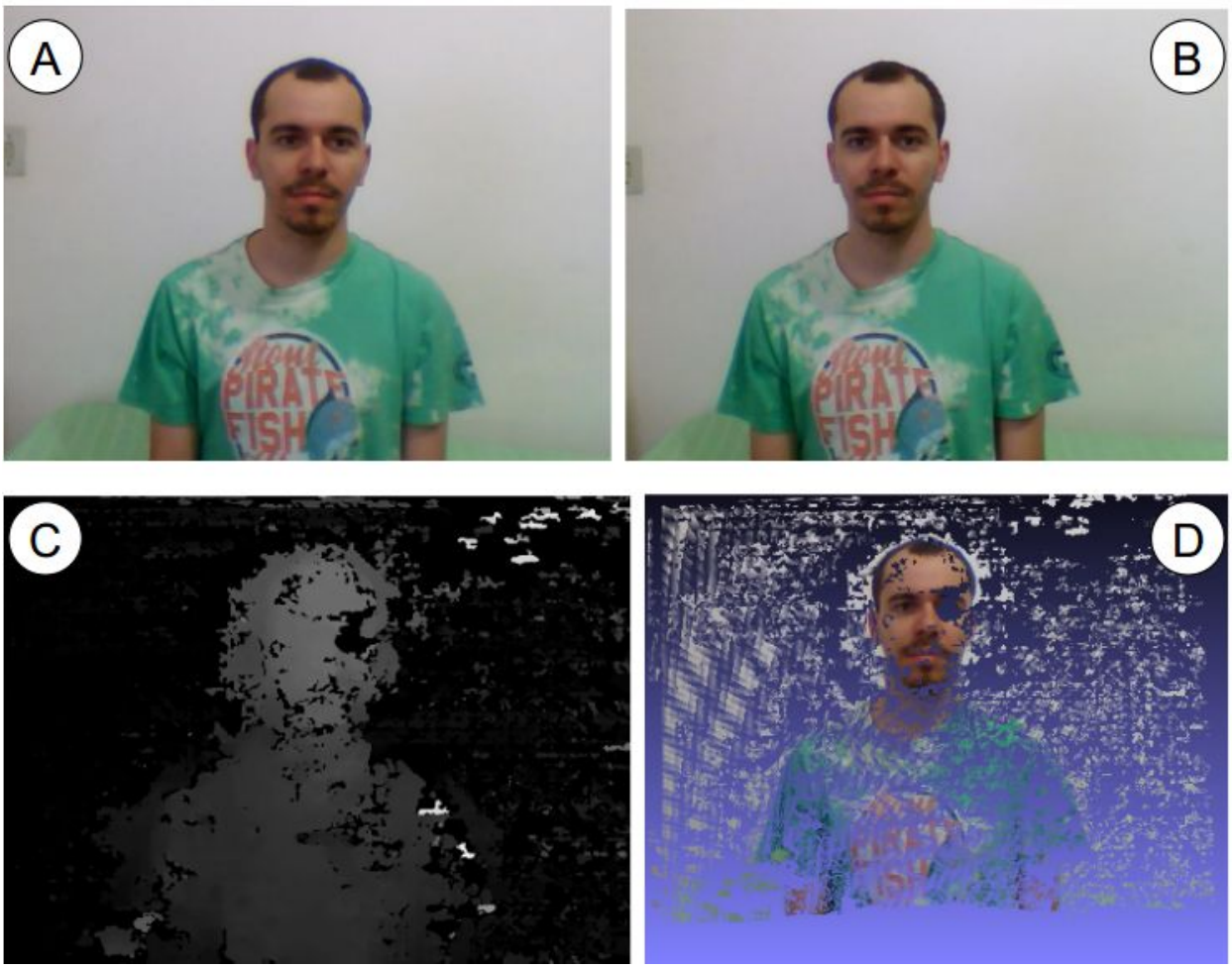

Figura 9.6: (A) Foto capturada pela câmera da esquerda e (B) foto capturada pela câmera da direita com 12 cm de espaçamento entre as cameras com luz ambiente; (C) Mapa de disparidade gerado a partir das fotos e (D) visualização da nuvem de pontos. Fonte: o autor.

É possível perceber que se houver sombra em alguma região da face, o mapa de disparidade é gerado sem informação da área. Na foto acima, a nuvem de pontos gerada não apresenta informação dos olhos.

9.2.2.2 Muita Luz

Foram feitos alguns disparos e imagens foram capturadas como visto na Figura 9.7. Foram ligadas as LEDs das webcams de forma que a face do usuário foi completamente iluminada sendo que as duas câmeras foram colocadas exatamente abaixo da lâmpada do local presa ao teto.

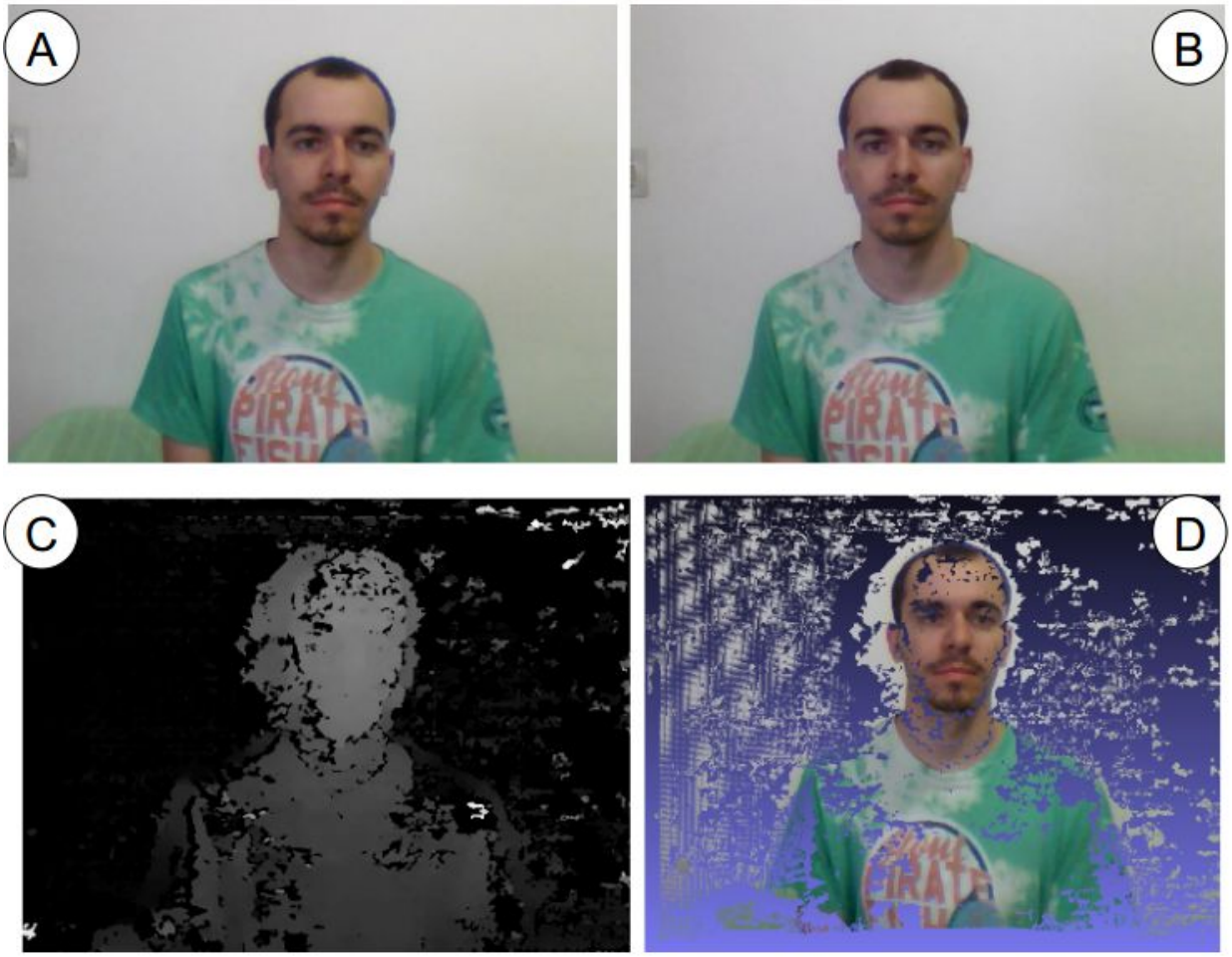

Figura 9.7: (A) Foto capturada pela câmera da esquerda e (B) foto capturada pela câmera da direita com 12 cm de espaçamento entre as cameras e com muita luz incidente; (C) Mapa de disparidade gerado a partir das fotos e (D) visualização da nuvem de pontos. Fonte: o autor.

A luz frontal ajudou a melhorar o mapa de dispersão da face, principalmente no centro, porém, as laterais foram escurecidas apresentando sombras. Outros detalhes como os da camiseta firacam mais evidentes, mas também apresentaram perda de informação. A manga direita da camiseta quase não existe na nuvem de pontos. No rosto, abaixo do queixo, a sombra fez por não construir totalmente o pescoço.

9.2.2.3 Luz Lateral e Frontal

Foram feitos alguns disparos e imagens foram capturadas como visto na Figura 9.8. Foi utilizada apenas a luz do local, sendo que as duas câmeras foram colocadas exatamente abaixo da lâmpada.

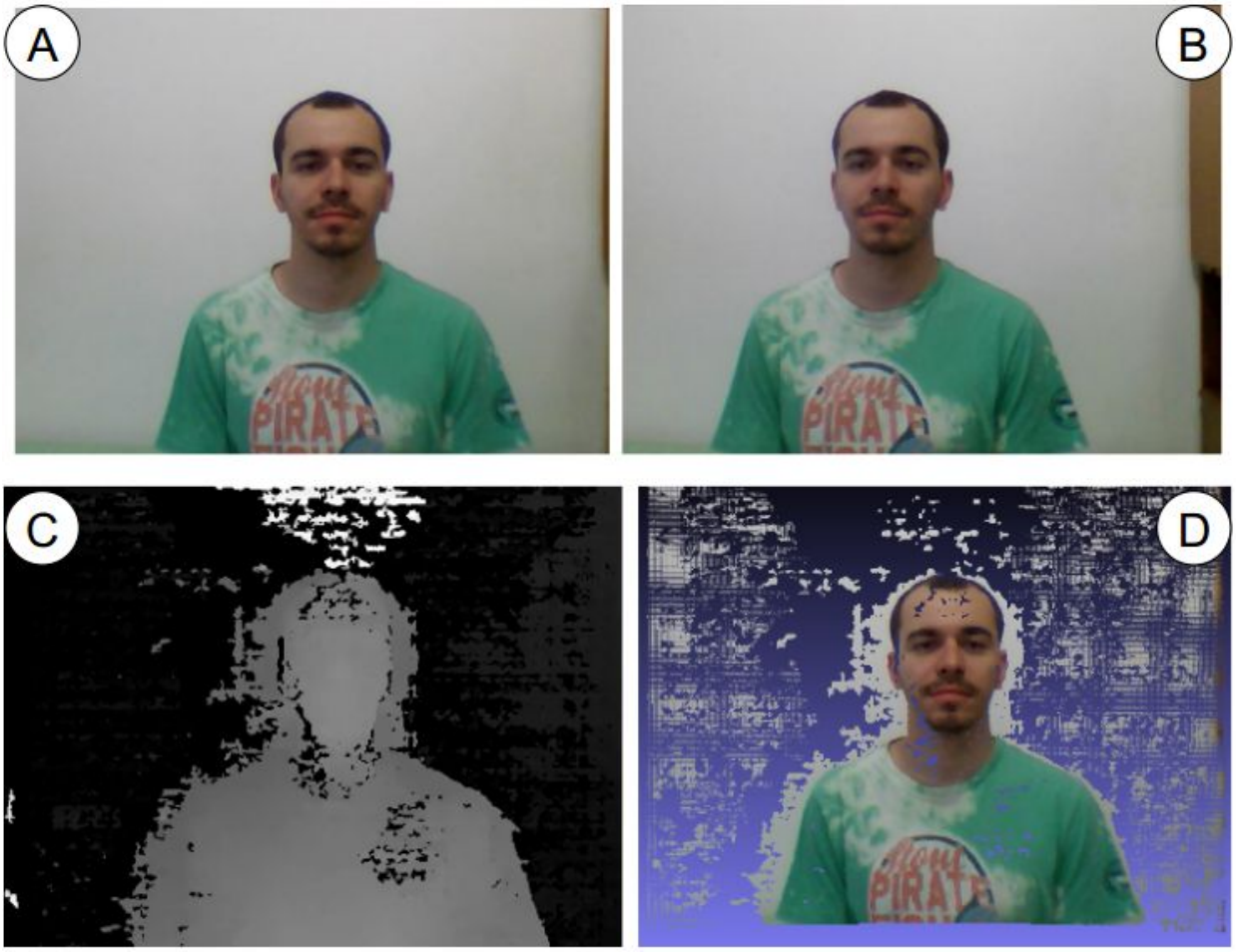

Figura 9.8: (A) Foto capturada pela câmera da esquerda e (B) foto capturada pela câmera da direita com 12 cm de espaçamento entre as cameras e luzes frontais e laterais; (C) Mapa de disparidade gerado a partir das fotos e (D) visualização da nuvem de pontos. Fonte: o autor.

Com luzes partindo de outras direções foi possível minimzar o erro e o mapa de disperção gerado foi talvez um dos melhores obtidos nos testes realizados. As sombras projetadas na face do usuário foram minimizadas e o rosto apresentou mais informações e pode ser visualizado mais nítido que os testes anteriores. A camiseta foi gerada quase em sua totalidade.

9.2.3 Outros Resultados

Foram realizados outros testes simples, apenas para verificação de comportamento do sistema. Dentre eles:

9.2.3.1 Cabelos longos

Em um caso em que um usuário de cabelos longos fosse projetado em um objeto tri-dimensional como visto na Figura 9.9.

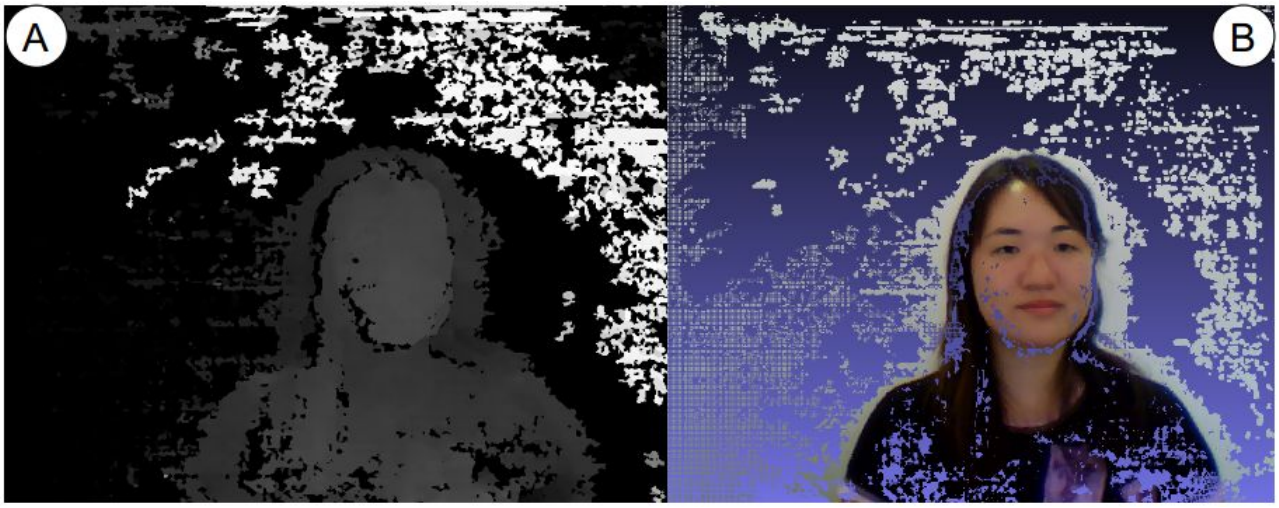

Figura 9.9: Fotos capturadas com espeçamento entre as câmeras de 12 cm de uma pessoa de cabelos longos; (A) Mapa de disparidade gerado a partir das fotos e (B) visualização da nuvem de pontos. Fonte: o autor.

O rosto foi obtido de forma considerável e apresentou boa definição. Os cabelos também foram gerados no modelo tri-dimensional e apresentaram sua forma, porém certos dados foram perdidos no processamento da nuvem de pontos. Importante observar que por se tratar de um cabelo escuro (objetos escuros) o mapa de disparidade pode não ser gerado corretamente, pois não consegue correlacionar os pontos por serem todos iguais.

9.2.3.2 Pele morena

Em um caso em que um usuário de pele morena fosse projetado em um objeto tri-dimensional como visto na Figura 9.10.

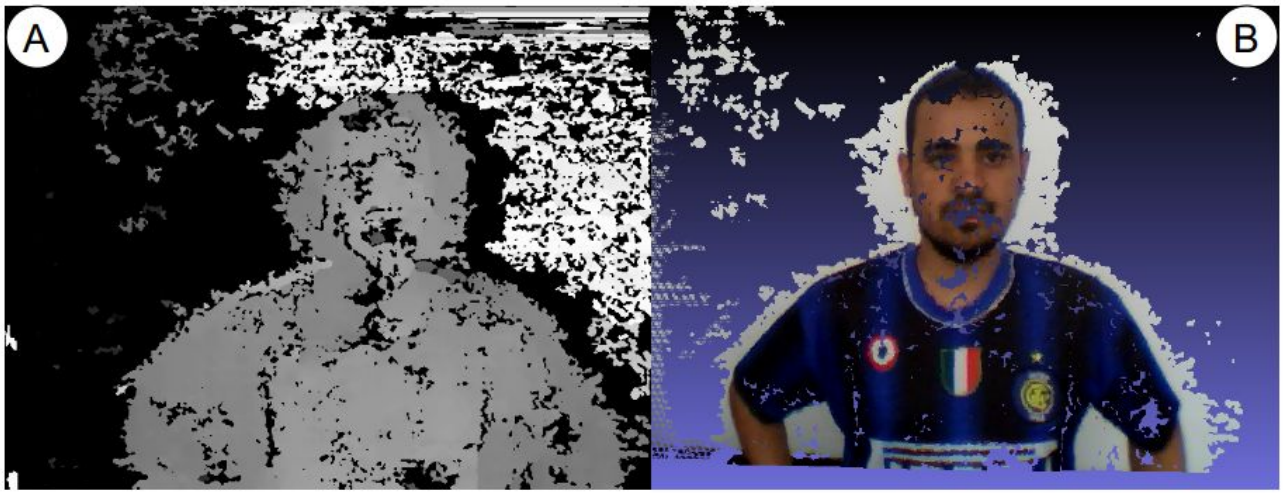

Figura 9.10: Fotos capturadas com espeçamento entre as câmeras de 12 cm de uma pessoa de pele morena; (A) Mapa de disparidade gerado a partir das fotos e (B) visualização da nuvem de pontos. Fonte: o autor.

Foi observado que a qualidade da nuvem de pontos não foi alterada perceptivelmente. O modelo gerado possui definição e alguns dos pontos não foram incluidos na nuvem de pontos principalmente devido a iluminação do ambiente, neste caso, no laboratŕoio da UFABC.

9.2.3.3 Objetos na cena

Em um caso em que um usuário e todo o ambiente fossem projetados em um objeto tri-dimensional como visto na Figura 9.11.

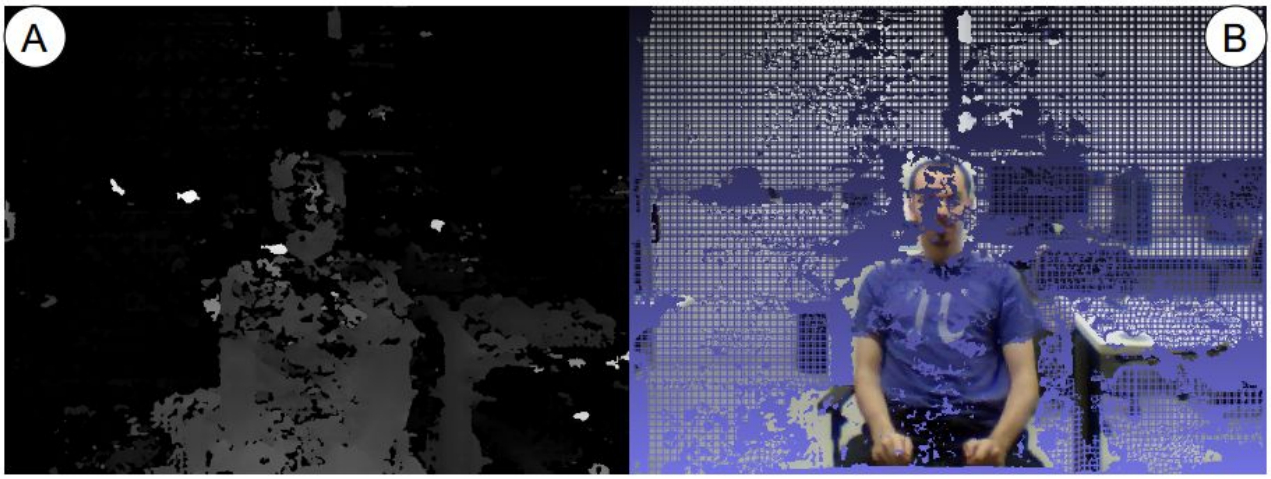

Figura 9.11: Fotos capturadas com espeçamento entre as câmeras de 12 cm de uma pessoa e os obejtos presentes no ambiente; (A) Mapa de disparidade gerado a partir das fotos e (B) visualização da nuvem de pontos. Fonte: o autor.

É possível projetar os objetos do ambiente, porém, novamente a iluminação tem papel importante na obtenção de uma nuvem de pontos com qualidade. E para webcams, o campo de visão é de cerca de 3 metros, ou seja, depois desta distância os objetos não tem pontos registrados nas coordenadas XYZ com aspecto tri-dimensional. São projetados como uma imagem chapada em uma parede.

9.2.4 Resultado Geral

A influência da distância entre câmeras afeta o efeito tri-dimensional do modelo gerado, assim como a qualidade da nuvem de pontos, obtendo ou não informações relativas a determinados pixels correlacionados entre si de cada foto capturada. Dos testes realizados e inseridos na Tabela 9.1 a distância entre câmera com melhores resultados foi a de doze centímetros.

Tabela 9.1: Visão geral dos resultados obtidos referentes a qualidade da nuvem de pontos obtidas em cada caso para distância entre câmeras de 12 centímetros.

Teste	Distância Entre Câmeras (cm)	Qualidade da Nuvem de Pontos
1	20	Muita informação perdida e muitos pontos descorrelacionados
2	16	Informação perdida e face pouco definida
3	12	Muitos pontos e face definida. Efeito 3D bom
4	8	Muitos pontos e face definida. Efeito 3D ruim.

Definida a distância entre câmeras igual a 12 centímetros, outro fator de importantância é a iluminação do local. A variação da luz pode causar efeitos indesejáveis como sombras, brilhos intensos etc que comprometem a qualidade da nuvem de pontos gerada, causando perda de informação. Dos testes realizados, o melhor esquema de iluminação foi a frontal e lateral. A qualidade subjetiva de cada experimento podem ser observados na Tabela 9.2.

Tabela 9.2: Visão geral dos resultados obtidos referentes a qualidade da nuvem de pontos obtidas em cada caso com 12 centímetros de distância entre as câmeras.

Teste	Iluminação	Qualidade da Nuvem de Pontos
1	Luz Ambiente	Muitos pontos, mas com falhas em regiões próximas aos olhos
2	Muita Luz	Muitos pontos, mas com falhas nas regiões de sombras e brilho intenso
3	Luz Lateral e Frontal	Muitos pontos e poucas falhas pois as sombras são minimizadas

9.3 Análise de Desempenho

Um dos objetivos do projeto é realizar uma video conferência em tempo real. Portanto, é importante que as nuvens de pontos obtidas por todo o processo possam ser visualizadas continuamente, ou seja, que o período tempo de atualização entre uma nuvem de pontos gerada e a outra não seja considerável.

Há diferentes fatores que limitam o tempo mínimo de visualização do modelo tri-dimensional: o tempo necessário para capturar as imagens, tempo de transmissão delas pela rede local, tempo de processamento das imagens e principalmente o tempo necessário para gerar o arquivo com a nuvem de pontos. Este último depende exclusivamente do hardware utilizado e processamento do computador, por este motivo, o projeto fica limitado aos computadores utilizados, todavia, foi observado que o visualizador de nuvem de pontos pode desempenhar um papel importante na qualidade da video conferência.

9.3.1 Video Conferência IP 3D Experimental

A video conferência experimental foi realizada de duas formas. A primeira utilizando um visualizador implementado em Python, a segunda com um visualizador em linguagem C++.

Na implamentação em linguagem Python do visualizado foi utilizada a biblioteca mayavi2 cujo objetivo é gerar e animar objetos tri-dimensionais, assim como nuvem de pontos. O único ponto negativo da biblioteca é que esta não tem função de importar cores específicas a cada ponto presente na nuvem de pontos, deixando os pontos com uma única coloração.

9.3.1.1 Video Conferência IP 3D - Python

O programa implementado permite ao usuário configurar o tempo de transição entre as nuvens de pontos que são obtidas e exibidas mudando o valor no campo *DELAY* , entretanto, devido a limitação no processamento dos arquivos *.ply* esse tempo pode ser maior dependendo do número de pontos existentes. O modelo visualizado pode ser rotacionado, movimentado ou mesmo ampliado. Os botões de animação permitem que o usuário que visualiza o modelo tri-dimensional possa pausar a atualização da nuvem de pontos a qualquer momento como pode ser visto na Figura 9.12.

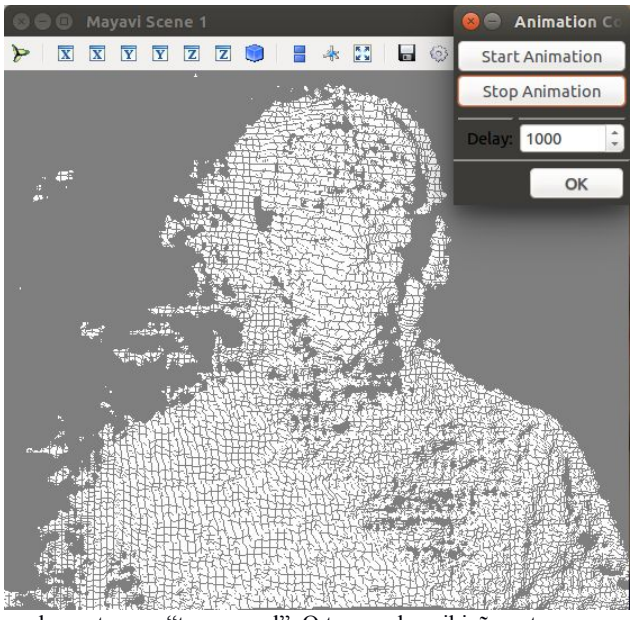

Figura 9.12: Visualizador da nuvem de pontos em "tempo real". O tempo de exibição entre nuvens de pontos foi de um segundo. Fonte: o autor.

A voz foi enviada e recebida com sucesso e os participantes puderam iniciar uma conversa. Um dos computadores, por ter apenas uma única webcam, transmitiu o vídeo ao computador com duas webcams em estéreo utilizando o codificador Theora.

O processo de obtenção e visualização da nuvem de pontos não foi satisfatório, devido principalmente a criação de um arquivo de extensão *.ply* e sua leitura. Também, a biblioteca Python para visualização da nuvem de pontos é limitada e o processamento dos pontos é muito lento, o que dificultou a implementação em tempo real da videoconferência 3D.

9.3.1.2 Video Conferência IP 3D - C++

O programa foi implementado em C++ e exibe a nuvem de pontos e as respectivas cores de cada ponto e permite interação do usuário com o modelo tri-dimensional utilizando os botões do mouse como pode ser visto na Figura 9.13.

Figura 9.13: Visualizador da nuvem de pontos em “tempo real”. O tempo de exibição entre nuvens de pontos foi entre um a quatro segundos. Fonte: o autor.

O programa foi modificado para que atualizasse periodicamente com uma nova nuvem de pontos. Foi observado que o programa consegue renderizar e exibir a nuvem de pontos mais rapidamente que o visualizador implementado em Python, devido principalmente, a linguagem C++ ser mais rápida no processamento de dados.

O processo de transmissão de dados e criação da nuvem de pontos foi modificado quando comparado com o programa anterior em Python. Neste processo, o computador com as duas webcams é responsável por capturar as imagens, gerar o mapa de disparidade e enviar ao outro computador. Este recebe apenas a foto capturada pela câmera da esquerda e o mapa de disparidade, gerando e atualizando a cada segundo a nuvem de pontos e exibindo ela ao usuário. Diferentes processos foram excluídos neste novo programa a fim de tornar o processamento de dados mais eficiente. Por exemplo, a nuvem de pontos não foi escrita em um arquivo de extensão *.ply*, não necessitando ler o arquivo posteriormente. Todos os valores foram gerados e armazenados em variáveis do próprio visualizador ao fazer a leitura da imagem da câmera esquerda e do mapa de disparidade.

Recebida as imagens a nuvem de pontos é gerada e visualizada pelo usuário como visto na Figura 9.14. Este pode interagir rotacionando, movimentando, ampliando ou diminuindo ela.

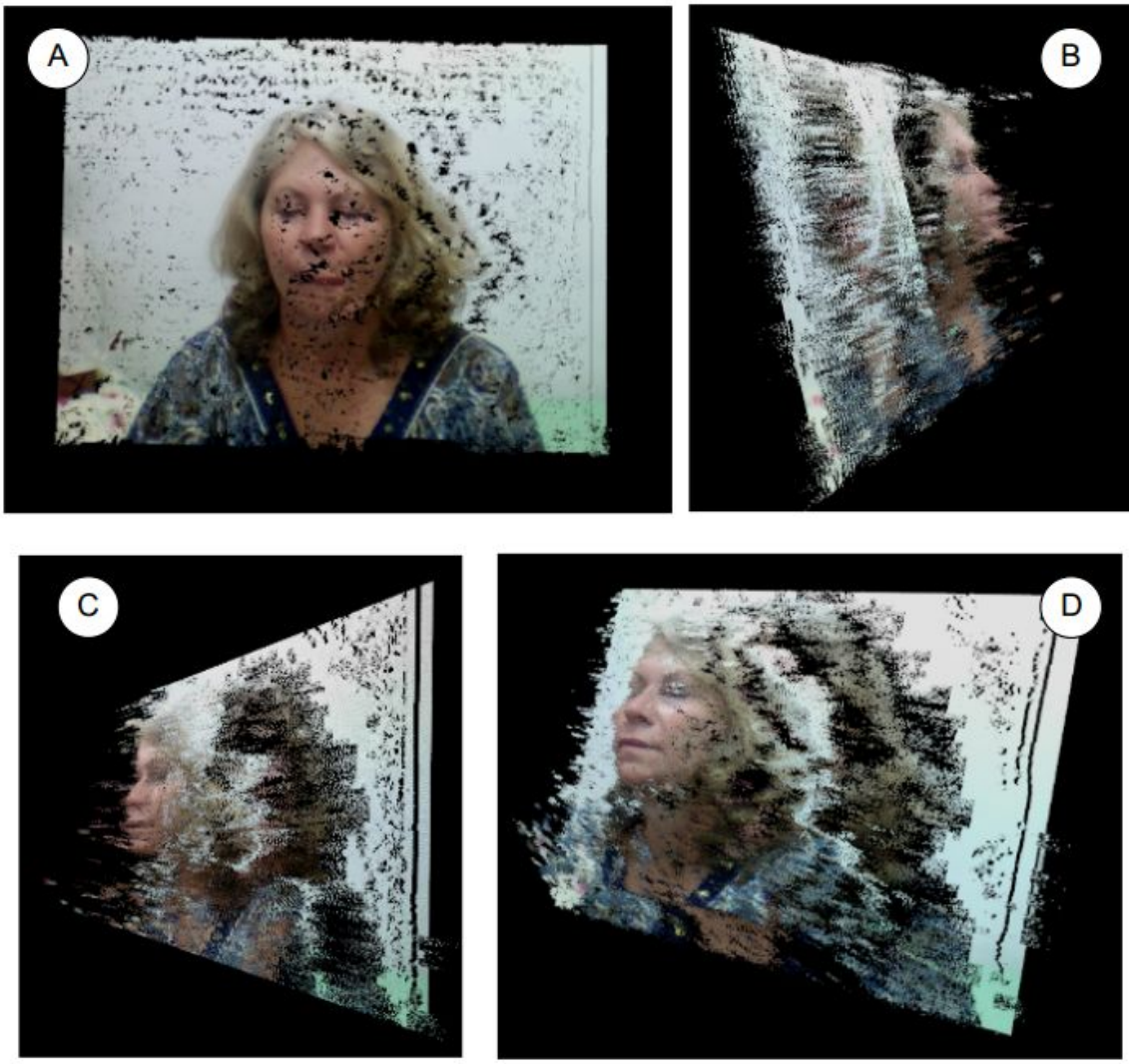

Figura 9.14: (A) Visualização da nuvem de pontos visão frontal do participante; (B) visão da lateral direita do participante; (C) visão da lateral esquerda do participante; (D) visão do pescoço e parte da lateral esquerda e lateral direita do participante. Fonte: o autor.

Os resultados obtidos foram satisfatórios, mas o programa ainda é instável. Devido principalmente a leitura do mapa de disparidade e da imagem capturada. Também foi possível observar demora entre a voz que sai do auto-falante e a movimentação da boca do participante que tem a nuvem de ponto gerada. Essa demora pode variar entre um a quatro segundos.

A vozes do usuários foram transmitidas e recebidas com sucesso e os participantes puderam iniciar uma conversa, embora com atraso. Um dos computadores, por ter apenas uma única webcam, transmitiu o video ao computador com duas webcams em estéreo utilizando o codificador Theora.

9.4 Conclusão

Implementar um sistema de video conferência 3D em rede local exige conhecimento, técnica e principalmente experiência. Alinhar as câmeras requer paciência e muitos testes. Por serem utilizadas duas câmeras, é difícil obter imagens sincronizadas com mesmo grau de iluminação, brilho e contraste.

Fatores diversos podem influenciar muitas vezes de forma negativa a obtenção das nuvens de pontos que são baseadas nas imagens capturadas pelas câmeras. Por exemplo, foi observado que se há pouca luz, o mapa de disparidade não contem informações suficientes, mas se há muita luz, pode causar perda de informações devido a sombras ou mesmo influenciar no mapa de disparidade indicando posicionamento errado dos pontos. Portanto, é imprescindivel que o ambiente tenha a iluminação adequada e controlada de forma que se possa minimizar efeitos de sombra na face do usuário.

Foi observado que o processamento e visualização das nuvens de pontos em tempo real requer computadores com bom processamento e que o desempenho do visualizador de nuvem de pontos pode estar ligado diretamente da linguagem que este foi implementado e a forma de implementação. Neste projeto, ficou evidente que o visualizador em linguagem C++ obteve resultados melhores que o visualizador em Python. Foi observado também que o desempenho do visualizador está relacionado com os processos de obtenção da nuvem de pontos e como são executados.

O sistema de video conferência 3D implementado obteve resultados satisfatórios e a comunicação entre os participantes ocorreu sem perda de qualidade de voz, entretanto, o processamento das imagens e principalmente a obtenção da nuvem de pontos foram fatores impactantes na execução da video conferência, pois foram os processos que demandaram maior processamento computacional e que influenciou no tempo de inicialização da video conferência.

Com a intenção de implementar uma video conferência IP 3D, o projeto obteve resultados satisfatórios utilizando poucos recursos e investindo em equipamentos de baixo custo, não havendo a necessidade de utilização de programas ou dispositivos específicos para execução do sistema.

O presente projeto utilizou de técnicas de calibração e obtenção de nuvem de pontos existentes em bibliotecas muito utilizadas como do OpenCV, porém um novo passo foi dado, gerando não apenas uma nuvem de pontos estática, mas atualizando ela de forma que o usuário final pudesse interagir com uma nuvem de pontos dinâmica do outro usuário gerada a cada período de segundos. Isto proporcionou maior imersividade e trouxe um atributo importante de uma video conferência que é a visualização da imagem de outro usuário em tempo real e com movimentos.

Um dos fatores positivos do sistema é a necessidade de utilizar pouca banda da rede local. São transmitidas duas imagens em *JPEG* por segundo cujo tamanho total é menor que 130 kB e a taxa de transmissão de voz menor que 20 kbps. Uma banda larga de 2 MB é mais que suficiente para a transmissão de dados.

O projeto porém, possui algumas limitações. Dentre elas o tempo de envio, recebimento e posterior geração da nuvem de pontos, o que ocasiona em atraso entre voz e a nuvem de pontos visualizada. O movimento da boca do usuário, por exemplo, é atrasada em relação a voz que sai do auto-falante. Também, por se tratar de um sistema de câmeras estéreo, a face do usuário obtida na nuvem de pontos possui poucas

informações das laterais da face. Isso pode ser melhorado se o sistema for implementado com mais câmeras.

O sistema ainda precisa de diversas melhorias para torná-lo um sistema estável, confiável e de fácil instalação. Atualmente algumas empresas tem investido em câmeras estéreo, o que contribuiria positivamente na facilidade de instalação do sistema em uma casa por exemplo, e utilizá-lo sem necessidade de processos mais complexos ou mesmo alinhamento de câmeras.

Futuramente, espera-se melhorar o sistema implementado tal que este possa gerar uma nuvem de pontos ou criar um modelo tri-dimensional do usuário com maior qualidade e de diferentes ângulos. Espera-se, dessa forma, obter maior imersão e interatividade quando comparado com os sistemas de video conferência presentes.

REFERÊNCIAS

APC. Let's get physical explaining how kinect for xbox works. Disponível em <http://apcmag.com/lets-get-physical-explaining-how-kinect-for-xbox-works.htm> . Acesso em: 22 dez. 2014.

AKIMOTO, T; SUENAGA, Y; WALLACE R.S. *Automatic Dreation of 3D Facial Models*. IEEE Computer graphics and Applications, p 16-22, 1993.

CICEROMORAES. Ideias Sutis Servido Grandes Realizações. Disponível em: <http://www.ciceromoraes.com.br/?p=85>. Acesso em: 27 ago. 2014.

Depth Biomechanics. Disponível em: <http://www.depthbiomechanics.co.uk/?cat=18>. Acesso em: 27 ago. 2014.

DRÖPPELMANN, S.; HUETING, M.; LATOUR, S.; VAN DER VEEN, M.; *Stero Vision using the OpenCV library.* University of Amsterdam, Junho 2010.

ENSENSO und IDS Imaging Development Systems; *Obtaining Depth Information from stereo Images.* Ensenso und IDS Imaging Development Systems GMbH, 2012.

FELDMANN, I; WAIZENEGGER, W.; ATZPADIN, N.; SCHREER, O. *Real-time Depth Estimation for Immersive 3D Videoconferencing*. 3DTV-Conference: the True Vision – Capture, Transmission and Display of 3D Video. Tampere, Finlândia: IEEE, Julho 2010, p 1-4. ISBN 978-1-4244-6377-0.

FILHO, J.C.S.; BALTAZAR, A.P.; PERONTI, R.; LAGES, W.; Princípios Teóricos da Estéreoscopia. AIVITS – Ambientes de Imersão Virtual de Tecnologia Simplificada. LAGEAR – Laboratório Gráfico para Ensino da Arquitetura. EVA – Estúdio Virtual de Arquitetura. Escola de Arquitetura da UFMG, 2008.

FLORENCIO, D.; ZHANG, C. Multiview video Compression and Streaming Based on Predicted Viewer Position. In Proceedings of ICASSP, Taipei, China: IEEE, Abril 2009 , p. 657-660.

FURUKAWA, Y.; PONCE,J.; *Accurate, Dense, and Robust Multi-View estéreopsis*. IEEE Transactions on Pattern Analysis and Machine Intelligence, August 2010. Vol. 32, Issue 8, p. 1362-1376.

FVVR - Free-View Video Render. Disponível em <http://sourceforge.net/projects/fvvr>. Acesso em: 22 maio de 2014.

INERN - Ireland’s National Education & Research Networkhttp. Disponível em <http://www.heanet.ie/videoconferencing/tutorial/history>. Acesso em: 10 maio 2014.

INSIGHT3D - Open Source Image Based 3D Modeling Software. Disponível em <http://insight3d.sourceforge.net/>. Acesso em: 22 maio 2014.

KURASHIMA, C.S.; YANG, R.; LASTRA, A. ; *Combining Approximate Geometry with View-Dependent Texture Mapping - A Hybrid Approach to 3D Video Teleconferencing.* In: SIBGRAPI Brazilian Symposium on Computer Graphics and Image Processing, 2002, Fortaleza, CE. Proceedings of XV Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2002). Porto Alegre, RS: Sociedade Brasileira da Computação, 2002. p. 112-119.

LORENSEN, W.E.; CLINE, H.E. *Marching Cubes: A High Resolution 3D Surface Construction Algorithm.* SIGGRAPH '87 Proceedings of the 14th annual conference on Computer graphics and interactive techniques. New York: ACM, Julho 1987. p. 163-169. ISBN:0-89791-227-6.

MESHLAB. open source, portable, and extensible system for the processing and editing of unstructured 3D triangular meshes. Disponível em <http://meshlab.sourceforge.net/>. Acesso: 21 jun. 2014.

NEFSIS. Video Conferencing History. Disponível em <http://www.nefsis.com/Best-Video-Conferencing-Software/video-conferencing-history.html> . Acesso em: 10 maio 2014.

NIEM, W.; WINGBERMUHLE, J. *Automatic Reconstruction of 3D Objects Using a Mobile Monoscopic Camera*. IEEE 3-D Digital Imaging and modeling. Ottawa: IEEE, Maio 1997 proceedings. p. 173-180. ISBN 0-8186-7943-3.

PIAZENTIN, J. H.O.; SEMENTILLE, A.C.; CALDEIRA, M.A.C.; MARAR, J.F. *Poisson Surface Reconstruction with Local Mesh Simplification*. Ouro Preto: Proc. SIBGRAPI 2012 - XXV Conference on Graphics, Patterns and Images, 2012. p. 39-40.

REALPLAYER. 3D Video Conference Hologram. Disponível em <http://www.real.com/resources/3d-video-conference-hologram>. Acesso em: 10 maio 2014.

RIBEIRO, D.; KURASHIMA; C.S.; GUELFI, A.E.; ZUFFO, M.K. *Evaluation of Theora and Speex Performance in IP Videoconferencing*. 18th Brazilian Symposium on Multimedia and the Web - Webmedia'12, 2012, Sao Paulo, SP. Proceedings of the 18th Brazilian Symposium on Multimedia and the Web. New York: ACM, 2012. p. 135-138.

YANG, R..; KURASHIMA, C.S.; TOWLES, H.; NASHEL, A.; ZUFFO, M.K. *Immersive Video Teleconferencing with User-Steerable Views.* Presence - Teleoperations and Virtual Envoronments. Massachusetts: MIT Press, Abril 2007. p. 188-205.

SPEEX. A Free Codec for Speech. Disponível em <http://www.speex.org/>. Acesso em: 21 jun. 2014.

WEIK, S.; WINGBERMUEHLE J.; KOPERNIK, A. H*ighly Realistic Modeling of Persons for 3D Videoconferencing Systems*. IEEE Workshop on Multimedia Signal Processing. Princeton: IEEE, Junho 1997. p. 286-291. ISBN: 0-7803-3780-8.

ANEXO: Descrição da Instalação e Execução do Software

1. Descrição Funcional

O script em Python do cliente foi executado primeiro no computador com apenas uma webcam e o script do servidor executado no computador com as webcams em estéreo.

Para executar os scripts, extraia os arquivos em uma pasta e siga os procedimentos na seção seguinte (Manual Operacional). Realizado cada procedimento, para executar os scripts é necessário dar permissão a cada um deles. Vá até a pasta onde os arquivos foram extraídos e execute como no Código 1.

```
chmod +x executa_cliente.sh
chmod +x executa_servidor.sh
```

Código 1: Comandos para conceder permissão a cada script.

Confirmadas as permissões, foi executado, nesta ordem, o computador com uma webcam o script cliente e no computador com as duas webcams em estéreo, o servidor (Código 2).

```
./executa_cliente.sh
./executa_servidor.sh
```

Código 2: Comandos para execução de cada script.

2. Manual Operacional

Esta subseção descreve os passos necessários para instalação de dependências e uso do sistema descrito anteriormente. Importante observar que o sistema foi implementado em ambiente Linux de distribuição Ubuntu 14.04 LTS (64-bit) e os procedimentos a seguir podem variar para versões anteriores.

2.1 Instalação de dependências Gstreamer e Python

O projeto utiliza algumas APIs importantes para transmissão de dados, utilização de bibliotecas Python. O seguinte comando pode ser executado como no Código 3.

```
sudo apt-get install gst* python-pip python-bottle imagemagick
```

Código 3: Instação de dependências para Gstreamer e bibliotecas Python.

Com a instalação completa, é possível utilizar o GStreamer para transmissão de voz e a biblioteca bottle para Python importante para criação do servidor de arquivos. Imagemagick é utilizado para conversão de imagens.

2.2 Instalação estéreoVision

O pacote estéreoVision é utilizado para fazer a calibração e processamento das imagens capturadas. É ele que gera a nuvem de pontos. Sua instalação pode ser feita pelo comando como no Código 4.

```
sudo pip install estéreoVision
```

Código 4: Instação do pacote Python estéreoVision.

Este depende de outra biblioteca Python chamada progressbar. A versão dela nos repositórios oficiais do Ubuntu é a 2.3, mas não funcionará corretamente. Baixe a versão 2.2 e instale ou copie os arquivos junto com este relatório para o diretório correto fazendo os procedimentos a seguir.

Supondo que o arquivo está na pasta /home/usuario/Downloads execute os comandos do Código 5.

```
cd ~/Downloads
unzip bibliotecas
sudo cp -ar ~/Downloads/bibliotecas/usr/local/lib/python2.7/dist-packeges/* /usr/local/lib/python2.7/dist-packeges
```

Código 5: Instação de dependências do pacote estéreoVision.

2.3 Instalação OpenCV e dependências

A obtenção das imagens, obtenção do mapa de disparidade, processamento de nuvem de pontos é possível com as bibliotecas do OpenCV e PCL. Elas podem ser instaladas com o comando no Código 6.

```
sudo apt-get install opencv* libpcl* python-opencv mayavi2
```

Código 6: Instação das bibliotecas do OpenCV e PCL.

2.4 Informações de Suporte Técnico

Alguns problemas podem ocorrer ao executar cada comandos. A seguir, os mais comuns.

Nenhuma foto é capturada

- Certifique-se que as webcams estão conectadas e na ordem correta. Utilize o programa Cheese e em preferências descubra se a câmera da esquerda está no caminho /dev/video0 ou /dev/video1.

Nenhum arquivo é transferido entre computadores

- Certifique-se que foi instalado o API python-bottle
- Feche qualquer programa que possa estar utilizando as portas 8080, 8081 ou 8082.
- Tenha Python 2.7 instalado (pode não funcionar com versão 3 do Python).

Não é possível transmitir voz entre computadores

- Certifique-se que o microfone esteja conectado corretamente e funcionando.
- Certifique-se que o Gstreamer esteja instalado.
- A voz é codificada com o codec Speex utilizando o GStreamer. Baixe os plugins good do Gstreamer.