João Otávio Brandão Antunes de Lucena

REALIDADE AUMENTADA FACIAL USANDO RASTREAMENTO DE CARACTERÍSTICAS NATURAIS

Trabalho de Conclusão de Curso

Universidade Federal Rural de Pernambuco
secretaria@preg.ufrpe.br
http://www.ufrpe.br/br/graduacao

RECIFE
2016

Universidade Federal Rural de Pernambuco
Departamento de Estatística e Informática
Bacharelado em Ciência da Computação

João Otávio Brandão Antunes de Lucena

REALIDADE AUMENTADA FACIAL USANDO RASTREAMENTO DE CARACTERÍSTICAS NATURAIS

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: *João Paulo Silva do Monte Lima*

RECIFE
2016

MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO (UFRPE)
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

http://www.bcc.ufrpe.br

FICHA DE APROVAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO

Trabalho defendido por João Otávio Brandão Antunes de Lucena como requisito para conclusão do curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, intitulado *Realidade aumentada facial usando rastreamento de características naturais*, orientado por João Paulo Silva do Monte Lima e aprovado pela seguinte banca examinadora:

João Paulo Silva do Monte Lima
DEINFO/UFRPE

João Marcelo Xavier Natário Teixeira
DEINFO/UFRPE

Valmir Macário Filho
DEINFO/UFRPE

Dedico este trabalho a minha mãe, avó, noiva, amigos e professores, os quais foram os principais responsáveis por me apoiar nessa longa jornada.

Agradecimentos

Primeiramente, agradeço a minha mãe, Ligya, por ter me proporcionado uma vida digna, confortável e repleta de ensinamentos, sendo a principal responsável por me fazer chegar até este momento. Ela sempre foi meu porto seguro e minha fortaleza, e, sem ela, eu não teria forças para enfrentar as batalhas diárias que a vida nos submete.

Em segundo lugar, agradeço a uma pessoa que está comigo a pouco tempo, mas que teve participação fundamental no período de escrita deste trabalho: minha noiva, Juliana. Ela foi fundamental nos momentos mais difíceis desta jornada, me fornecendo amor, carinho, companheirismo e cumplicidade quando eu mais necessitei.

Agradeço também a minha querida avó, Laurenita, a minha segunda mãe, essencial no meu crescimento e desenvolvimento educacional no decorrer da vida. Na ausência de minha mãe, ela me proveu tudo que precisei para ter uma vida digna.

Ao meu tio, Luciano, também deixo meus sinceros agradecimentos, por sempre ter sido como um pai para mim, me apoiando e me fazendo sorrir no momentos mais complicados. Ele é um exemplo de homem. Chegou a beira da morte por conta de uma doença, e sobreviveu, para me guiar nos caminhos mais obscuros.

Por fim, agradeço a instituição Universidade Federal Rural de Pernambuco, incluindo todos os professores e amigos, por terem sido a espinha dorsal da minha formação acadêmica, uma conquista que me fez evoluir como homem e adquirir sabedoria e conhecimentos que vou levar para resto da minha vida.

The Imagination is the most real world that we know, because each of us knows from own experience.

—JOHN FRUSCIANTE

Resumo

A Realidade Aumentada é uma forma de Realidade Virtual que permite a adição de informações virtuais no mundo real por meio de sensores (como câmeras de vídeo). Pesquisas avançadas incluem o uso de dados em rastreamento de movimento, reconhecimento de marcadores (como códigos 2D) e construção de ambientes controlados contendo sensores e atuadores. Nos dias de hoje, pacotes de desenvolvimento de software fornecem uma gama de ferramentas e bibliotecas para facilitar o desenvolvimento de aplicações de Realidade Aumentada utilizando marcadores físicos.

Em contrapartida, algoritmos de detecção e rastreamento de características naturais – em tempo real – tornaram-se, há algum tempo, técnicas amplamente utilizadas e eficientes, como uma alternativa para o sensoriamento de cenários do mundo real. O rastreamento de faces é um exemplo dessa evolução. Rastrear pontos de interesse da face permite o desenvolvimento de aplicações robustas sem a necessidade de rastrear qualquer objeto que não seja natural. Unindo Realidade Aumentada Facial e algoritmos de rastreamento de face é possível desenvolver aplicações interessantes em áreas como entretenimento, venda de acessórios, maquiagem, etc.

Este trabalho tem como objetivo principal criar uma solução para o desenvolvimento de aplicações de Realidade Aumentada, utilizando um algoritmo de rastreamento de características faciais em um cenário real e técnicas de visão computacional. Foi realizado um estudo acerca dos algoritmos de detecção e rastreamento de faces, a partir do qual um deles foi escolhido como base para o desenvolvimento de funções de Realidade Aumentada que permitam a inserção de objetos virtuais 2D na face do usuário, dispensando o uso de marcadores fiduciais planares para este fim.

Uma aplicação de autoria foi desenvolvida para a validação das funções, avaliação de desempenho – feita pela análise da taxa de quadros por segundo – e avaliação qualitativa, embasada na qualidade do alinhamento entre o virtual e o real. As funções foram disponibilizadas em um repositório na internet.

Palavras-chave: Realidade Aumentada, marcadores, rastreamento de faces, visão computacional

Abstract

Augmented Reality is a Virtual Reality form that allows adding virtual information to the real world by means of sensors (such as video cameras). Advanced researches include the use of data in motion tracking, recognition of physical markers (such as 2D codes) and construction of controlled environments containing sensors and actuators. Nowadays, software development kits provide a range of tools and libraries in order to facilitate the development of Augmented Reality applications that use physical markers.

In contrast, real time feature detection and tracking algorithms became for quite some time a set of widely used and efficient techniques, as alternative for sensing real world scenarios. Face tracking is an example of this evolution. Face landmarks tracking by means of various existing techniques allow the building of robust applications without using any object that is not natural. By joining Facial Augmented Reality and face tracking algorithms it is possible to develop interesting applications in areas such as entertainment, accessories sales, makeup, etc.

This work aims to create a solution for Augmented Reality application development by using a facial feature tracking algorithm in a real scenario and computer vision techniques. A detailed study about face detection and tracking algorithms was peformed, which on of them was chosen as basis to develop Augmented Reality functions that allow the insertion of 2D virtual objects on the user's face, without needing fiducial planar markers.

An application was developed to validate the functions, evaluate the performance – performed by analyzing framerate – and proceed a qualitative evaluation, based on the quality of alignment between virtual and real elements. The functions are available in an internet repository.

Keywords: Augmented Reality, markers, face tracking, computer vision

Lista de Figuras

Lista de Pseudocódigos

Lista de Acrônimos

RA	Realidade Aumentada
RV	Realidade Virtual
QR	Quick Response
2D	Two Dimensions
3D	Three Dimensions
SDK	Software Development Kit
IDE	Integrated Development Environment
MAR	Markerless Augmented Reality
TLD	Tracking-Learning-Detection
DOF	Degrees of Freedom
CLM	Constrained Local Model
CLM-Z	3D Constrained Local Model
RGB	Red, Green, Blue
FPS	Frames per Second
iOS	IPhone Operating System
PNG	Portable Network Graphics
MVS	Microsoft Visual Studio
JPG	Joint Photographic Experts Group
RGBA	Red, Green, Blue, Alpha
RGB-D	Red, Green, Blue, Depth

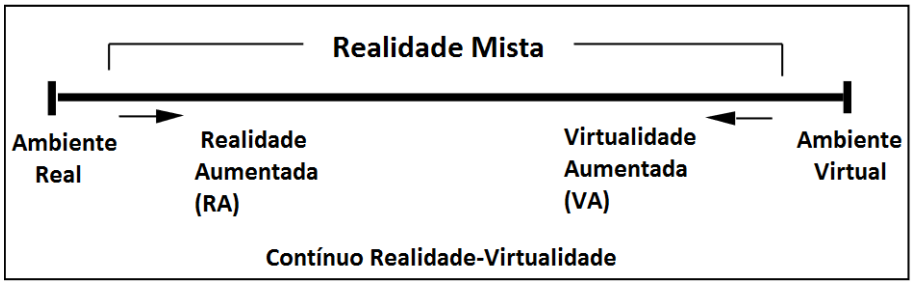
Sumário

1

Introdução

Define-se Realidade Aumentada (RA) como a inserção de elementos virtuais – em tempo real – no ambiente real, dando a impressão de que eles pareçam fazer parte do mesmo (AZUMA, 1997). Considera-se que RA está indiretamente relacionada à Realidade Virtual (RV). Todavia, em RV, o usuário é totalmente imerso ao mundo virtual, que pode ou não imitar propriedades de ambiente do mundo real, mas que também pode ultrapassar os limites da realidade física (MILGRAM et al., 1995). Em vez de definir os dois conceitos como opostos, é mais conveniente visualizá-los em lados oposto de um contínuo chamado Realidade-Virtualidade, como ilustrado no modelo da figura abaixo.

Figura 1.1: Relação entre RA e RV no Contínuo Realidade-Virtualidade



Fonte: (MILGRAM et al., 1995)

Atualmente, a maioria das pesquisas em RA está atrelada ao uso de vídeos transmitidos ao vivo (capturados por câmeras), os quais são digitalmente processados e "ampliados" pela adição de objetos gráficos modelados por um computador. Deste modo, sistemas de RA precisam rastrear o mundo real de forma a inserir corretamente tais objetos virtuais. Uma maneira comumente adotada para realizar essa tarefa é detectando marcadores fiduciais planares (Figura 1.2), usando uma câmera de vídeo (KATO; BILLINGHURST, 1999). Outro exemplo usual destes marcadores é o código de barras bidimensional, chamado Código QR (Figura 1.3). Além destes, qualquer objeto do mundo real pode ser usado como marcador.

Contudo, muitas aplicações de RA encontram limitações no uso desses marcadores, sobretudo em ambientes controlados. Uma solução seria o sensoriamento do mundo real por

Figura 1.2: Adição de objetos virtuais no mundo real por meio do rastreamento de marcadores fiduciais planares

Fonte: (KATO; BILLINGHURST, 1999)

Figura 1.3: Código de barras bidimensional, comumente chamado Código QR

Fonte: `http://www.magicwebdesign.com.br/blog/tecnologia/qr-code-e-um-mundo-de-possibilidades/`

meio da detecção e rastreamento de objetos reais, usando características naturais da cena. Em alguns cenários de aplicação, sistemas de RA precisam sobrepor conteúdo virtual na face do usuário. Para realizar este procedimento, tais sistemas necessitam detectar e rastrear faces em tempo real. A título de exemplo, o aplicativo de redes sociais Snapchat (SPIEGEL; MURPHY; BROWN, 2011), para dispositivos móveis, possibilita sobrepor animações digitais 3D na face rastreada do usuário, em tempo real, como forma de entretenimento, por meio da câmera do dispositivo (Figura 1.4).

Outra interessante aplicação neste âmbito é o XMING Augmented Face (LIVET; YANG, 2015). Por meio de um robusto rastreamento de face e orientação da cabeça – e também em tempo real – o usuário está apto a aumentar faces (inserir objetos virtuais na face) usando

Figura 1.4: Animações 3D sobrepostas na face do usuário pelo aplicativo Snapchat

Fonte: http://www.imore.com/how-add-emoji-filters-lenses-text-and-sketches-snapchat

acessórios virtuais em formato 3D, como óculos digitais (Figura 1.5). Observa-se na figura que, mesmo com os movimentos de rotação da cabeça em ângulos em que os olhos continuem à mostra, a posição dos óculos virtuais inserido não muda e eles se mantêm fixados à face.

Portanto, nota-se que, para realizar o aumento de faces, tais aplicações necessitam de alguma técnica de rastreamento de faces em tempo real. Por meio da utilização de métodos de processamento de imagens e visão computacional, como estimação de pose e transformações geométricas de imagens (HARTLEY; ZISSERMAN, 2003), é possível realizar a correspondência entre planos distintos, conforme ilustrado nas Figuras 1.4 e 1.5. Combinando algum desses métodos com um algoritmo de rastreamento de faces, a inserção de elementos virtuais casados com a face do usuário pode ser amplamente facilitada.

Figura 1.5: Óculos virtual 3D sobreposto à face do usuário pela aplicação XMING Augmented Face

Fonte: O autor

1.1 Problema de pesquisa

Partindo do contexto explanado na seção acima, a definição do problema de pesquisa deste presente trabalho pode ser enumerado pela seguinte questão: como fornecer uma forma acessível e eficaz de inserir objetos virtuais sobre a face do usuário em tempo real?

1.2 Objetivos

Tendo em vista o problema de pesquisa citado na seção anterior, os objetivos deste trabalho são elencados nas subseções seguintes.

1.2.1 Objetivo geral

O objetivo geral deste trabalho é propor um conjunto de funções algorítmicas, de código aberto (*open source*), com o intuito de auxiliar o aumento de faces, em tempo real, sem a necessidade da utilização de marcadores, mediante o rastreamento de pontos de referências faciais.

1.2.2 Objetivos específicos

Considerando o objetivo geral da subseção antecedente, este projeto de pesquisa visa alcançar os seguintes objetivos específicos:

- Propor funções algorítmicas para facilitar a inserção de dados virtuais na faces, a partir de rotinas de manipulação de imagens e visão computacional preexistentes;
- Aplicar as funções propostas junto a um algoritmo de rastreamento de faces previamente selecionado;
- Aplicar as funções implementadas em um sistema de autoria, com a finalidade de validar e viabilizar as mesmas;
- Avaliar as funções implementadas, de forma a verificar se a mesma contribui para o desenvolvimento de aplicações RA que necessitam sobrepor conteúdo virtual à face de usuários.

1.3 Metodologia

Aqui é descrito o processo de desenvolvimento do presente trabalho, o qual iniciou-se em meados de fevereiro de 2015, fruto de um projeto de pesquisa de Iniciação Científica Voluntária, orientado pelo Prof. João Paulo Silva do Monte Lima. Em primeiro lugar, foi definido o âmbito

de visão computacional como área de concentração. A partir desta definição, as linhas de pesquisa também puderam ser estabelecidas. São elas: Realidade Aumentada e Rastreamento. Posteriormente, foi fundamental uma revisão bibliográfica aprofundada, a partir de trabalhos já conceituados na literatura.

Sendo assim, conceitos de Realidade Aumentada foram extraídos de (AZUMA, 1997), uma das literaturas mais citadas em trabalhos acadêmicos de RA, pioneiro nesta área. Em seguida, foram explorados trabalhos sobre rastreamento de faces, como (KATO; BILLINGHURST, 1999), (VIOLA; JONES, 2004), (KALAL; MIKOLAJCZYK; MATAS, 2010), (CRISTINACCE; COOTES, 2006), (BALTRUŠAITIS; ROBINSON; MORENCY, 2012), (FANELLI et al., 2011) e (WEISE et al., 2011). Tal pesquisa foi primordial para a seleção do algoritmo de rastreamento de face apropriado, usado como base para elaborar a solução proposta neste trabalho. Após este estudo, o CLM (BALTRUŠAITIS; ROBINSON; MORENCY, 2012) foi o escolhido, uma vez que encontra-se disponível na internet e apresenta resultados satisfatórios.

Em um momento posterior, foi feita um pesquisa acerca dos pacotes de desenvolvimento de software (SDKs) existentes para RA, com o objetivo de encontrar rotinas capazes de inserir objetos virtuais na face do usuário, em uma cena real, sem o uso de marcadores físicos. A tabela de comparação contida em Social Compare (FRUHINSHOLZ; BERGER; ELZIERE, 2013) foi de suma importância para esta atividade. Após analisar tais bibliotecas, foi realizada uma pesquisa a respeito de aplicações de RA existentes relacionadas ao projeto de pesquisa corrente, com o intuito de encorajar o desenvolvimento de aplicações que tomam proveito do rastreamento de faces baseado em características naturais. No Capítulo 2 são mostradas estas aplicações.

Finalmente, foi feita a implementação de funções – tomando como base o algoritmo de rastreamento de faces escolhido – para sobrepor conteúdo virtual 2D à face do usuário, em tempo real. Estas utilizaram-se de rotinas de processamento de imagens e visão computacional preexistentes. Vale ressaltar o uso do ambiente de desenvolvimento integrado (IDE) Microsoft Visual Studio 2012 (MICROSOFT, 2011). As funções foram desenvolvidas em C++, com o auxílio da biblioteca de visão computacional OpenCV (BRADSKI; KAEHLER, 2008) (ITSEEZ, 2000a). A mesma foi escolhida por ser uma das bibliotecas mais consolidadas na área. A validação das funções implementadas foi feita por uma aplicação de autoria, que demonstra o uso das funções em um ambiente real. A partir desta aplicação, foram feitas análises de qualidade e desempenho. Os pseudocódigos da implementação podem ser visto no Capítulo 3, e os resultados são apresentados no Capítulo 4.

1.4 Estrutura do trabalho

O restante deste trabalho estrutura-se de maneira a, primeiramente, definir a problematização e, em seguida, levantar os trabalhos relacionados como forma de justificativa, para que posteriormente seja abordada a solução proposta.

Deste modo, a organização dos próximos capítulos é definida da seguinte forma: o Capítulo 2 versa acerca dos resultados da pesquisa feita sobre algoritmos de rastreamento de faces (Seção 2.1) e aplicações de RA existentes que estão diretamente relacionadas com este trabalho (Seção 2.2). Esta última teve o intuito de justificar o desenvolvimento da solução proposta. Ainda neste capítulo, na Seção 2.3, são expostas as lições aprendidas e as dificuldades encontradas na pesquisa realizada. O desenvolvimento da solução proposta é apresentado no Capítulo 3. No início, é feita uma breve introdução das ferramentas utilizadas na construção da solução. Em seguida, é feita uma descrição sobre o CLM, algoritmo de rastreamento de face utilizado como base da solução, para que, na Seção 3.2, seja descrito em detalhes como foi realizado o desenvolvimento das funções, bem como uma conceituação da implementação das mesmas, descrevendo a metodologia usada para tal, os pseudocódigos criados e os resultados parciais obtidos. Ainda neste capítulo, é apresentado o sistema de autoria construído para testes e validação das funções (Seção 3.3); e a Seção 3.4 descreve algumas conclusões extraídas do desenvolvimento da solução, como dificuldades encontradas e justificativa dos métodos utilizados.

No Capítulo 4 são apresentados os resultados provenientes da aplicação de autoria. Em seguida, a Seção 4.1 descreve as avaliações de desempenho e qualidade extraídas da execução da aplicação. No final, a Seção 4.2 levanta as lições e dificuldades encontradas na validação da solução. Finalmente, o Capítulo 5 encerra o trabalho explanado neste documento, descrevendo as conclusões e trabalhos futuros.

2

Realidade Aumentada Facial

A Realidade Aumentada vem demonstrando ser um ramo de vanguarda da computação, combinando conceitos e técnicas das áreas de visão computacional, computação gráfica e processamento de imagens. Atualmente existe uma gama de aplicações de RA, facilmente encontradas na internet ou em *stores* de aplicativos para dispositivos móveis. Diversas delas fazem o uso de marcadores para a realizar a tarefa de rastreamento do ambiente real. Outras, utilizam-se da detecção e reconhecimento de características naturais para inserção de objetos virtuais em cenários reais. Tais características podem ser extraídas da face do usuário, como visto no Capítulo 1. Estas aplicações implantam técnicas de RA sem marcadores (Markerless Augmented Reality - MAR). Nas seções seguintes são apresentados estudos acerca de técnicas de algoritmos de detecção e rastreamento de faces e aplicações de RA que tomam proveito das mesmas.

2.1 Algoritmos de detecção e rastreamento de faces

Nos dias atuais, os sistemas de detecção e rastreamento de características naturais no mundo real estão se tornando cada vez mais robustos e eficazes. Derivado deste fato, os métodos de detecção e rastreamento de faces também encontram-se em um patamar muito além da qualidade dos algoritmos há alguns anos atrás. Os fatores que induzem a erros como iluminação, ruídos, movimento da cabeça e oclusão estão sendo tratados de forma cada vez mais precisa. Tendo em conta tais fatores, o restante desta seção expõe técnicas algorítmicas para detecção e rastreamento de faces aplicadas em câmeras de vídeo em tempo real.

A partir da técnica de detecção 2D do método Viola-Jones (VIOLA; JONES, 2004) e o rastreamento 2D do método Tracking-Learning-Detection (TLD) (KALAL; MIKOLAJCZYK; MATAS, 2010), os quais proveem uma área da imagem onde o alvo está localizado, é possível detectar e identificar faces. Os resultados dos métodos podem ser vistos nas Figuras 2.1 e 2.2, respectivamente.

Entretanto, mesmo que tais técnicas forneçam informações sobre a localização 2D e a escala das faces, elas não são capazes de estimar a orientação da cabeça. Em contrapartida,

Figura 2.1: Resultado da detecção de faces do método Viola-Jones

Fonte: (VIOLA; JONES, 2004)

algumas técnicas são capazes de realizar tal estimativa com seis graus de liberdade (Degrees of Freedom - DOF) completos (MCNEELY; PUTERBAUGH; TROY, 2005), representados pelo movimento de translação e rotação, tais como as técnicas de Modelo Local Restrito (Constrained Local Model - CLM) (CRISTINACCE; COOTES, 2006) e Modelo Local Restrito3D (CLM-Z) (BALTRUŠAITIS; ROBINSON; MORENCY, 2012). O CLM-Z é, tanto na precisão quanto na taxa de convergência, mais eficiente em relação ao CLM, pois ele usa um sensor de profundidade – *Kinect* – que provê em resultados melhores que os do CLM, o qual por sua vez utiliza apenas imagens no modelo RGB. Na Figura 2.3, as 8 imagens no topo são resultados do rastreamento obtido pelo CLM, mostrando algumas falhas quando ocorre movimentos da cabeça. Em contrapartida, os resultados do CLM-Z (8 imagens mais abaixo da Figura 2.3) apresenta um maior nível de precisão.

Outros métodos robustos são as Florestas de Regressão Aleatórias (Discriminative Random Regression Forests - DRRF) (FANELLI et al., 2011) e o Faceshift (WEISE et al., 2011), os quais também são capazes de realizar o rastreamento com seis graus de liberdade. Rastreadores de face também podem ser classificados como rígidos e não-rígidos (BALTRUŠAITIS; ROBINSON; MORENCY, 2012). O método de rastreamento rígido busca estimar a pose da cabeça, ou seja, a sua localização e possivelmente sua orientação (VIOLA; JONES, 2004) (FANELLI et al., 2011) (KALAL; MIKOLAJCZYK; MATAS, 2010). Por outro lado, métodos não-rígidos são capazes de rastrear posições dos pontos de referência da face, tais como olhos, nariz e boca (CRISTINACCE; COOTES, 2006) (BALTRUŠAITIS; ROBINSON; MORENCY, 2012) (WEISE et al., 2011).

Figura 2.2: Resultado do rastreamento de faces do método TLD

Fonte: (KALAL; MIKOLAJCZYK; MATAS, 2010)

Figura 2.3: Resultados do rastreamento de face feitos pelo CLM (8 quadros acima) e CLM-Z (8 quadros abaixo)

Fonte: (BALTRUŠAITIS; ROBINSON; MORENCY, 2012)

Com as técnicas descritas nesta subseção, é possível desenvolver aplicações em diversas áreas de concentração, incluindo RA. A próxima seção apresenta algumas aplicações, no contexto de RA, desenvolvidas a partir do uso de técnicas de detecção e rastreamento de faces, como as vistas aqui.

2.2 Aplicações de Realidade Aumentada Facial

Tendo em vista os métodos e técnicas de detecção e rastreamento de faces discutidos anteriormente, a seção atual descreve as aplicações exploradas que se aproveitam de tais técnicas.

Utilizando um sensor *Kinect*, a loja Timberland introduziu uma aplicação para detectar e sobrepor roupas e acessórios sobre o corpo do usuário (ORANGE, 2005). A aplicação captura uma imagem do corpo humano e, a partir dessa imagem, o usuário está apto a interagir com esta, por meio de movimento dos membros, inserindo acessórios virtuais em seu próprio corpo, como roupas e calçados (Figura 2.4). Todavia, apesar de haver o rastreamento do corpo e detecção de movimentos, tal aplicação não se aproveita de técnicas de visão computacional para manipulação de face, como visto na Seção 2.1.

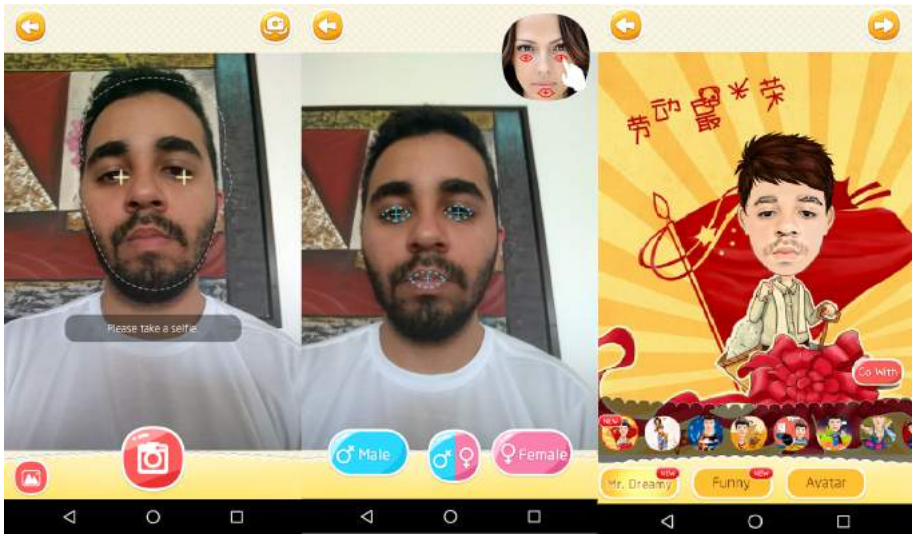
Figura 2.4: Sobreposição de roupas virtuais pela aplicação da loja Timberland

Fonte: (ORANGE, 2005)

Por outro lado, o aplicativo Cartoon Face (TEAM, 2012) restringe-se ao rastreamento de face para realizar seus objetivos. Assim como a aplicação da loja Timberland (ORANGE, 2005), é realizado previamente a captura de uma imagem. Posteriormente, a imagem é remodelada para formar uma caricatura da face do usuário (Figura 2.5). Apesar de exibir ótimos resultados no mapeamento da face e na modelagem virtual, o Cartoon Face necessita de uma imagem estática – mais precisamente uma foto – como entrada, contrariando os conceitos de RA, cuja definição determina que o aumento de face deve ser feito em tempo real, por meio de uma câmera de vídeo (AZUMA, 1997).

Em contrapartida, outras aplicações baseiam-se nesta captura de imagens em tempo real para inserir objetos virtuais no mundo real. O aplicativo TAT Augmented ID (INTERFACES, 2005) usa reconhecimento facial para identificar informações pessoais acerca da pessoa rastreada. Após a identificação, a aplicação insere conteúdo virtual na tela com informações pessoais, como nome, profissão e *links* para os perfis de redes sociais associadas à tal pessoa (Figura 2.6).

Figura 2.5: Caricatura da face aplicada pela modelagem do aplicativo Cartoon Face



Fonte: O autor

Entretanto, o objetivo final de tal aplicação não condiz com o intuito de sobrepor acessórios virtuais no rosto rastreado. Sobre outra perspectiva, imagens em 3D podem ser sobrepostas pela aplicação de *desktop* Transformer Augmented Reality (INTERACTIVE, 2005), cujo resultado pode ser visto na Figura 2.7. Além deste, a aplicação 3D Face Tracking (MARIN, 2005) também é capaz de rastrear faces e inserir objetos 3D. O diferencial em relação ao Transformer Augmented Reality é que o usuário, em primeiro lugar, deve interagir com a aplicação, inserindo uma *face box mask* – caixa em formato de máscara de um rosto padrão pré-calculada – para demarcar a área da face a qual será rastreada (Figura 2.8).

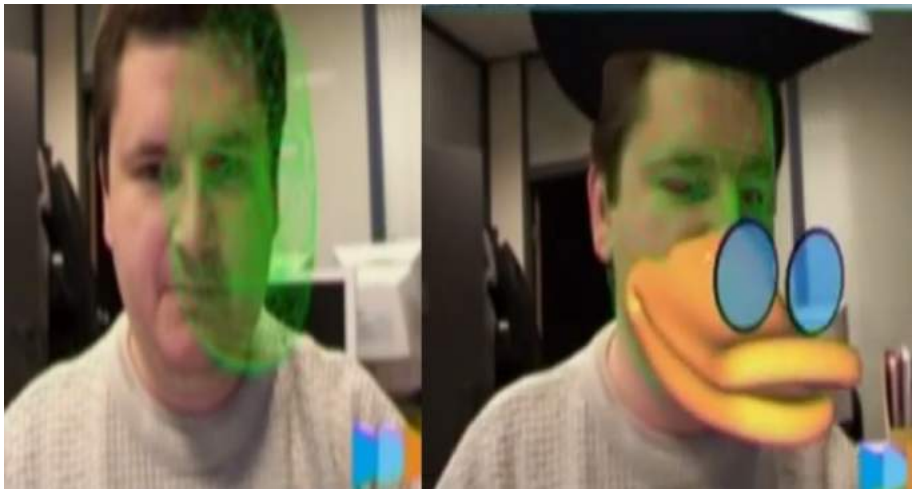
Figura 2.6: Reconhecimento facial e inserção de informações virtuais no aplicativo TAT Augmented Id

Fonte: Internet

Figura 2.7: Máscara virtual 3D inserida na face do usuário pela aplicação Transformer Augmented Reality

Fonte: (INTERACTIVE, 2005)

Figura 2.8: Mascára virtual 3D inserida na face por meio de uma *face box mask* pela aplicação 3D Face Tracking



Fonte: (MARIN, 2005)

Em tempo real, o aplicativo Selfit (LTD, 2005), desenvolvido para iOS, permite inserir, a partir de pontos de referência da face (como olhos, boca e nariz, por exemplo), máscaras virtuais 2D no rosto do usuário. O resultado pode ser visto na Figura 2.9. O usuário é capaz de interagir com o aplicativo, escolhendo outros formatos de máscara e animações. Outra aplicação interessante é o AR Face Book (PENG, 2015), que toma proveito do Classificador de Haar (GAO; LU, 2008), adaptado para detectar características faciais eliminando falsos positivos, facilitando a inserção de modelos virtuais 3D na face (Figura 2.10). Nesta aplicação, foi usada a biblioteca OpenCV – mais especificamente o algoritmo de Classificação de Haar pré-definido – e a linguagem de programação C++.

Figura 2.9: Máscara virtual 2D inserida na face do usuário pelo aplicativo Selfit

Fonte: (LTD, 2005)

Figura 2.10: Máscara virtual 3D sobreposta na face do usuário pelo aplicativo AR Face Book

Fonte: PENG (2015)

Snapchat (SPIEGEL; MURPHY; BROWN, 2011) (citado no Capítulo 1), Selfit (LTD, 2005), Transformer Augmented Reality (INTERACTIVE, 2005), 3D Face Tracking (MARIN, 2005) e AR Face Book PENG (2015) são capazes de inserir máscaras virtuais utilizando como base de projeção final o contorno da face, encobrindo todo o rosto do usuário. Outras aplicações são capazes de usar pontos de referência específicos do rosto para inserir objetos virtuais de menores proporções. O aplicativo MSQRD (INC, 2015) é uma dessas aplicações. Além de ser capaz de inserir objetos virtuais que encobrem toda a face, ele também é capaz de aumentar faces com objetos menores, como ilustrado na Figura 2.11.

Figura 2.11: Objetos virtuais inseridos em pontos de referência do rosto pelo aplicativo MSQRD

Fonte: `http://andro4all.com/2016/03/msqrd-aplicacion-transformar-cara`

O XZMING Augmented Face (LIVET; YANG, 2015), de antemão citado no Capítulo 1, também segue os mesmo conceitos do MSQRD. Assim como o Atol les Opticiens (OPTICIENS, 2015) – aplicativo para óticas implantado na França – que também provê a inserção de óculos virtuais 3D na face do usuário em tempo real (Figuras 2.12 e 2.13). Esta foi implantada no mercado para auxiliar a experimentação de óculos em óticas. Com isso, o cliente não necessita estar presente na ótica para realizar compras de óculos, visto que ele está apto a usar o aplicativo no dispositivo móvel, experimentando quaisquer óculos à venda por meio da função de aumento de face provida pela aplicação.

Figura 2.12: Óculos virtuais 3D inseridos na face do usuário pelo aplicativo Atol les Opticiens

Fonte: http://echip.com.vn/thuc-tai-tang-cuong-a20130314160058393-c1107.html

Figura 2.13: Óculos virtuais 3D inseridos na face do usuário pelo aplicativo Atol les Opticiens visto de um angulo diferente

Fonte: http://www.t-immersion.com/project-gallery/atol-face-tracking-application-set-sights-higher

2.3 Discussão

Neste capítulo foram apresentados trabalhos e aplicações que serviram como base de estudo, essencial para explanar os recursos que podem ser alcançados com a proposta de solução deste trabalho de pesquisa.

Em primeiro lugar, vale destacar que cada uma dessas aplicações foi desenvolvida pelo proveito de bibliotecas já existentes. No endereço do Social Compare (FRUHINSHOLZ; BERGER; ELZIERE, 2013) é possível encontrar uma gama de SDKs, possivelmente utilizadas para a criação das aplicações de RA citadas na seção anterior. Contudo, não é possível ter acesso aos códigos de tais aplicações. Em suma, os códigos são propriedades das empresas e de desenvolvedores autônomos que os implementam. Desta maneira, um desenvolvedor de sistemas de RA, que deseja criar aplicações no ramo de aumento de faces não possui acesso a tais código, os quais poderiam servir como base de seu desenvolvimento. Além disso, não foi possível utilizar e testar todos os SDKs existentes no Social Compare, com a finalidade de encontrar funções de aumento de faces predefinidas, visto que o custo relacionado ao tempo para configurar tais bibliotecas é alto, além de ser uma atividade um tanto trabalhosa. Todavia, foi possível usufruir de funções básicas da biblioteca Metaio, uma das pioneiras e mais habitualmente utilizada no âmbito de RA. Porém, tal extensão básica não fornece rotinas de aumento de faces sem o uso de marcadores. Uma outra dificuldade é o fator comercial. As funções básicas das bibliotecas de RA estão disponíveis, porém outras funções só podem ser adquiridas pelo pagamento da extensão comercial. Esta é outra restrição para os desenvolvedores, pois torna-se economicamente custoso aderir a estes pacotes, os quais, de modo geral, são muito caros.

Todavia, as aplicações de RA apresentadas nesse capítulo aproveitam-se de métodos de visão computacional para realizar seus objetivos. São conhecidas diversas bibliotecas da área, como o CCV (NUIGROUP, 2006) e o SimpleCV (INC, 2006). Outra, considerada a mais consolidada, é o OpenCV (ITSEEZ, 2000a) (BRADSKI; KAEHLER, 2008), amplamente usada no mercado tecnológico de visão computacional. A vantagem é que todas são código aberto (*opensource*), diferentemente de algumas bibliotecas de RA (FRUHINSHOLZ; BERGER; ELZIERE, 2013). Uma hipótese é que as aplicações de RA mostrada neste presente capítulo fazem uso de rotinas dessas bibliotecas. O AR Face Book (PENG, 2015), por exemplo, foi implementado por meio do OpenCV. Entretanto, não são conhecidas rotinas pré-definidas para o aumento de faces nas mesmas, que resultem, por exemplo, na inserção de acessórios virtuais das aplicações XZIMG (LIVET; YANG, 2015) e Atol les Opticiens (OPTICIENS, 2015). Para este fim, é necessário desenvolver algoritmos, por meio das ferramentas providas pelo OpenCV, CCV ou SimpleCV, por exemplo, como rotinas de projeção de imagens e transformações geométricas.

Contudo, seria interessante desenvolver rotinas funcionais com a intenção de auxiliar o desenvolvedor na construção de aplicações de aumento de faces sem o uso de marcadores, sem custo econômico e com um nível de qualidade e desempenho satisfatórios e, integrá-las à alguma biblioteca de visão computacional *open source*. Visto este conceito, a solução proposta por este

trabalho de pesquisa é definida da seguinte forma: uma função, que agregue um algoritmo de rastreamento de face e outras sub-rotinas de uma biblioteca de visão computacional existente, a qual o desenvolvedor forneça como entrada a imagem da câmera de vídeo, em tempo real, e a imagem do objeto virtual e obtenha como resultado o objeto sobreposto na face. O próximo capítulo descreve detalhadamente esta proposta.

3

Solução de Realidade Aumentada Facial

O presente capítulo apresenta a descrição detalhada da proposta deste trabalho. Como foi dito na Seção 1.3, os algoritmos implementados como solução para o problema de sobreposição de conteúdo virtual na face, sem o uso de marcadores, foram desenvolvidos na linguagem C++, por meio de rotinas da biblioteca OpenCV (BRADSKI; KAEHLER, 2008) (ITSEEZ, 2000a), na versão 2.4.11. Além disso, o CLM foi a técnica de rastreamento de face selecionada como base para a projeção dos objetos digitais. A solução proposta restringe-se ao uso de imagens de óculos 2D – diferentemente do XZIMG Augmented Face (LIVET; YANG, 2015), que utiliza óculos virtuais 3D, por exemplo – com extensão PNG e plano de fundo transparente. Para chegar a um primeiro resultado parcial, foi usada a imagem da Figura 3.1, encontrada aleatoriamente na internet. A implementação dos códigos foi feita no MVS 2012.

A próxima seção mostra uma breve explanação acerca do CLM, o algoritmo de rastreamento de face selecionado. Posteriormente, na Seção 3.2, é descrito detalhadamente o passo-a-passo da construção da solução algorítmica, definindo a conceituação teórica, os métodos utilizados e os resultados parciais obtidos. Por fim, a Seção 3.3 demonstra o uso das função em uma aplicação de autoria.

3.1 CLM

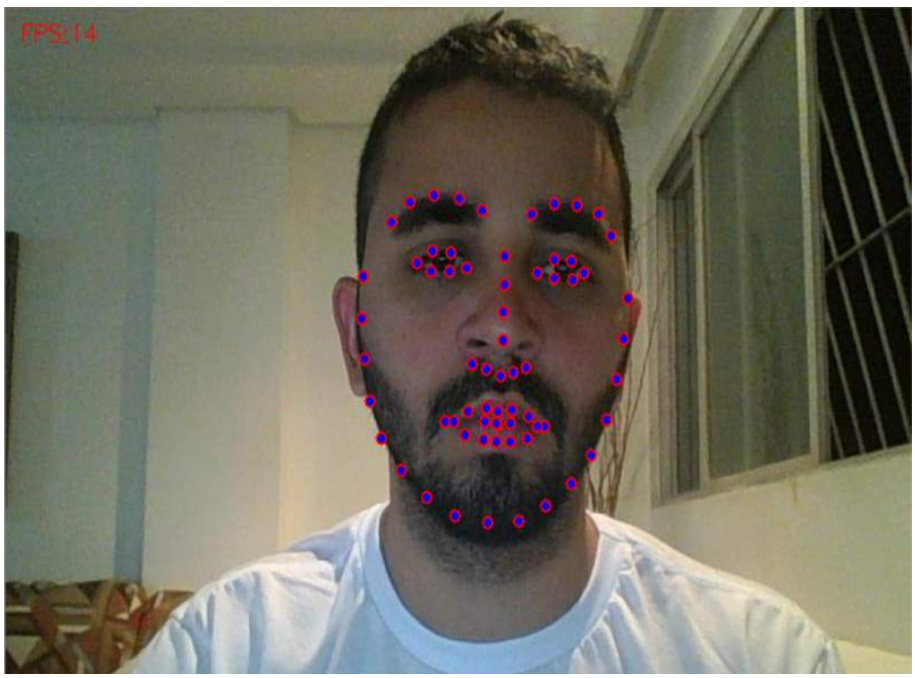
Antes de explanar a solução proposta na Seção 3.2, aqui é feita uma discussão sobre o CLM, com o intuito demonstrar o funcionamento do algoritmo, visando esclarecer o proveito de tal na implementação da solução. O CLM mapeia alguns pontos de referência da face, como destacado na Figura 3.2. Observe que são destacados pontos nos olhos, sobrancelhas, nariz, boca, contorno do bigode e contorno da face. Tal ilustração exibe o resultado do rastreamento quando a face está posicionada frontalmente à câmera. Esta é a posição da cabeça em que o CLM apresenta os resultados mais satisfatórios.

Como a técnica CLM funciona em câmeras com imagens em tempo real, haverá sempre uma variação dos pontos na imagem, ocasionada principalmente pelo movimento e orientação da cabeça do usuário no cenário real. Visto isso, é importante que a técnica consiga rastrear

Figura 3.1: Óculos virtual 2D usado na solução proposta

Fonte: O Autor

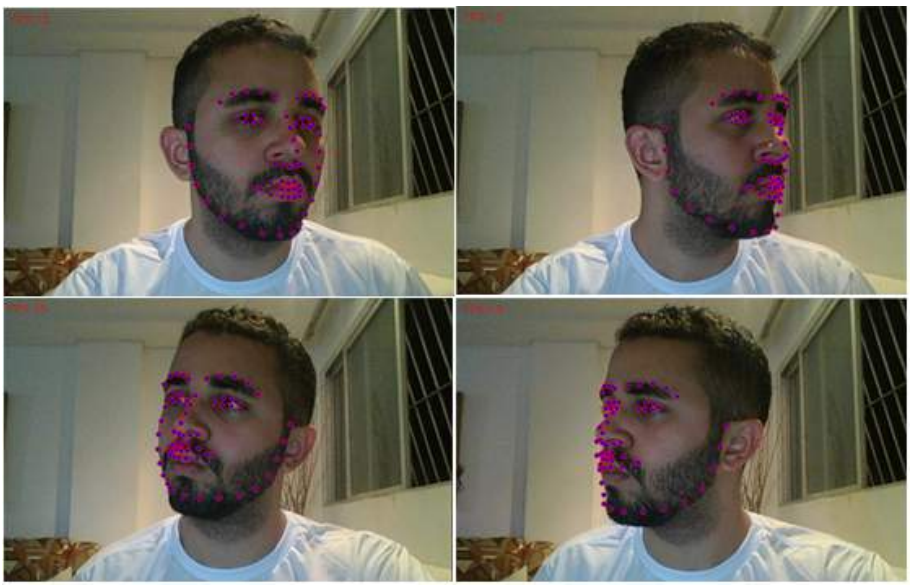
Figura 3.2: Pontos de referência da face destacados pelo algoritmo CLM na posição frontal



Fonte: O Autor

os pontos a partir da posição da cabeça. Na Figura 3.3, são ilustrados os resultados do CLM no movimento de rotação da cabeça, a qual o usuário realiza um movimento sobre os ombros esquerdo e direito. Observe que neste movimento um dos lados da face fica totalmente à amostra. Contudo, o outro lado fica ocludido. Com isso, os pontos de referência do lado da face ocludida se aglomeram, mas não somem, o que torna o resultado adequado.

Figura 3.3: Pontos de referência da face destacados pelo algoritmo CLM no movimento de rotação da cabeça



Fonte: O Autor

Os resultados dos movimentos de extensão e flexão são ilustrados nas Figuras 3.4 e 3.5, respectivamente. O movimento de extensão ocorre quando o usuário inclina a cabeça para trás, olhando para cima e estendendo o pescoço. Em contrapartida, no movimento de flexão, o usuário inclina a cabeça para baixo, buscando tocar o queixo no peito. Na Figura 3.4, observa-se uma falha mínima no rastreamento dos pontos de contorno da face. O movimento de flexão, ilustrado na Figura 3.5, apresentou melhores resultados.

Figura 3.4: Pontos de referência da face destacados pelo algoritmo CLM no movimento de extensão da cabeça

Fonte: O Autor

Figura 3.5: Pontos de referência da face destacados pelo algoritmo CLM no movimento de flexão da cabeça

Fonte: O Autor

Outro movimento em que o resultado do rastreamento pelo CLM é satisfatório é o movimento de inclinação lateral, visto na Figura 3.6, em que o usuário realiza a aproximação da orelha em direção ao ombro, em ambos os lados. Este movimento é mais fácil de ser rastreado pelo CLM, já que a posição da cabeça continua frontal.

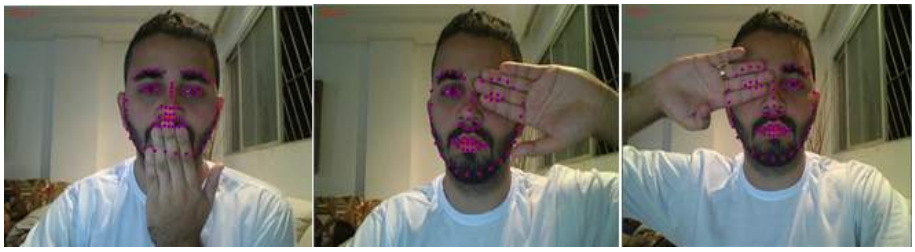
Figura 3.6: Pontos de referência da face destacados pelo algoritmo CLM no movimento de inclinação lateral da cabeça

Fonte: O Autor

Além de apresentar resultados adequados no rastreamento da face com diferentes orientação da cabeça, o CLM também é eficaz em casos de oclusão parcial dos pontos (Figura 3.7). Nesta ilustração, a mão do usuário obstrui os pontos de referência dos olhos e da boca. Apesar disso, a técnica se mantém eficaz e rastreia os pontos em posições adequadas, mesmo que parte da imagem não esteja visível.

Percebe-se que o CLM é eficaz em diversos pontos levantados. Contudo, é importante haver uma iluminação adequada no cenário, pois o algoritmo apresenta falhas ao rastrear a face

Figura 3.7: Pontos de referência da face destacados pelo algoritmo CLM quando ocorre oclusão



Fonte: O Autor

Figura 3.8: Falha no algoritmo CLM pelo fator iluminação

Fonte: O Autor

Figura 3.9: Falha do algoritmo CLM pelo fator reflexão

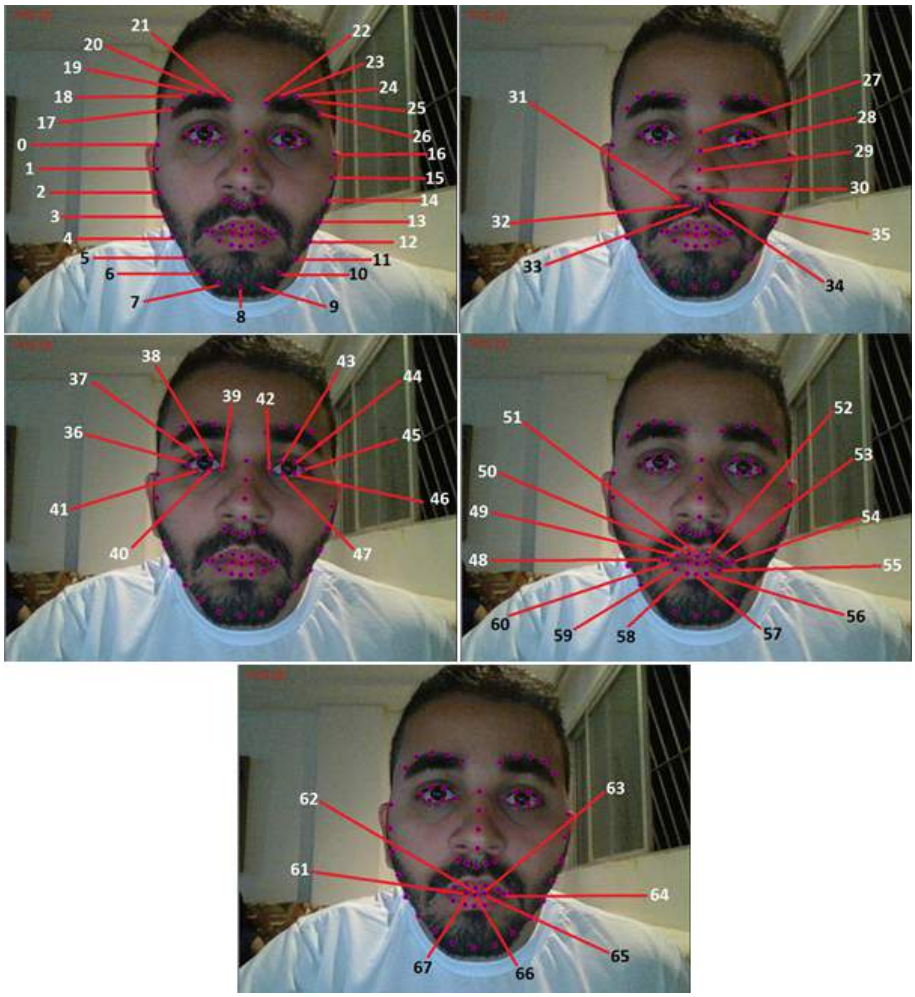
Fonte: O Autor

num cenário com baixa iluminação ou quando ocorre reflexão (por exemplo, quando a câmera está posicionada para uma janela e a luz solar reflete a imagem da câmera nela mesma). Tais falhas podem ser visualizadas nas Figuras 3.8 e 3.9. Observe que os pontos de referência faciais são rastreados incorretamente.

Cada ponto de referência da face é indexado por um valor fixo, como pode ser observado na Figura 3.10. Ao todo, são 68 pontos rastreados. Independentemente da orientação da cabeça ou de oclusões da face, os índices dos pontos não mudam. Nos pseudocódigos da próxima seção, o vetor que referencia os pontos da face (*PtsFace*) pode utilizar qualquer um dos 68 pontos para realizar a correspondência com a imagem do objeto. Vale salientar que não existe um limite de pontos pré-definidos para calcular a projeção em perspectiva, que também será vista na próxima seção.

Feita a análise dos resultados do CLM, a seção posterior apresenta a solução proposta por este presente trabalho, baseada no proveito do algoritmo aqui demonstrado.

Figura 3.10: Índices dos pontos de referência rastreados pelo CLM



Fonte: O Autor

3.2 Implementação

Foram implementadas quatro funções para se alcançar a solução proposta por este projeto. Cada pseudocódigo destacado nesta seção contém uma dessas funções, as quais ainda agregam rotinas preexistentes do OpenCV. As rotinas de transformação e projeção em perspectiva (Pseudocódigo 3) foram fundamentais para atingir os resultados pretendidos. Destaca-se também o proveito da função de combinação de imagens por meio do *alpha blending*, como pode ser visto no Pseudocódigo 4. Porém, antes de explorar os pseudocódigos, é fundamental que o leitor observe o embasamento teórico da solução de aumento de faces, de modo à facilitar sua compreensão em relação aos mesmos.

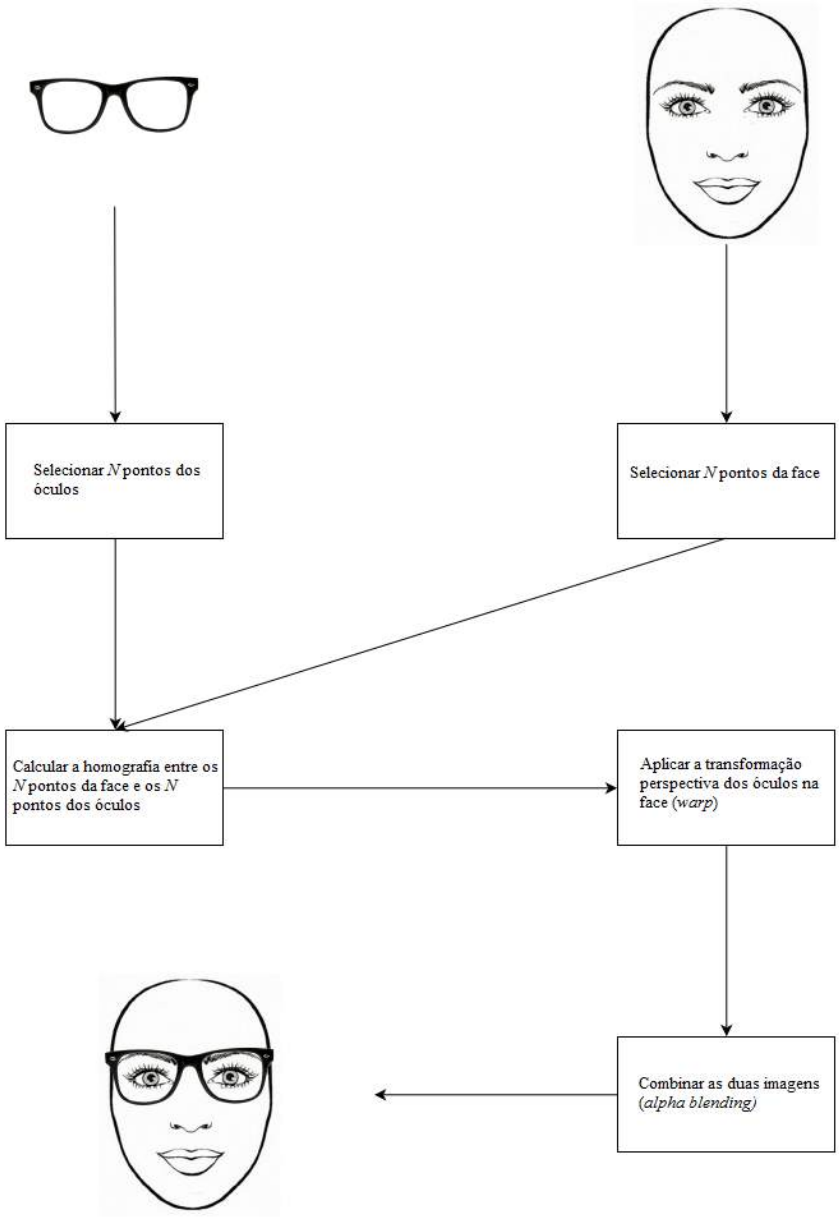
Inicialmente, a Figura 3.11 esquematiza uma visão teórica da solução de aumento de face. A mesma expõe de forma simplificada como foi realizada paulatinamente o desenvolvimento da solução. Preliminarmente, são escolhidos os pontos de referência do objeto virtual (óculos) e da face do usuário. Tais pontos são usados, a posteriori, como entrada para o mapeamento entre os pontos das duas imagens. Para isso, é fundamental que a quantidade N de pontos escolhida numa imagem seja a mesma na outra. Não existe restrição quanto ao valor de N. Na Figura 3.12, foram escolhidos previamente 5 pontos (em vermelho) de cada plano.

Subsequentemente, é feito o cálculo da homografia entre os pontos pré-selecionados dos dois planos distintos. A homografia em imagens 2D é a correspondência de um ponto (x_1, y_1) em um plano relacionado a um ponto (x_2, y_2) em outro plano (AGARWAL; JAWAHAR; NARAYANAN, 2005), como poder ser visto na Figura 3.13. Observe na ilustração que são mapeados alguns pontos da imagem da caixa correspondentes em uma cena diferente. A homografia calcula exatamente o valor da coordenada da imagem atual em outra. Sendo assim, é possível obter os valores da transformação em perspectiva. Para o caso proposto na solução deste trabalho de pesquisa, a Figura 3.12 exibe a correspondência entre os planos distintos dos 5 pontos pré-selecionados. Posteriormente, a homografia calculada é usada na aplicação da transformação em perspectiva de todos os pontos da imagem dos óculos virtuais na imagem da face, provida pela câmera de vídeo em tempo real. A Figura 3.14 exibe um exemplo desta aplicação para imagens JPG, sem plano de fundo transparente.

Ao final, é realizada a combinação das imagens, ou seja, a imagem dos óculos virtuais com fundo transparente é adicionada à imagem da câmera, onde se encontra a face. Tal combinação é conhecida como *alpha blending* (SZELISKI, 2010), que usa o valor da opacidade para definir a cor do *pixel* que será pintado na imagem de plano de fundo. Por conseguinte, os óculos virtuais são sobrepostos à face (como visto na Figura 3.11), dando a impressão de que parecem fazer parte do ambiente real, caracterizando assim uma aplicação de RA.

Vale salientar que é essencial a escolha correta dos pontos de correspondência das duas imagens no momento do calculo da homografia, para que o resultado final seja satisfatório. Com a transformação em perspectiva, as distorções por movimento da cabeça não afetam o resultado da projeção. Pelo contrário, a técnica de transformação em perspectiva realiza uma deformação

Figura 3.11: Esquematização da solução de sobreposição de um objeto virtual à face



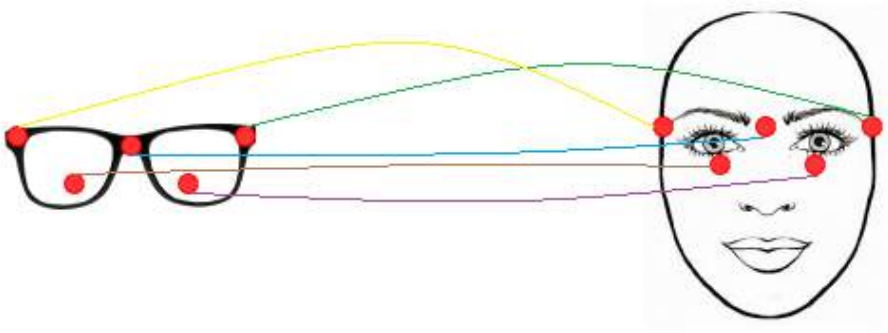
Fonte: O autor

da imagem final para que o objeto virtual mantenha-se fixado aos pontos de correspondência predefinidos, contanto que eles não sejam ocludidos.

O restante desta seção põe em prática os conceitos teóricos descritos acima. Para um entendimento preciso dos pseudocódigos, os tipos de variáveis e rotinas de C++ e do OpenCV são escritos em negrito. Por exemplo, **Mat** é uma matriz para manipulação de imagens e **imread** é um função para ler imagens no OpenCV. Em contrapartida, as referências para as palavras dos pseudocódigos são descritos em itálico, como *ImgObj*, um pseudônimo para uma variável.

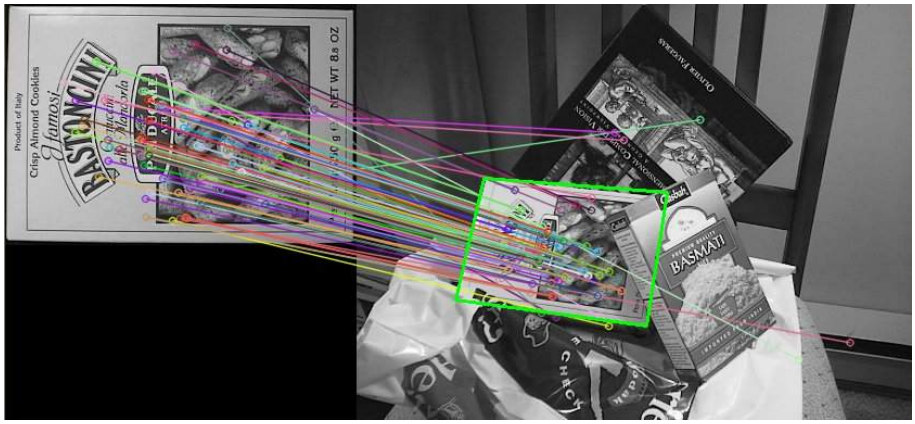
A função centralizadora, chamada *SobreporOculosNaFace*, é implementada no Pseudocódigo 1. Ela recebe como parâmetros a imagem da câmera (*ImgCam*), o diretório (endereço

Figura 3.12: Correspondência entre planos distintos



Fonte: O autor

Figura 3.13: Mapeamento de pontos entre duas imagens para o calculo da homografia



Fonte: `http://docs.opencv.org/2.4/doc/tutorials/features2d/feature_homography/feature_homography.html`

Figura 3.14: Aplicação da transformação em perspectiva usando uma homografia pré-calculada

Fonte: (GOLLA, 2014)

local) da imagem do objeto (*DirObjImg*) – especificamente neste caso uma imagem de óculos 2D – e os pontos de correspondência da face (*PtsFace*) e do objeto (*PtsObj*), respectivamente. *ImgCam* é um tipo **Mat**. *PtsFace* e *PtsObj* são vetores (**vector**) de **Mat**, e *DirObjImg* é uma **string**. O pseudocódigo segue com uma condicional para verificar se a imagem do objeto já foi lida e redimensionada. Isso ocorre porque o CLM funciona em um laço infinito, rastreando os pontos da face a partir da sequência de imagens da câmera. Como tal câmera gera diversas imagens em tempo real, a função *SobreporOculosNaFace* é chamada todas as vezes para cada imagem da sequência. Porém, a imagem do objeto é estática, e só precisa ser lida e redimensionada uma vez. Com isso, a variável *ImgObjConfig*, do tipo **bool**, é usada como uma *flag* binária, cujo resultado pode ser verdadeiro ou falso, indicando, caso seja verdadeiro, se o objeto foi lido e redimensionado, e falso, caso contrário. Essa ação é fundamental para o desempenho da função.

Pseudocódigo 1 Algoritmo de Sobreposição de Óculos Virtuais na Face

```
Procedimento SOBREPOROCULOSNAFACE(ImgCam ,DirObjImg, PtsFace, PtsObj)
    Se ImgObjConfig = falso Então
        ImgObj ← LerImgObj(DirObjImg)
    Fim
    D ← PtsFace.rows / 2
    ProjetarEmPerspectiva(ImgCam, ImgObj, PtsFace, PtsObj, D)
Fim
```

Deduzindo que a função seja chamada pela primeira vez, *ImgObj* recebe a imagem do objeto carregado pela função *LerImgObj*, que retorna uma imagem **Mat**. Tal função pode ser vista do Pseudocódigo 2, descrito a posteriori nesta subseção. Após carregar a imagem do objeto, *D* é calculado como o valor limite das linhas dos pontos de referência da face, providos pela forma da face 2D rastreada pelo CLM. *D* é uma variável do tipo **int**, importante para delimitar a área dos pontos da imagem do objeto correspondentes na imagem da câmera. Adiante, seu uso será observado no Pseudocódigo 3. Por fim, *ProjetarEmPerspectiva*, também descrito mais adiante no Pseudocódigo 3, recebe as imagens da câmera (*ImgCam*) e do objeto (*ImgObj*), os pontos da face (*PtsFace*) e do objeto (*PtsObj*), e *D*, os quais serão usados na projeção em perspectiva dos pontos de correspondência predefinidos.

Pseudocódigo 2 Algoritmo de Leitura e Redimensionamento da Imagem do Objeto

```
Procedimento LERIMAGEM(DirImg)
    ImgObj ← {}
    imread(ImgObj, DirImg, -1)
    Tam ← Size(Larg, Alt)
    resize(ImgObj, ImgObj, Tam)
    ImgObjConfig ← verdadeiro
    Retorne ImgObj
Fim
```

Então, uma vez implementado o Pseudocódigo 1, tem-se no Pseudocódigo 2 a função de carregamento da imagem do objeto (*LerImagem*). A função **imread** é responsável por ler a imagem do objeto no diretório *DirImg* e armazenar na variável *ImgObj*. Como este ultimo recebe uma imagem, ele é tipado como **Mat**. Ainda no **imread**, o parâmetro *-1* é passado como uma *flag* para indicar que o canal alfa deve ser carregado. O canal alfa é a transparência da imagem, essencial para que o plano de fundo não fique à mostra quando o objeto sobrepor a face. Em seguida, *Tam* recebe o tamanho padrão predefinido para a imagem do objeto. Vale ressaltar que a escolha desse tamanho é opcional. Contudo, para um resultado mais próximo do real, é importante redimensionar o objeto com um valor relativo à face do usuário. O tipo **Size(Larg, Alt)** define este tamanho escolhido para a imagem do objeto. Logo após, a função **resize** redefine as dimensões da imagem do objeto para os valores de largura e altura, referentes à *Larg* e *Alt*, respectivamente. Por último, *ImgObjConfig* é definido como *verdadeiro* e a matriz dos pontos da imagem do objeto é retornada.

Dado que a imagem do objeto virtual foi carregada e configurada, tem-se o *Algoritmo de Projeção em Perspectiva*, exposto no Pseudocódigo 3. Para a obtenção de um resultado satisfatório da função *ProjetarEmPerspectiva*, é estritamente importante que os pontos de correspondência predefinidos (*PtsFace* e *PtsObj*) estejam de acordo com o cenário real, como foi elucidado na concepção teórica descrita no início desta seção. Prosseguindo com o Pseudocódigo 3, inicialmente, o vetor *PtsCorresp* é declarado, do tipo **Point2f**. Este é um tipo do OpenCV para manipular *pixels* em imagens 2D – da forma *(x, y)* – com valor de ponto flutuante. No momento seguinte, para cada ponto do vetor de pontos de correspondência da face (*PtsFace*), a variável auxiliar PO_i recebe ponto *(x, y)* do objeto *PtsObj*, no índice P_i, referente ao ponto de correspondência do objeto virtual e, logo após, *PtsRef*, no índice P_i, recebe **Point2f** (x_1, y_1), correspondente à face. Neste momento, D é usado para definir a coluna y_1. Tendo os pontos de referencia facial (*PtsFace*) e os pontos do objeto correspondentes na face (*PtsCorresp*), a homografia é calculada. A função **findHomography** procura e retorna a transformação em perspectiva entre os planos de origem e destino (BRADSKI; KAEHLER, 2008) (ITSEEZ, 2000b). Observe a expressão abaixo:

$$s_i = \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \sim H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

Ela exibe o calculo de homografia para uma matriz S_i *3x3*, de modo a que o erro de retroprojeção seja minimizado (ITSEEZ, 2000b):

$$\sum_i \left(x_i' - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 + \left(y_i' - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2$$

Se o terceiro parâmetro da função, relativo ao modo de computação da matriz de homografia, é definido como o valor padrão 0, a função usa todos os pares de pontos para calcular uma estimativa de homografia inicial com um simples esquema de mínimos quadrados (ITSEEZ, 2000b).

Em posse da homografia, já é possível aplicar a projeção em perspectiva entre a imagem da câmera, onde se encontra a face do usuário, e o objeto virtual. Antecedente a isto, é necessário armazenar o tamanho da imagem da câmera (*Tam*), o qual será usado como parâmetro para a aplicação da transformação em perspectiva, e realizar uma cópia da mesma, evitando uma futura sobrescrita da imagem da câmera original (*CopImgCam*). A função **warpPerspective** é responsável por aplicar tal transformação na imagem da câmera (BRADSKI; KAEHLER, 2008) (ITSEEZ, 2000c). A mesma usa a matriz especificada na expressão seguinte para transformar a imagem de origem:

$$dst\,(x,y) = \left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}} \right)$$

Como entrada para esta função tem-se *ImgObj* como imagem de origem, *CopImgCam* como a imagem de destino, *Homografia*, calculada anteriormente, e *Tam*. Ao final, **warpPerspective** aplica a transformação da imagem do objeto (*ImgObj*) na imagem da câmera (*CopImgCam*), usando a homografia (*Homografia*) e o tamanho *Tam* da imagem de destino (*CopImgCam*).

Finalmente, a função *CombinarImagens*, do Pseudocódigo 4, é invocada. Para sobrepor a imagem dos óculos na face, foi usada a implementação de Michael Jepson (JEPSON, 2012), derivada do algoritmo de combinação de imagens do Opencv (ITSEEZ, 2012). A função inicia-se armazenando na variável *PtIni* o ponto inicial **Point**(0, 0) da primeira linha da imagem de plano de fundo, neste caso, da imagem da câmera. O uso desta variável será descrito aqui posteriormente. Em seguida, são armazenados nas variáveis *LinsImgCam* e *ColsImgCam* a quantidade de linhas e colunas da imagem transformada da câmera, respectivamente. Em seguida, também são armazenadas as quantidades de linhas (*LinsCopImgCam*) e colunas (*ColsCopImgCam*) da imagem original. Vale lembrar que a imagem da câmera original copiada (*CopImgCam*) permanece inalterada, enquanto que a imagem armazenada em *ImgCam* possui a transformação realizada pelo Pseudocódigo 3.

L recebe o valor máximo entre *PtIni* e *0*. Essa variável é usada para indicar se o ponto da posição atual, na varredura da matriz da imagem copiada, é a da linha superior do plano de fundo (L_i = *0*) ou da linha superior do primeiro plano (L_i = *PtIni*[L_i]). Este é armazenado em *PY* em cada fase da varredura nas linhas da imagem de primeiro plano. O mesmo ocorre para as colunas da imagem, em que *C* recebe o valor máximo **max**(*PtIni.x*, *0*), com *PX* recebendo a posição atual da coluna da imagem (C_j - *Ptini*[C_j]).

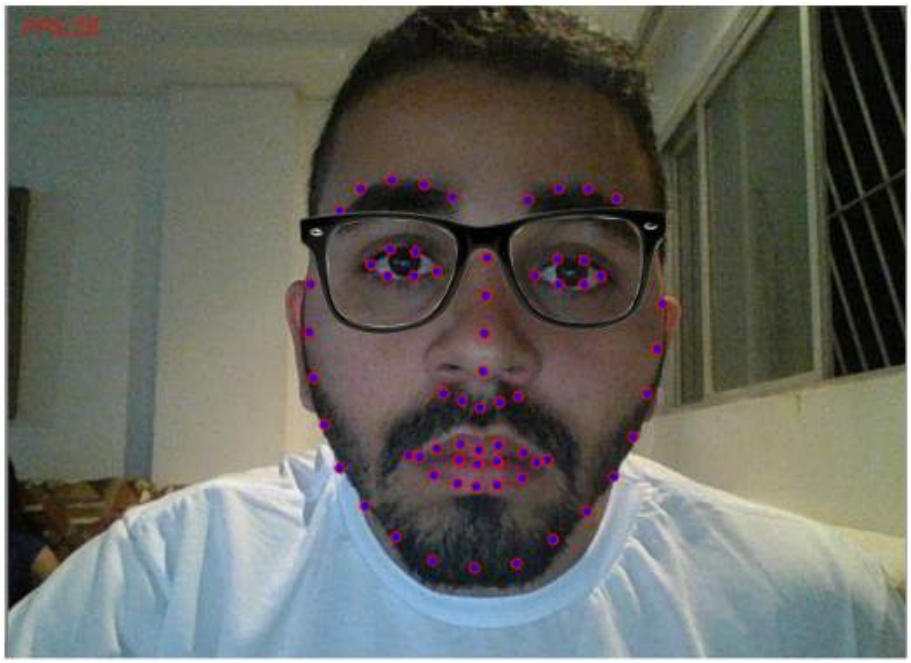
Com as coordenadas atuais da imagem do primeiro plano (*PY* e *PX*) em mãos, *Op* recebe o calculo da opacidade deste ponto. A opacidade é o nível de ausência de transparência em uma

Pseudocódigo 3 Algoritmo de Projeção em Perspectiva

```
Procedimento PROJETAREMPERSPECTIVA(ImgCam, ImgObj, PtsFace, PtsObj, D)
    PtsRef ← {}
    Para P ∈ PtsFace Faça
        PO_i ← PtsObj[P_i]
        PtsCorresp[P_i] ← Point2f(PtsFace[PO_i], PtsFace[PO_i + D])
    Fim
    Homografia ← findHomography(PtsFace, PtsCorresp)
    Tam ← ImgCam.size()
    CopImgCam ← ImgCam
    warpPerspective(ImgObj, CopImgCam, Homografia, Tam)
    CombinarImagens(ImgCam, CopImgCam)
Fim
```

imagem. Este define se o *pixel* da coordenada atual é transparente ou não, usando o canal *alpha* do sistema de cores RGBA da imagem de primeiro plano (*CopImgCam*). Sendo $Op > 0$, então o ponto não é transparente, Caso contrário, ele possui transparência. Posteriormente, para cada canal do RGB do pixel atual da imagem de primeiro plano, o pixel da imagem de plano de fundo (*ImgCam*) é pintado com o valor provido de Op. Por fim, a saída da função retorna *ImgCam*, a imagem da câmera com os *pixels* do objeto virtual pintados sobre a mesma, realizando assim o aumento da face, dando a sensação visual de que os óculos virtuais estejam atrelado ao rosto do usuário na câmera (Figura 3.15).

Figura 3.15: Resultado da solução de aumento de faces



Fonte: O Autor

Pseudocódigo 4 Algoritmo de Combinação de Imagens (Alpha Blending)

```
Procedimento COMBINARIMAGENS(ImgCam, CopImgCam)
    PtIni ← Point(0, 0)
    LinsImgCam ← ImgCam.rows
    ColsImgCam ← ImgCam.cols
    LinsCopImgCam ← CopImgCam.rows
    ColsCopImgCam ← CopImgCam.cols
    L ← max(PtIni.y, 0)
    Para L...LinsCopImgCam Faça
        PY ← L_i - PtIni[L_i]
        Se PY >= LinsImgCam Então
            Fim
        Fim
        C ← max(PtIni.x, 0)
        Para C...ColsCopImgCam Faça
            PX ← C_j - PtIni[C_j]
            Se PX >= ColsImgCam Então
                Fim
            Fim
            Op ← ImgCam.data[PY * ImgCam.step + PX * ImgCam.channels() + 3] / 255
            K ← 0
            Para K ... Op > 0 e K > CopImgCam.channels() Faça
                A_1 ← PY * ImgCam.step
                A_2 ← PX * ImgCam.channels()
                FPlan ← ImgCam.data[A_1 + A_2 + K]
                B_1 ← L * CopImgCam.step
                B_2 ← C * CopImgCam.channels()
                BPlan ← CopImgCam.data[ B_1 + B_2 + C]
                Q_1 ← L * CopImgCam.step
                Q_2 ← CopImgCam.channels()
                Q_3 ← BPlan * (1 - Op) + FPlan * Op
                CopImgCam.data[Q_1 + Q_2 * C + K] = Q_3
            Fim
        Fim
        Retorne ImgCam
    Fim
Fim
```

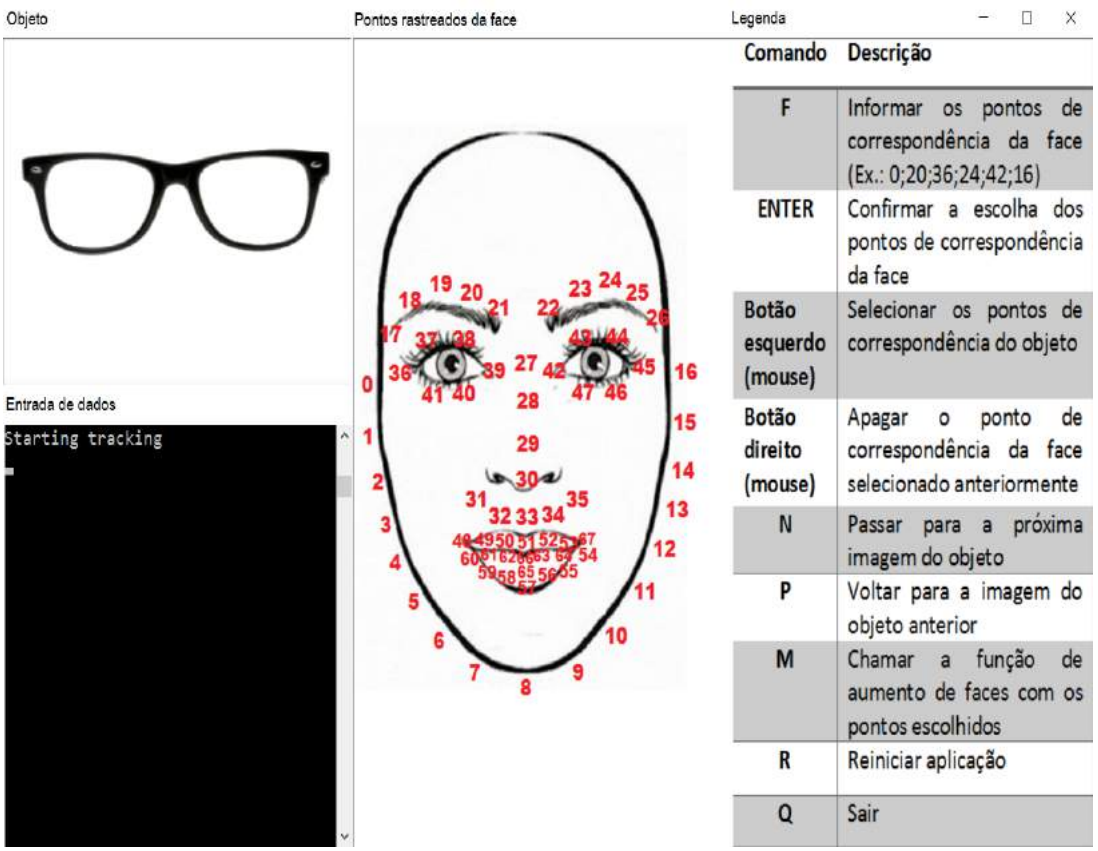
Foi visto nesta seção o detalhamento da implementação da solução de aumento de faces. Apesar de obter um resultado parcial satisfatório, ilustrado na Figura 3.15, foi necessário desenvolver uma aplicação que utilizasse as funções propostas, com o intuito de demonstrar como a solução pode contribuir para a área de RA. Tal aplicação é apresentada na próxima seção.

3.3 Aplicação de Autoria

Partindo da solução de Realidade Aumentada Facial, proposta no Capítulo 3, foi implementada um aplicação *desktop* com o intuito de realizar experimentos para a validação da mesma e exposição dos resultados. A aplicação foi desenvolvida em C++ e OpenCV, no ambiente de desenvolvimento MVS 2012, assim como as funções da solução. Vale ressaltar que a interface gráfica do sistema não é relevante para os objetivos deste trabalho, pois a meta principal é demonstrar uma maneira de informar os pontos de correspondência da face e do objeto e realizar uma análise de viabilidade das funções, a partir de análises de qualidade e desempenho, para a construção de aplicações futuras. Esta seção mostra o funcionamento do sistema de autoria e no Capítulo 4 serão mostrados os resultados e as análises feitas.

O ponto de partida da aplicação é a tela inicial destacada na Figura 3.16. A mesma aparece apenas quando a câmera rastreia a face do usuário. Ela é formada pelos seguintes componentes: a imagem do objeto; um terminal de comandos, o qual o usuário está apto à informar dados pelo teclado; uma imagem da face, que contém os índices dos pontos providos pelo algoritmo CLM mapeados na face (de 0 à 67); e a legenda, com os comando que o usuário poderá inserir na aplicação.

Figura 3.16: Tela inicial da aplicação de aumento de faces



Fonte: O Autor

Foram utilizadas 17 imagens de óculos virtuais 2D, com extensão PNG e plano de fundo transparente, como expõe a Figura 3.17. É possível notar que há uma variação de formas, cores e tamanhos. Para realizar a escolha das imagens o usuário poderá pressionar as teclas *N* e *P*. Pressionando *N* a próxima imagem será selecionada. Em contrapartida, pressionando a tecla *P*, a imagem do objeto anterior é selecionada. Dessa forma, o usuário é capaz de escolher quaisquer dos 17 óculos. A Figura 3.18 exibe um exemplo quando o usuário pressiona *N* em um primeiro momento. Observe que os óculos virtuais anteriores, vistos na Figura 3.16, foram trocado.

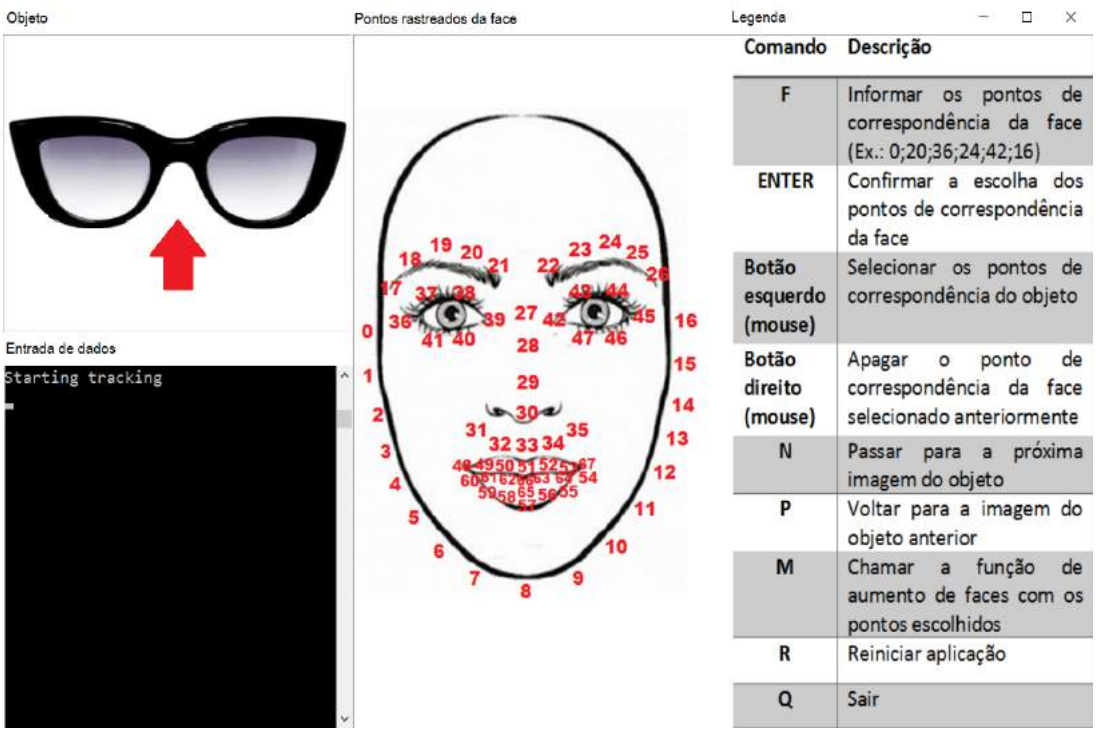
Figura 3.17: Óculos virtuais 2D utilizados na aplicação de aumento de faces

Fonte: O Autor

É opcional fazer a escolha dos óculos em primeira instância. Antes, o usuário pode informar os pontos de correspondência da face. Ao pressionar a tecla *F*, o *prompt* de comando pede que o usuário informe os índices de tais pontos (Figura 3.19). Observe na figura que a seguinte mensagem é exibida no *prompt*: "*Informe os pontos de correspondência da imagem da face*". Então, o usuário deve selecionar a janela do *prompt* e digitar os valores dos índices dos pontos de referência da imagem da face, separados por "*;*" (Ex.: 0;10;18;27;28;24;16;2). Neste momento, é fundamental basear-se no desenho da face com os índices dos pontos de referência providos pelo CLM. Informado os pontos, o usuário deve pressionar "*Enter*" para confirmar. Logo, a mensagem "*Pontos de correspondência da imagem da face selecionados...*" é exibida. Toda esta ação pode ser vista na Figura 3.20.

Em posse dos pontos de correspondência da face, o usuário deve informar agora os pontos de correspondência dos óculos. Para tal, é preciso usar o *botão esquerdo do mouse*. O

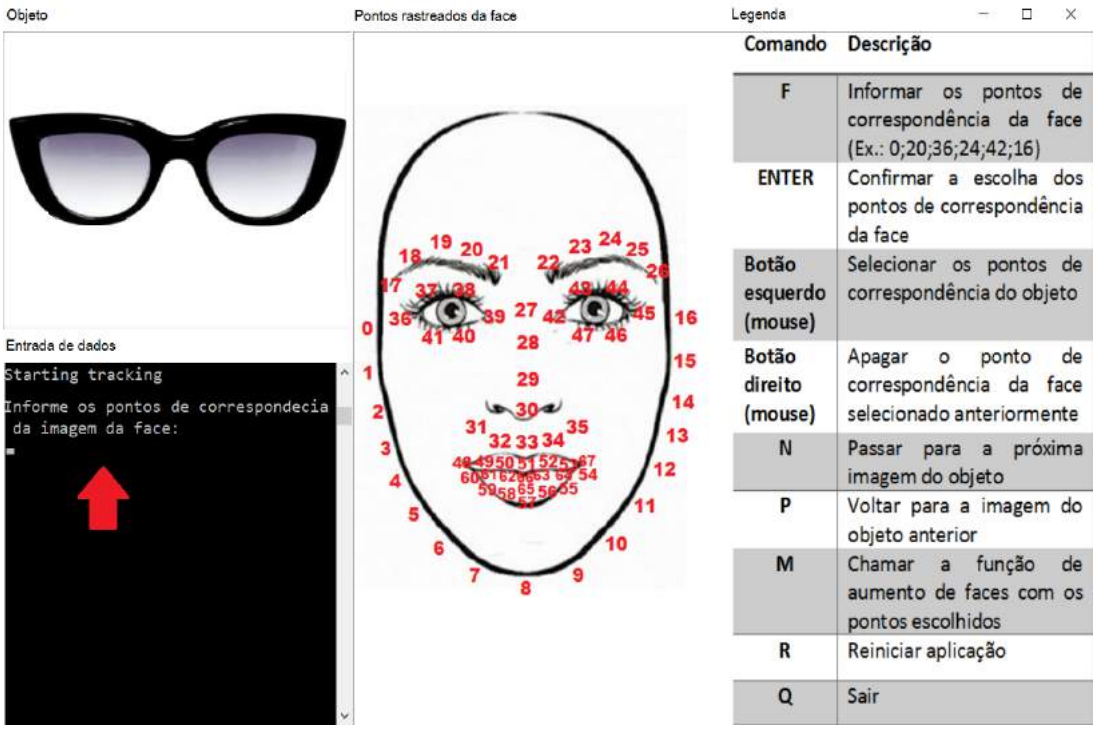
Figura 3.18: Óculos virtuais trocados após pressionar *N*



Comando	Descrição
F	Informar os pontos de correspondência da face (Ex.: 0;20;36;24;42;16)
ENTER	Confirmar a escolha dos pontos de correspondência da face
Botão esquerdo (mouse)	Selecionar os pontos de correspondência do objeto
Botão direito (mouse)	Apagar o ponto de correspondência da face selecionado anteriormente
N	Passar para a próxima imagem do objeto
P	Voltar para a imagem do objeto anterior
M	Chamar a função de aumento de faces com os pontos escolhidos
R	Reiniciar aplicação
Q	Sair

Fonte: O Autor

Figura 3.19: Aplicação requerendo os pontos de correspondência da face após o usuário pressionar *F*

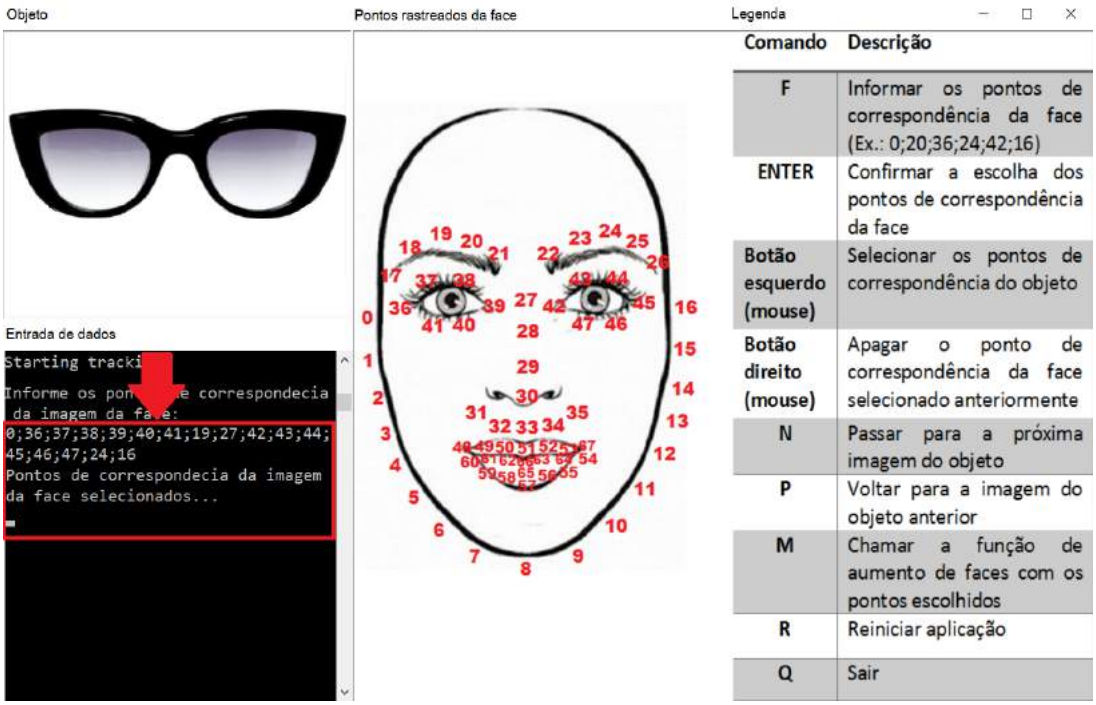


Comando	Descrição
F	Informar os pontos de correspondência da face (Ex.: 0;20;36;24;42;16)
ENTER	Confirmar a escolha dos pontos de correspondência da face
Botão esquerdo (mouse)	Selecionar os pontos de correspondência do objeto
Botão direito (mouse)	Apagar o ponto de correspondência da face selecionado anteriormente
N	Passar para a próxima imagem do objeto
P	Voltar para a imagem do objeto anterior
M	Chamar a função de aumento de faces com os pontos escolhidos
R	Reiniciar aplicação
Q	Sair

Fonte: O Autor

usuário está habilitado a mover o *mouse* e pressionar o botão esquerdo nas coordenadas dos óculos virtuais que sejam de seu interesse. Vale lembrar que o valor máximo de pontos que podem ser selecionados, tanto da face quanto dos óculos, é 68. Contudo, visto que outrora os pontos da face foram selecionados, o usuário deverá escolher exatamente a mesma quantidade para o objeto. Ao selecionar as coordenadas dos óculos, pontos em vermelho serão desenhados sobre ela, com o intuito de manter à mostra as coordenadas indicadas (Figura 3.21).

Figura 3.20: Aplicação informando que os pontos de correspondência da face foram aceitos

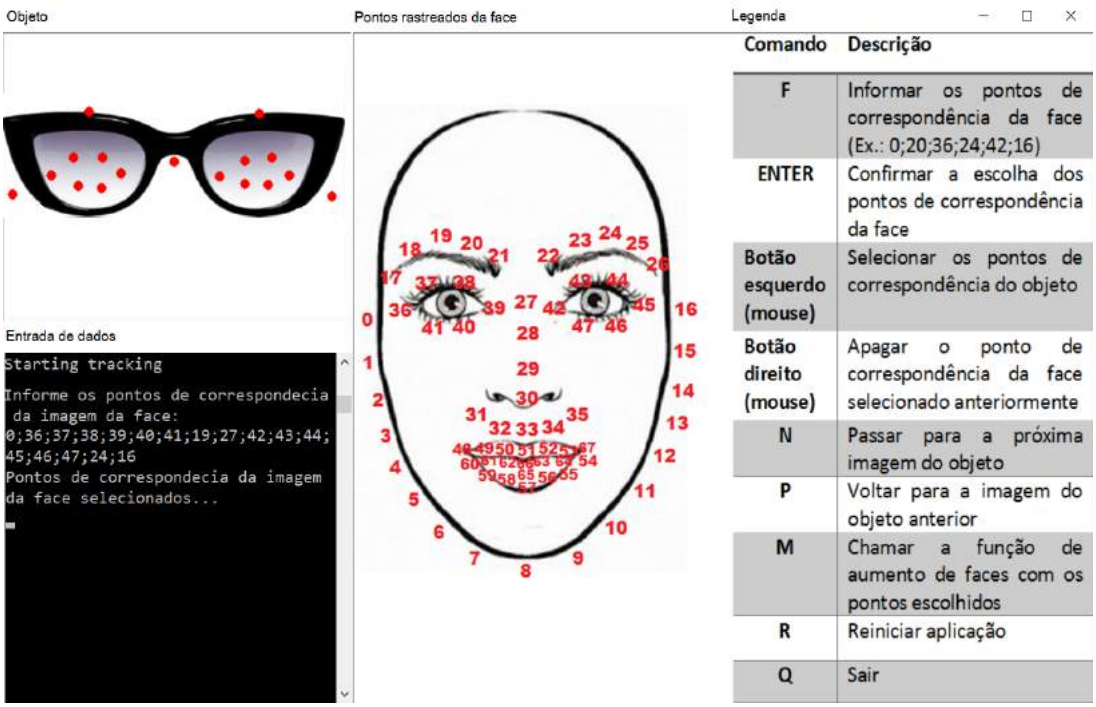


Fonte: O Autor

Observe nesta ilustração que foram escolhidos 17 pontos de correspondência em ambas as imagens. Caso o usuário pinte uma área da imagem do objeto mas opte por modificá-la, basta pressionar o *botão direito do mouse*. Com isso, o último ponto escolhido é apagado. Executando esta ação diversas vezes, os pontos serão apagados, um a um, descartando sempre o último ponto escolhido, até que não haja mais pontos e a imagem retorne para o estado original.

Por fim, com os pontos de correspondência da face e do óculos definidos, o usuário deve pressionar a tecla *M* para chamar a rotina centralizadora de aumento de faces, vista no Capítulo 3. Desse modo, os óculos virtuais são projetados na face capturada pela câmera de vídeo do computador. A posição dos óculos virtuais, no ambiente real, é relativa aos pontos dos mesmos, correspondentes aos pontos escolhidos previamente (Figuras 3.20 e 3.21). Caso haja movimentos e mudanças na orientação da cabeça, os pontos de referência da face variam em perspectiva. Atrelados a alguns desses pontos, há também uma variação dos pontos de correspondência dos óculos virtuais, pelo mapeamento realizado na projeção em perspectiva.

Figura 3.21: Pontos de correspondência do óculos providos pelo uso do *mouse*



Comando	Descrição
F	Informar os pontos de correspondência da face (Ex.: 0;20;36;24;42;16)
ENTER	Confirmar a escolha dos pontos de correspondência da face
Botão esquerdo (mouse)	Selecionar os pontos de correspondência do objeto
Botão direito (mouse)	Apagar o ponto de correspondência da face selecionado anteriormente
N	Passar para a próxima imagem do objeto
P	Voltar para a imagem do objeto anterior
M	Chamar a função de aumento de faces com os pontos escolhidos
R	Reiniciar aplicação
Q	Sair

Fonte: O Autor

Todos os resultados, provenientes da sobreposição dos 17 óculos virtuais, ilustrados na Figura 3.17, podem ser vistos no próximo capítulo.

3.4 Discussão

Neste capítulo foi apresentado e detalhado o desenvolvimento da solução de aumento de faces, que possibilita a inserção de objetos virtuais na face do usuário, sem o uso de marcadores fiduciais planares, assim como sem a necessidade da utilização de rotinas de bibliotecas de RA, por meio do algoritmo de rastreamento de face CLM.

As funções desenvolvidas permitem sobrepor qualquer acessório virtual 2D na face, como óculos (objeto usado como exemplo durante todo este capítulo), brincos, maquiagem (batom, rímel, sombras, etc.), máscaras e etc. Para isso, basta selecionar os pontos de correspondência da face e do objeto virtual e aplicar o Pseudocódigo 1, que consequentemente realiza a leitura e configuração da imagem do objeto (Pseudocódigo 2), a projeção em perspectiva (Pseudocódigo 3) e a sobreposição das imagens (Pseudocódigo 4). Nesta primeira versão da solução, algumas dificuldades foram encontradas. A primeira delas está relacionada à escolha dos pontos de correspondência. Esta é realizada manualmente, ou seja, o usuário necessita selecionar as coordenadas do objeto que serão mapeadas com alguns dos 68 pontos da face. Sendo assim, é preciso abrir a imagem do objeto em um editor de imagens e escolher as coordenadas, tornando o trabalho um tanto árduo. Todavia, com a escolha correta dos pontos de correspondência e do tamanho da imagem, os resultados parciais mostram-se satisfatórios (Figura 3.15).

A aplicação de autoria foi desenvolvida para auxiliar a escolha dos pontos de correspondência. Sendo assim, o usuário é capaz de escolhê-los de uma maneira mais ágil, por meio do uso de *mouse* e teclado. Contudo, não é possível selecionar os pontos da face por meio do *mouse*, pois a face é exibida na câmera e esta exibe imagens dinamicamente. Mesmo que os índices dos pontos de referência faciais providos pelo CLM não mudem, suas posições em relação ao ambiente real são constantemente modificadas. Assim, caso fosse solicitado que o usuário use o *mouse*, ele teria que clicar exatamente no ponto de referência da face, o que poderia demandar um pouco mais de esforço e coordenação motora.

No próximo capítulo são apresentados os resultados obtidos da aplicação de autoria. Além disso, é realizada uma análise qualitativa dos mesmos, ressaltando os pontos positivos e negativos da solução proposta. Também é realizada uma análise de desempenho, a qual apresenta a média da taxa quadros por segundo (FPS) da execução das funções, em um intervalo de tempo, e o cálculo de desvio padrão. Todas as análises foram feitas em tempo de execução da aplicação desenvolvida.

4

Resultados

As imagens abaixo exibem as sobreposições dos óculos virtuais resultantes da aplicação de autoria vista do capítulo anterior.

Figura 4.1: Resultado 1

Fonte: O Autor

Figura 4.2: Resultado 2

Fonte: O Autor

Figura 4.3: Resultado 3

Fonte: O Autor

Figura 4.4: Resultado 4

Fonte: O Autor

Figura 4.5: Resultado 5

Fonte: O Autor

Figura 4.6: Resultado 6

Fonte: O Autor

Figura 4.7: Resultado 7

Fonte: O Autor

Figura 4.8: Resultado 8

Fonte: O Autor

Figura 4.9: Resultado 9

Fonte: O Autor

Figura 4.10: Resultado 10

Fonte: O Autor

Figura 4.11: Resultado 11

Fonte: O Autor

Figura 4.12: Resultado 12

Fonte: O Autor

Figura 4.13: Resultado 13

Fonte: O Autor

Figura 4.14: Resultado 14

Fonte: O Autor

Figura 4.15: Resultado 15

Fonte: O Autor

Figura 4.16: Resultado 16

Fonte: O Autor

Figura 4.17: Resultado 17

Fonte: O Autor

4.1 Análises

Para verificar a viabilidade da solução proposta foram feitos dois tipos de análise: desempenho e qualidade. A primeira tem como objetivo medir a performance da execução das funções. A segunda mede o nível de qualidade do aumento de faces nas imagens resultantes. Ambas foram feitas sobre a aplicação apresentada no capítulo anterior.

4.1.1 Análise de desempenho

Como medida de desempenho foram feitos cálculos a partir da taxa de quadros por segundo (FPS). Essa medida é chamada de cadência, que significa o número de imagens, por uma unidade de tempo, que um dispositivo audiovisual (câmera, por exemplo) consegue registrar, processar ou exibir. Vale constar que tal análise foi executada em um notebook com as seguintes especificações:

- Sistema Operacional Windows 10 Pro;
- Processador Intel Core i7-4510U, com 4 núcleos de CPU de 2.00 GHz e 2.60 GHz;
- Memória RAM de 16 GB DDR3;
- Placa de Vídeo Intel HD Graphics;
- Memória de Disco (HD) de 1 TB;

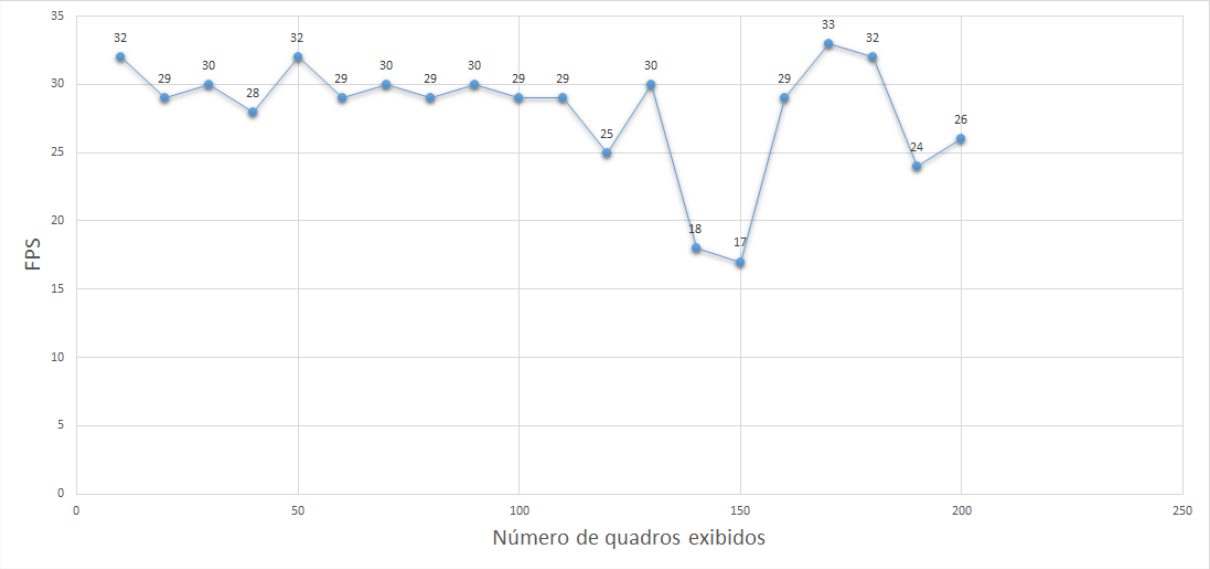
É importante frisar que a performance depende diretamente das especificações do sistema. Outro ponto importante foi a utilização de modo *Release* de compilação, junto ao MVS. Em um primeiro momento, foi extraído o FPS médio a cada 10 quadros exibidos. Observe no gráfico da Figura 4.18. É possível ver a variação do FPS até serem processados 200 quadros. O pico, ou a maior taxa registrada, ocorre no momento em que a função de aumento de faces alcança 170 quadros. A base – menor taxa registrada – é 17 e ocorre quando 150 quadros foram processados.

Também foi realizado o cálculo do desvio padrão, que significa a variação (ou dispersão) existente em relação a uma média. Este teve como objetivo calcular o grau de oscilação em relação à média geral do FPS, até os 200 quadros registrados. Quanto menor o valor do desvio padrão, menor a taxa de dispersão, e, consequentemente, melhor a consistência do processamento. A média da amostra foi de **28,05** quadros por segundo e o desvio padrão, baseado nos desvios absolutos dos valores de FPS a cada 10 quadros registrados, foi de aproximadamente **3,03**. Tendo em vista que o menor valor para o desvio padrão é 0, a variação do FPS é satisfatória.

4.1.2 Análise de qualidade

Tendo como base os resultados do aumento de faces mostrados no Capítulo 3, foi possível levantar alguns pontos em relação a qualidade do alinhamento entre o virtual e o real. Vale

Figura 4.18: Resultado do valor médio de quadros por segundos a cada 10 quadros exibidos



Fonte: O Autor

ressaltar que tal qualidade depende diretamente dos pontos de correspondências previamente selecionados e das imagens, que variam suas características (Figura 3.17). Com isso, a avaliação foi feita a partir de 17 pontos de referência da face. Seus índices são: **0, 36, 37, 38, 39, 40, 41, 19, 27, 42, 43, 44, 45, 46, 47, 24, 16**. Tais pontos foram os mesmos definidos nos resultados da aplicação de aumento de faces do capítulo anterior. Sendo assim, para esta avaliação, foram analisadas as imagens ilustradas nas figuras 4.1 à 4.17.

Como pontos positivos alcançados temos:

- A qualidade do alinhamento quando o usuário se posiciona com a face frontal à câmera é ótima em 100% dos resultados;
- No movimento lateral da cabeça, a sobreposição dos óculos também possui um alto grau de adequação;
- No movimento de rotação da cabeça, os resultados do alinhamento mostram-se satisfatórios, com falhas mínimas em algumas imagens (como nas figuras 4.1, 4.7 e 4.8).

Como pontos negativos observados:

- O aumento de faces no movimento de extensão da cabeça foi o que obteve um maior número de falhas. Mesmo que o nível de alinhamento seja ótimo, os óculos virtuais não condizem, na maioria dos casos, com as dimensões pré-definidas quando ocorre este movimento (como ilustrado nas figuras 4.4, 4.5 e 4.9).

- No movimento de flexão da cabeça, algumas imagens obtiveram níveis de satisfação mais abaixo do esperado (como nas figuras 4.2 e 4.7).

Em suma, as análises realizadas nesta seção viabilizam a solução de aumento de faces em um primeiro momento. A performance alcançou uma taxa de quadros por segundo satisfatória, e não houve nenhuma interrupção ou atraso perceptível. O alinhamento entre o real e o virtual também obteve resultados satisfatórios, mesmo com os pontos negativos citados acima. Falhas ocorreram pelo fato que os pontos de correspondência predefinidos variam demasiadamente, viso que o usuário deve escolhe-los manualmente. Ainda sim, nas posições frontais e laterais, as quais são as posições mais adequadas para visualizar os resultados, a sobreposição dos óculos virtuais 2D é robusta.

4.2 Discussão

Neste capítulo foi possível observar a aplicabilidade da solução de aumento de faces proposta. Partindo dos resultados, foram feitas avaliações, como forma de realizar uma validação para as funções da solução proposta.

O principal problema encontrado foi a escolha manual dos pontos de correspondência dos objetos virtuais, como foi abordado no Capítulo 3. Derivado disto, alguns resultados, em movimentos específicos da cabeça, mostraram algumas inconsistências no alinhamento entre o virtual e o real. Porém, em suma, a qualidade do alinhamento foi eficaz, visto que são necessárias pequenas correções. Quanto a performance, por meio da análise de desempenho exposta na Seção 4.1.1, as funções de aumento de faces possuem um desempenho eficaz para o uso em aplicações de RA em um sistema com especificações adequadas para o processamento de vídeos em tempo real.

Conclui-se que a viabilidade para a aplicação da solução em bibliotecas de visão computacional ou Realidade Aumentada é apropriada, visto que nas pesquisas feitas para os fins deste trabalho, não é conhecida nenhuma rotina de código aberto com o intuito de aumentar faces, sem o uso de marcadores físicos.

5

Conclusão

Este documento apresentou uma solução *opensource*, que contribui no desenvolvimento de sistemas de RA que propõem inserir objetos virtuais na face do usuário, sem o uso prévio de marcadores. Como foi visto no Capítulo 2, existe um grande número de aplicações nesse contexto, as quais são atuantes no mercado tecnológico de aplicações *web*, *desktop* e dispositivos móveis. Entretanto, os desenvolvedores não compartilham as soluções implementadas, muito menos em formato de biblioteca para o uso aberto.

As bibliotecas de RA estudadas para este trabalho fornecem apenas recursos básicos em extensões livres, não contendo rotinas com o propósito de aumentar faces a partir de características naturais da cena, necessitando de marcadores físicos para este fim. Outro problema relacionado à tais bibliotecas são as extensões comerciais pagas. O alto valor cobrado por elas é um empecilho para os desenvolvedores, que muitas vezes não se encontram em condições de adquiri-las pelo fator econômico. Provavelmente, estas extensões contém funções de aumento de faces mais avançadas, que poderiam resultar numa solução como a proposta por este trabalho.

Todavia, como alternativa para esse empecilho, foram listadas algumas bibliotecas de visão computacional (OpenCV, CCV e SimpleCV), *opensource*, amplamente utilizadas também na área de RA. Mesmo assim, estas também não implementam rotinas prontas de aumento de faces, existindo a necessidade de criar algoritmos por meio de suas rotinas para ser possível chegar a esse propósito. Posto isto, usando algumas rotinas do OpenCV (BRADSKI; KAEHLER, 2008) (ITSEEZ, 2000a), foi possível implementar a solução de aumento de faces descrita durante todo este documento.

Uma validação foi feita, por meio de uma aplicação *desktop*, com o objetivo de avaliar a solução, mediante análises de qualidade e desempenho. Apesar do aumento de faces apresentar alguns erros – providos principalmente da escolha incorreta dos pontos pelo usuário –, pelo curto espaço de tempo para concluir este projeto, o nível de qualidade da sobreposição de imagens foi satisfatória. O desempenho das funções atingiu um nível de FPS também satisfatório.

A implementação das funções apresentadas nos pseudocódigos descritos no Capítulo 3, juntamente com a aplicação de autoria exposta no Capítulo 4, encontram-se no repositório GitHub em (LUCENA, 2016).

5.1 Trabalhos Futuros

Este trabalho encerra-se aqui com uma explanação do que pode ser desenvolvido futuramente, como incremento da solução proposta.

Foi observado que a principal dificuldade encontrada na execução das funções propostas é a escolha dos pontos de correspondência entre planos distintos, a qual é realizada de forma manual. Então, um dos trabalhos futuros seria automatizar a escolha dos pontos de correspondência entre as imagens, que poderiam ser escolhidos dinamicamente, conforme o formato da imagem do objeto, como ocorre em algumas aplicações exibidas neste trabalho (OPTICIENS, 2015) (INC, 2015) (LIVET; YANG, 2015). Dessa maneira, os erros mostrados na análise qualitativa poderiam ser minimizados.

Outra contribuição futura seria adaptar as funções propostas para realizar a sobreposição de imagens com objetos 3D. Com isso, seria possível alcançar resultados mais próximos do real, como na aplicação XZIMG Augmented Face (LIVET; YANG, 2015). Para tal, será necessário o uso de sensores RGB-D, como o *Kinect*, para obter coordenadas 3D da face do usuário, o que propiciaria uma maior precisão para o aumento de faces.

Também há a pretensão de contribuir disponibilizando o código da solução para que este possa fazer parte da biblioteca do OpenCV.

Outro ponto que fica pra trabalhos futuros é a implementação da solução em outras plataformas, como dispositivos móveis e *web*, uma vez que as funções foram implementadas apenas para plataformas desktop.

Referências

AGARWAL, A.; JAWAHAR, C.; NARAYANAN, P. A survey of planar homography estimation techniques. **Centre for Visual Information Technology, Tech. Rep. IIIT/TR/2005/12**, [S.l.], 2005.

AZUMA, R. T. A survey of augmented reality. **Presence: Teleoperators and virtual environments**, [S.l.], v.6, n.4, p.355–385, 1997.

BALTRUŠAITIS, T.; ROBINSON, P.; MORENCY, L.-P. 3D constrained local model for rigid and non-rigid facial tracking. In: COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2012 IEEE CONFERENCE ON. **Anais...** [S.l.: s.n.], 2012. p.2610–2617.

BRADSKI, G.; KAEHLER, A. **Learning OpenCV**: computer vision with the opencv library. [S.l.]: "O'Reilly Media, Inc.", 2008.

CRISTINACCE, D.; COOTES, T. F. Feature Detection and Tracking with Constrained Local Models. In: BMVC. **Anais...** [S.l.: s.n.], 2006. v.2, n.5, p.6.

FANELLI, G. et al. Real time head pose estimation from consumer depth cameras. In: **Pattern Recognition**. [S.l.]: Springer, 2011. p.101–110.

FRUHINSHOLZ, A.; BERGER, V.; ELZIERE, T. **Social Compare: Augmented Reality SDK Comparison**, 2013. Disponível em: <http://socialcompare.com/en/comparison/augmented-reality-sdks>. Acesso em: 12 mar. 2016.

GAO, C.; LU, S.-L. Novel FPGA based Haar classifier face detection algorithm acceleration. In: FIELD PROGRAMMABLE LOGIC AND APPLICATIONS, 2008. FPL 2008. INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2008. p.373–378.

GOLLA, R. **Perspective projection with Homography – OpenCV**, 2014. Disponível em: <https://ramsrigoutham.com/2014/06/14/perspective-projection-with-homography-opencv/>. Acesso em: 05 mar. 2016.

HARTLEY, R.; ZISSERMAN, A. **Multiple view geometry in computer vision**. [S.l.]: Cambridge university press, 2003.

INC, M. T. **MSQRD**, 2015. Disponível em: <http://msqrd.me/>. Acesso em: 22 fev. 2016.

INC, S. M. **SimpleCV**, 2006. Disponível em: <http://simplecv.org/>. Acesso em: 30 fev. 2016.

INTERACTIVE, P. **Transformers Augmented Reality**, 2005. Disponível em: <https://www.youtube.com/watch?v=pzB4mIPdm9k>. Acesso em: 19 fev. 2016.

INTERFACES, T. M. U. **TAT Augmented ID**, 2005. Disponível em: <https://www.youtube.com/watch?v=tb0pMeg1UN0>. Acesso em: 18 fev. 2016.

ITSEEZ. **Open Source Computer Vision - OpenCV**, 2000. Disponível em: <http://opencv.org>. Acesso em: 15 fev. 2016.

ITSEEZ. **Opencv. Find Homography**, 2000. Disponível em: <http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html?highlight=findhomography>. Acesso em: 06 mar. 2016.

ITSEEZ. **Opencv. Warp Perspective**, 2000. Disponível em: <http://docs.opencv.org/3.0-beta/modules/imgproc/doc/geometric_transformations.html#warpperspective>. Acesso em: 06 mar. 2016.

ITSEEZ. **Adding (blending) two images using OpenCV**, 2012. Disponível em: <http://docs.opencv.org/2.4/doc/tutorials/core/adding_images/adding_images.html>. Acesso em: 15 mar. 2012.

JEPSON, M. **Overlay transparent image in OpenCV**, 2012. Disponível em: <http://jepsonsblog.blogspot.com.br/2012/10/overlay-transparent-image-in-opencv.html>. Acesso em: 10 mar. 2012.

KALAL, Z.; MIKOLAJCZYK, K.; MATAS, J. Face-tld: tracking-learning-detection applied to faces. In: IMAGE PROCESSING (ICIP), 2010 17TH IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2010. p.3789–3792.

KATO, H.; BILLINGHURST, M. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In: AUGMENTED REALITY, 1999.(IWAR'99) PROCEEDINGS. 2ND IEEE AND ACM INTERNATIONAL WORKSHOP ON. **Anais...** [S.l.: s.n.], 1999. p.85–94.

LIVET, N.; YANG, X. **XZIMG Augmented Face**, 2015. Disponível em: <http://www.xzimg.com/>. Acesso em: 10 mar. 2016.

LTD, N. M. **Selfit – Mixed Reality Photo and Video Selfie Creator**, 2005. Disponível em: <https://itunes.apple.com/es/app/selfit-mixed-reality-photo/id916830852?l=en&mt=8>. Acesso em: 19 fev. 2016.

LUCENA, J. **Aplicação de Aumento de Faces com CLM**, 2016. Disponível em: <https://github.com/joaootavio93/Aplica-o-de-Aumento-de-Faces-com-CLM>.

MARIN, E. **3D Face Tracking**, 2005. Disponível em: <https://www.youtube.com/watch?v=i_bZNVmhJ2o>. Acesso em: 22 fev. 2016.

MCNEELY, W. A.; PUTERBAUGH, K. D.; TROY, J. J. Six degree-of-freedom haptic rendering using voxel sampling. In: ACM SIGGRAPH 2005 COURSES. **Anais...** [S.l.: s.n.], 2005. p.42.

MICROSOFT. **Microsoft Visual Studio Professional 2012**, 2011. Disponível em: <https://www.microsoft.com/en-us/download/details.aspx?id=30682>. Acesso em: 20 fev. 2016.

MILGRAM, P. et al. Augmented reality: a class of displays on the reality-virtuality continuum. In: PHOTONICS FOR INDUSTRIAL APPLICATIONS. **Anais...** [S.l.: s.n.], 1995. p.282–292.

NUIGROUP. **Community Core Vision - CCV**, 2006. Disponível em: <http://ccv.nuigroup.com/#home>. Acesso em: 30 fev. 2016.

OPTICIENS, A. les. **Atol les Opticiens App**, 2015. Disponível em: <https://play.google.com/store/apps/developer?id=Atol+les+Opticiens>. Acesso em: 23 fev. 2016.

ORANGE, L. . **Timberland Augmented Reality Campaign**, 2005. Disponível em: <https://www.youtube.com/watch?v=5TZmQPdhpak>. Acesso em: 18 fev. 2016.

PENG, H. Application Research on Face Detection Technology based on OpenCV in Mobile Augmented Reality. **International Journal of Signal Processing, Image Processing and Pattern Recognition**, [S.l.], v.8, n.4, p.249–256, 2015.

SPIEGEL, E.; MURPHY, B.; BROWN, R. **Snapchat**, 2011. Disponível em: <https://www.snapchat.com>. Acesso em: 10 mar. 2016.

SZELISKI, R. **Computer vision**: algorithms and applications. [S.l.]: Springer Science & Business Media, 2010.

TEAM, U. **Cartoon Face**, 2012. Disponível em: <https://play.google.com/store/apps/details?id=com.cam001.crazyface>. Acesso em: 18 fev. 2016.

VIOLA, P.; JONES, M. J. Robust real-time face detection. **International journal of computer vision**, [S.l.], v.57, n.2, p.137–154, 2004.

WEISE, T. et al. Realtime performance-based facial animation. In: ACM TRANSACTIONS ON GRAPHICS (TOG). **Anais...** [S.l.: s.n.], 2011. v.30, n.4, p.77.