**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO**

**CENTRO TECNOLÓGICO**

**DEPARTAMENTO DE INFORMÁTICA**

**COLEGIADO DO CURSO DE CIÊNCIA DE COMPUTAÇÃO**

Miguel Rios Escóssia de Oliveira

# Reconstrução Facial a Partir de Escaneamento 3D com Kinect

Projeto de Graduação apresentado ao Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Orientador: Thiago Oliveira dos Santos

**VITÓRIA**

**2016**

**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO**

**CENTRO TECNOLÓGICO**

**DEPARTAMENTO DE INFORMÁTICA**

**COLEGIADO DO CURSO DE CIÊNCIA DE COMPUTAÇÃO**

Miguel Rios Escóssia de Oliveira

# Reconstrução Facial a Partir de Escaneamento 3D com Kinect

**COMISSÃO EXAMINADORA**

_____

**Prof. Thiago Oliveira dos Santos, D. Sc.**

_____

**Prof. 2, D. Sc.**

_____

**Prof. 3, D. Sc.**

**VITÓRIA**

**2016**

# RESUMO

A criação de personagens virtuais, conhecidos como “avatares”, atualmente é popularmente conhecida por ser uma das tarefas mais trabalhosas e demoradas de se realizar no mundo dos jogos digitais. Porém, é uma tarefa de muita relevância na maioria deles, por ser o momento inicial da criação de um personagem, onde lhe são atribuídas suas principais características visuais, que o farão único naquele mundo.

Uma das formas mais comuns, se não a mais comum, de criação de um avatar é atribuir-lhe características visuais que se assemelhem as características do jogador que o está criando, fazendo assim um avatar que representa sua cópia virtual do mundo real. Mas, para atingir este feito, é necessário que o jogador dispenda muito tempo e paciência para avaliar todas as características faciais do avatar e aceitá-las.

Buscando facilitar a criação de avatares de qualidade, este trabalho propõe o desenvolvimento de um sistema de criação automática de um avatar facial a partir de escaneamentos realizados pelo sensor Kinect, da Microsoft.

Este trabalho faz uso de estruturas de dados como Nuvens de Pontos, para manipulação dos dados dos escaneamentos do Kinect, e de Modelos Poligonais 3D, necessários para a criação dos avatares para os futuros usuários deste sistema. Também faz uso de técnicas de alinhamento de Nuvens de Pontos, como a de Registro Rígido e o algoritmo ICP, de criação e aplicação de texturas a modelos 3D, como a Parametrização de Superfícies 3D, e de detecção facial e de características faciais.

Foram analisados os sensores presentes no Kinect e entendidos os meios de captura de dados destes sensores. A partir destas análises, são capturadas nuvens de pontos com o Kinect e estas passam por processos de corte e suavização de pontos, para remoção de possíveis falhas de captura. Após esta etapa de captura e suavização de dados, estes são utilizados para a deformação de um modelo poligonal estatístico e então para a criação e aplicação de um arquivo de textura neste modelo.

Para validação da proposta deste trabalho, foram realizados testes de reconstrução facial com voluntários, e estes foram convidados a dar suas opiniões quanto a qualidade de suas respectivas reconstruções. Além de uma análise subjetiva de qualidade, também foi realizada uma análise objetiva, isto é, foi realizada uma medição de distâncias entre os dados de nuvem de pontos capturados durante os escaneamentos dos voluntários e os resultados das reconstruções faciais apresentados ao final.

Ao final, são feitas comparações entre os resultados das avaliações subjetivas e objetivas, e, consequentemente, são realizadas as conclusões sobre o trabalho como um todo. Além de apontamentos para possíveis melhorias e extensões deste trabalho.

**Palavras-chave**: Reconstrução facial 3D, Nuvens de pontos, Modelos estatísticos.

# ABSTRACT

The creation of virtual characters, also known as “avatars”, is now popularly known for being one of the most difficult and time-consuming tasks to accomplish in the world of digital games. However, it is a task of great importance in most of them, because that is the initial moment of creation of a character, which its main visual features are assigned, that will make the character unique in the world.

One of the most common ways, if not the most common, of creating an avatar is to give it visual features that resemble the features of the player who is creating, thus making an avatar that is a virtual copy of the real world person. But to achieve this, it is necessary that the player spend much time and patience to evaluate all the facial features of the avatar and accept them.

Seeking to facilitate the creation of avatars with higher quality, this work proposes the development of an automated system for the creation of a facial avatar from scans conducted by the Microsoft Kinect.

This work makes use of data structures such as Point Clouds, for manipulation of data from the Kinect scans, and 3D Polygonal Models, needed for the creation of avatars for future users of this system. It also makes use of Point Clouds alignment techniques, such as Rigid Registration and the ICP algorithm, creation and application of textures to 3D models, such as the technique of Parameterization of 3D Surfaces, and detection of faces and facial features.

The sensors present in Kinect were studied, as well as the forms of data capturing from it. From this research, point clouds are captured with Kinect and these go through cutting and smoothing processes, for removal of possible failures in the capture. After this step of data capture and smoothing, these captures are used for the deformation of a statistical polygonal model and then to the creation and application of a texture file in this model.

To validate the purpose of this work, facial reconstruction tests were conducted with volunteers, and they were invited to give their views on the quality of their respective reconstruction. Besides a subjective quality analysis, an objective analysis was also performed, i.e. a measurement of distances between the data of the point cloud captured during the scans of volunteers and the results of facial reconstructions presented at the end.

Finally, comparisons are made between the results of subjective and objective assessments, and consequently conclusions are made about the whole work. In addition to notes for possible improvements and extensions of this work.

**Keywords**: 3D facial reconstruction, Point clouds, Statistical polygonal model.

# LISTA DE FIGURAS

# SUMÁRIO

# 1 Introdução

Neste trabalho é apresentado um projeto de reconstrução facial de um personagem virtual (conhecido como "*avatar*") 3D a partir de digitalizações (*scans*) de faces humanas, utilizando o sensor de captura da Microsoft, o Kinect. Este foi concebido inicialmente como um controlador para o videogame da própria Microsoft, o Xbox 360, mas a partir da criação de drivers e bibliotecas de desenvolvimento de código aberto, seu uso como um sensor em um computador comum tornou-se viável, pois seu preço de aquisição é relativamente baixo comparado à maioria dos sensores usados para esta finalidade.

Existem muitas aplicações no cenário atual que demandam criação automática de avatares 3D. O mundo dos jogos digitais, principalmente o dos jogos digitais online, seja talvez o maior exemplo desta necessidade, como apontado por (ZOLLHÖFER *et al*., 2011). Os jogos multijogador massivos online (*Massive Multiplayer Online Games* – MMOG) como por exemplo *World of Warcraft*, *Aion* e *Lineage* chamam para si muita atenção, pois neles existem milhões de jogadores, que são representados por avatares que se encontram num mundo tridimensional. Partindo do princípio que cada jogador queira se distinguir dos outros, consideramos importante que eles devam ter um avatar único, como uma representação da vida real. Porém, a grande maioria dos MMOGs oferece fraca variedade de opções de personalização de seus avatares. Outros, que oferecem grande variedade de personalização, fazem com que o jogador tenha que empenhar muito tempo neste processo. Deste modo, na grande maioria dos casos, é inviável para um usuário comum criar uma cópia virtual no mundo online. Para contribuir para a resolução deste problema, este trabalho propõe criar um avatar de forma mais simples, automática e confiável. Isto é, a proposta é que a criação do avatar seja feita a partir de **Nuvens de Pontos 3D** do rosto do jogador, providas a partir de *scans* realizados pelo Kinect, este que deverá ser utilizado como um acessório para criação do avatar no jogo. Porém, o uso exclusivo destas nuvens de pontos não é viável, pois é muito mais difícil trabalhar com pontos soltos no espaço, sem ligações entre si. Deste modo, como os pontos não estão mapeados a lugar algum no espaço, a morfologia da nuvem é desconhecida para o desenvolvedor, dificultando muito o trabalho. A proposta é utilizar estes dados de nuvens de pontos aliados a **Modelos Poligonais 3D**, que são mais fáceis de serem trabalhados do ponto de vista do desenvolvedor. Ainda dentro desta proposta, este trabalho foi construído com o objetivo de minimizar a interação do usuário, realizando a maior quantidade do trabalho de construção possível sozinho, permitindo assim que usuários de qualquer nível de conhecimento possam facilmente criar seus próprios avatares 3D.

Além do uso em jogos online, avatares 3D personalizados também podem ser usados em diversas aplicações de simulação, por exemplo:

- Simulações de envelhecimento facial. Processo muito utilizado para realização de buscas de pessoas desaparecidas por muitos anos;
- Simulações de produção visual. Processo que serve para ajudar na escolha de opções de corte de cabelos, aplicação de maquiagens e toda sorte de produção visual em salões de beleza. O mesmo processo também pode servir para lojas de roupas e acessórios.

## 1.1 Objetivo do Trabalho

O objetivo geral deste trabalho é desenvolver e validar um sistema para reconstruir um avatar facial 3D de uma pessoa a partir de *scans* do Kinect de modo similar ao realizado por (ZOLLHÖFER *et al*., 2011) e, consequentemente, pôr em prática os conhecimentos de Programação (CELES *et al*., 2004), Computação Gráfica (FOLEY *et al*., 1995) e Estatística Básica (BUSSAB & MORETTIN, 2002), adquiridos ao longo do curso de graduação em Ciência da Computação. Este objetivo geral pode ser detalhado nos seguintes objetivos específicos:

- Implementar um módulo para aquisição de dados brutos (nuvens de pontos e *frames* RGB) dos sensores do Kinect;
- Realizar suavizações necessárias nas nuvens de pontos (corte de pontos desnecessários) para uma melhor manipulação das mesmas;
- Implementar um módulo para registro de nuvens de pontos e de geração de uma nuvem cheia (nuvem densa, com muito mais pontos que uma nuvem de pontos comum que possa ser adquirida na etapa anterior);
- Realizar deformação de um *morphable model* afim de conseguir um melhor encaixe em relação aos dados coletados;
- Realizar correspondência entre modelo final deformado e textura facial colorida;
- Realizar testes para validação do sistema com voluntários.

## 1.2 Trabalhos Relacionados

A prática de reconstrução de avatares faciais 3D tem se tornado bastante popular nos últimos anos. Existem dois tipos de algoritmos para realizarmos a reconstrução do avatar 3D, como constatado por (ZOLLHÖFER *et al*., 2011):

- Reconstruir diretamente de imagens 2D (fotografias da pessoa);
- Reconstruir a partir de dados de profundidade recebidos dos *scans* de um sensor.

O primeiro tipo de algoritmos, se refere a utilização de uma ou mais imagens da pessoa. Esta forma se baseia na extração de pontos de interesse de pelo menos uma imagem facial frontal e uma em perfil, como demonstrado por (TANG & HUANG, 1996). (KEMELMACHER-SHLIZERMAN & BASRI, 2011) apresentam uma proposta similar, porém utilizando uma única imagem como referência e levando em consideração algumas propriedades importantes presentes em faces humanas em geral, como tamanho, relação entre largura e altura (*aspect ratio*) e a localização dos principais pontos de interesse da face. (BREUER *et al*., 2008) utiliza Máquinas de Vetores de Suporte (*Support Vector Machines* – SVMs) para realizar a detecção da face na imagem 2D e então extrair os pontos de interesse da mesma utilizando técnicas probabilísticas de regressão e classificação. Todos estes exemplos fazem uso de um Modelo Facial Médio, muitas vezes chamado de *Morphable Model*, que representa o resultado da média de um conjunto de faces 3D. Basicamente tentam, de formas variadas, adaptar este modelo aos pontos de interesse encontrados nas imagens 2D.

Já o segundo tipo de algoritmo, adotado nesse trabalho, se refere a utilização de dados de profundidade provenientes de *scans* de sensores, e também consiste, em maioria, em fazer com que o modelo médio seja modificado (deformado) a fim de se aproximar da forma destes dados. Tais "dados de profundidade" são frequentemente referenciados neste trabalho como "nuvem de pontos", que significa um conjunto de pontos 3D representando o resultado de um *scan* realizado pelo sensor Kinect. (LI *et al*. 2008) apresenta um algoritmo deste tipo, que realiza Registro Não-rígido entre *scans* inicial e final. Neste caso, o *scan* final poderia ser substituído por um modelo médio, que possui as mesmas características básicas. (SCHEIDEGGER *et al.*, 2005) apresenta um algoritmo que, ao invés de deformar um modelo médio para fazê-lo se adaptar à nuvem de pontos, cria seu próprio modelo diretamente dos pontos da nuvem, baseado no método *Moving Least-Squares* (MLS) para reconstrução de superfícies a partir de um conjunto de pontos.

## 1. 3 Contribuições deste Trabalho

Este trabalho se utiliza de bibliotecas de terceiros para realizar algumas de suas partes, como o FaceTracker de (SARAGIH *et al*., 2009). O FaceTracker é uma biblioteca que dispõe de métodos de detecção de faces e características faciais em imagens 2D, e é utilizado neste trabalho por apresentar resultados na detecção facial que melhor se adequaram, em comparação às ferramentas de detecção da popular biblioteca OpenCV (OPENCV, 2016). De posse desta biblioteca, este trabalho apresenta uma forma eficiente de detecção de faces e características faciais em imagens 2D, que é parte essencial para o funcionamento correto do sistema.

Além do FaceTracker, este trabalho utiliza uma forma eficiente de criação e aplicação de texturas ao modelo alinhado e deformado, utilizando Parametrização de Superfícies 3D (MULLEN, 2009) e aplicações de equações de Coordenadas Baricêntricas (MÖBIUS, 1827). Trata-se de um procedimento de mapeamento de uma imagem 2D para um modelo poligonal 3D, e da criação de um arquivo de imagem que deverá ser preenchido com os valores das cores em cada um de seus pixels. Tais valores serão obtidos das nuvens de pontos capturadas anteriormente pelo Kinect e atribuídos aos respectivos pixels na imagem.

Ao fim desta etapa, teremos um arquivo de imagem com uma face desenhada – de modo que cada pixel está mapeado para algum ponto da nuvem capturada, exemplificado pelas figuras Figura 1 a, Figura 1 b, Figura 1 c e Figura 1 d – que será utilizado como textura para o modelo criado e deformado para se ajustar aos *scans* da face.

Figura 1 a: Textura do modelo

Figura 1 b: Nuvem cheia

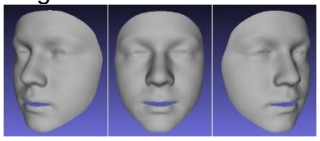

Figura 1 c: Modelo deformado

Figura 1 d: Modelo texturizado

## 1.4 Organização do Texto

Neste capítulo é apresentada uma introdução e uma breve descrição dos capítulos por vir deste trabalho. Além deste capítulo, este texto possui os seguintes capítulos:

- **Capítulo 2 – Fundamentação Teórica**: Apresenta quais são e dá uma breve fundamentação teórica sobre as técnicas computacionais utilizadas na construção deste trabalho.
- **Capitulo 3 – Reconstrução Facial 3D**: Descreve todo o processo de reconstrução facial 3D abordado neste trabalho, desde a aquisição dos dados do Kinect até a aplicação final de textura no modelo deformado. Também está descrito nesta seção a maneira pelos quais foram obtidos os pontos de interesse da face do usuário.
- **Capítulo 4 – Metodologia Experimental**: Descreve a maneira como foram realizados todos os experimentos para validação do que foi proposto inicialmente. Além de descrever detalhes técnicos sobre bibliotecas e ferramentas utilizadas na construção do projeto computacional, arquitetura básica do sistema e resultados dos experimentos de reconstrução realizados com voluntários.
- **Capítulo 5 – Conclusão**: Apresenta as considerações finais do trabalho, incluindo algumas dificuldades encontradas, contribuições e experiência adquirida em seu desenvolvimento. Também são identificados alguns trabalhos futuros.

# 2 Fundamentação Teórica

Nesta seção, serão esclarecidos como as técnicas utilizadas funcionam neste trabalho. As principais técnicas utilizadas são as técnicas de Detecção de Faces e Características Faciais, técnicas de Registro de Nuvens de Pontos e técnicas de Criação e Deformação de *Morphable Models*. Também foi utilizada a forma padrão de aplicação de texturas especificada pelo formato de arquivos OBJ (REDDY, 2016).

## 2. 1 Detecção de Faces e Características Faciais

Detecção facial é uma técnica de processamento de imagem e visão computacional criada para determinar a existência, ou não, de faces numa determinada imagem e, caso exista(m), retornar a localização da(s) mesma(s). Detectar características faciais consiste em: dada uma cena, encontrar onde está a face, ou várias faces, bem como identificar seus componentes como olhos, boca, nariz. Apesar de ser uma tarefa trivial para seres humanos, é um problema desafiador para computadores, tendo em vista que rostos podem variar em cor, iluminação, posicionamento e escala, dificultando a detecção automática (ZHANG & ZHANG, 2010). Segundo (LOPES, 2003), as principais técnicas de detecção de faces são:

- **Métodos Baseados em Conhecimento**: representam as técnicas que de alguma maneira codificam algum conhecimento sobre o que é uma face. Geralmente se baseiam em atributos geométricos da face codificados em forma de regras.
- **Métodos Baseados em Características Invariantes**: representam as técnicas que identificam características da face independente de sua orientação. Tais técnicas geralmente utilizam segmentação de pele e modelagem estatística da textura da face humana.
- **Métodos Baseados em *Templates***: é uma técnica geral utilizada para detectar objetos, onde o mesmo é representado por uma família de curvas que modelam suas características geométricas. Também pode ser considerada uma técnica baseada em conhecimento, uma vez que, o projeto da *template* necessita do conhecimento da forma do objeto.
- **Métodos Baseados na Aparência**: representa as técnicas que não necessitam de conhecimento prévio sobre a característica a ser detectada. Geralmente as técnicas que pertencem a este grupo, necessitam de várias imagens e, a partir delas, aprendem ou codificam somente o que é necessário para realizar a detecção da característica de interesse, sem a necessidade de intervenção humana. Exemplos de tal abordagem são as Redes Neurais, *Eigenfaces* e Modelos Ocultos de Markov.

## 2. 2 Métodos de Registro de Nuvens de Pontos

Os principais métodos utilizados neste trabalho são os métodos de registro de nuvens de pontos 3D. Tais métodos representam sequencias de transformações que devem ser realizadas em uma das nuvens em questão (normalmente chamada de “origem”), para que se aproxime o máximo possível da outra (normalmente chamada de “destino”).

Os métodos de registro de nuvens de pontos são essenciais para o funcionamento deste trabalho, tendo em vista que os sensores do Kinect capturam várias nuvens de pontos à medida que o tempo passa. Estas nem sempre representam a mesma área da face da pessoa. Por exemplo, em um momento o Kinect pode estar escaneando o lado esquerdo do rosto da pessoa e em outro momento pode estar escaneando o lado direito, gerando nuvens de pontos completamente diferentes da mesma pessoa. Deste modo, se faz necessário o uso de métodos de registro entre estas nuvens de pontos capturadas.

Os métodos utilizados neste trabalho são o método Registro de Corpo Rígido – ou apenas Registro Rígido – e o algoritmo ICP (*Iterative Closest Point*), descritos a seguir.

### 2.2.1 Registro Rígido

Dados dois conjuntos de pontos pareados (ou seja, cada ponto de um conjunto possui um par no outro), esta técnica produz uma transformação rígida que mapeia um conjunto ao outro. Uma transformação rígida é uma transformação que não altera a distância entre os pontos do mesmo conjunto, ou seja, a transformação é aplicada igualmente para todos os pontos do conjunto, fazendo com todas as distâncias entre quaisquer dois pontos (em cada conjunto) sejam mantidas. As transformações que compõem este tipo de registro são tipicamente translação, rotação, espelhamento e combinações destas ao conjunto de pontos (FITZGIBBON, 2003). As figuras Figura 2 a, Figura 2 b, Figura 2 c e Figura 2 d exemplificam este tipo de transformações. Qualquer objeto manterá a mesma forma e tamanho após aplicação transformações rígidas.

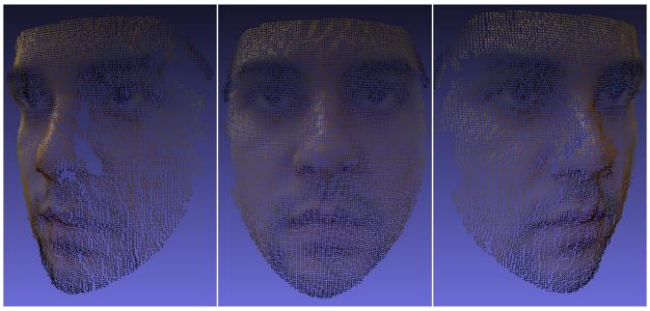

Figura 2 a: Escaneamento frontal, distribuição equilibrada de pontos.

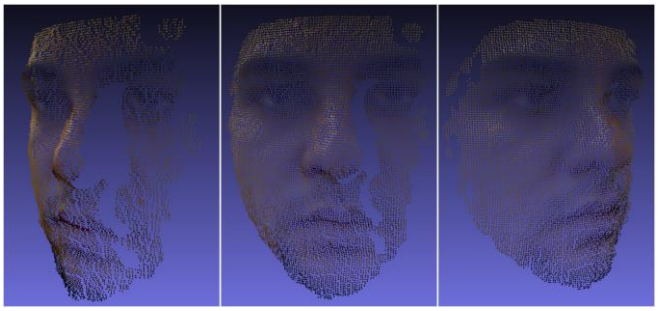

Figura 2 b: Escaneamento lateral direito do rosto, pontos se concentram em apenas um lado.

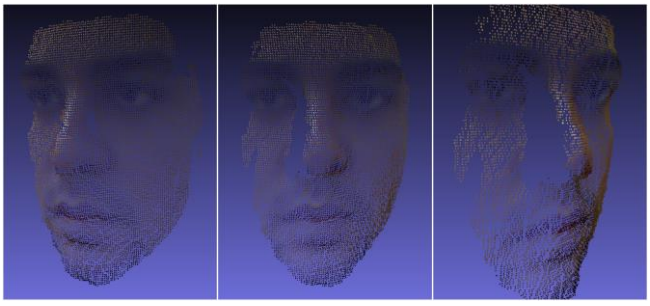

Figura 2 c: Escaneamento lateral esquerdo do rosto.

Figura 2 d: Nuvem Cheia, resultado do registro das nuvens anteriores.

A técnica de Registro Rígido é complementada pela técnica de Registro Não-Rígido, que mapeia um conjunto de pontos a outro a partir de transformações não-rígidas. Transformações não-rígidas incluem transformações como Escalas (aumentar ou diminuir o tamanho do objeto, isto é, aumentando ou diminuindo as distâncias entre os pontos do conjunto) e Cisalhamento (provocar a deformação do objeto, mantendo uma direção paralela).

### 2.2.2 ICP (Iterative Closest Point)

O ICP (POMERLEAU *et al*., 2015) é um algoritmo largamente utilizado no campo de registro de conjuntos de pontos, com o objetivo de minimizar as diferenças entre dois conjuntos (BESL & McKAY, 1992; ZHANG, 1994). Neste trabalho, este algoritmo é empregado em reconstruções de superfícies 3D a partir de diferentes *scans* de um mesmo local ou objeto, por exemplo: o rosto do usuário pode ser capturado de vários ângulos diferentes, resultando em nuvens de pontos potencialmente diferentes de uma mesma pessoa. Estas nuvens podem ser registradas para uma única nuvem, resultando num objeto mais cheio e detalhado do que as nuvens capturadas anteriormente.

Assim, uma nuvem de pontos, chamada de "referência", ou "alvo", é mantida, enquanto a outra, chamada de "origem", sofre transformações para melhor se ajustar à referência. A função do algoritmo é calcular iterativamente a transformação (combinação de translações e rotações) necessária para minimizar a distância da nuvem de pontos origem para a referência, mantendo as iterações até que um critério (geralmente distância média entre os pontos das duas nuvens) seja atingido (BESL & McKAY, 1992). Basicamente o algoritmo é composto por quatro passos:

- Para cada ponto da nuvem de pontos origem, encontre o ponto mais próximo na nuvem de pontos referência;
- Estime a combinação de rotação e translação utilizando Erro Quadrático Médio (*Mean Squared Error* – MSE) como função de custo para alinhar da melhor maneira possível cada ponto da origem com seu correspondente encontrado no passo anterior;
- Aplique a transformação encontrada no passo anterior aos pontos da origem;
- Refaça do passo 1 (reassociar os pontos e continuar) até que o critério de parada seja atingido.

Existem muitas outras variações do ICP. Por exemplo, (ZHANG, 1994) propõe um algoritmo que utiliza Árvores K-Dimensionais (*K-D Trees*) (BENTLEY, 1975) para uma busca mais eficiente pelo ponto mais próximo.

## 2. 3 Morphable Face Model

Um *Morphable Face Model* representa um modelo facial 3D poligonal, formado por vértices e arestas, que pode ser facilmente modificado, deformado, alterado, etc. Isto é, as informações referentes às coordenadas dos pontos podem ser alteradas livremente, fazendo com que sua forma visual também seja modificada.

Um modelo deste tipo é criado a partir de uma base de escaneamentos de faces de várias pessoas diferentes. Para extrair um modelo a partir de dados ruidosos como escaneamentos, é de grande ajuda ter um modelo base que represente o formato básico do objeto que se deseja extrair. Normalmente estas bases são geradas por artistas humanos para classes específicas de modelos, como faces de pessoas de etnias diferentes (africanos, asiáticos, latinos, europeus, etc.). Um exemplo de modelo base criado artisticamente são os *blendshape models* (LI *et al*., 2010). Este modo de gerar bases requer um conhecimento prévio grande sobre as possíveis deformações que uma classe de modelos pode – ou não – ter, além de ser um processo não-automatizado custoso e, por vezes, tedioso. Em outros estudos foram utilizadas técnicas de *Machine Learning* para gerar um *morphable model* a partir de um conjunto de *scans* de treinamento como a base de dados proposta

por (YIN *et al*. 2006), usando análises estatísticas. Deste modo temos uma maneira mais rápida e automatizada de encontrar modelos base, como analisado por (BRUNTON *et al*., 2014). O modelo utilizado neste trabalho foi criado utilizando a técnica apresentada em (BRUNTON *et al*., 2014) a partir da base de dados proposta por (YIN *et al*. 2006).

## 2. 4 Aplicação de Texturas em Objetos 3D

Seguindo o objetivo principal deste trabalho, o processo de reconstrução facial consiste em gerar um modelo facial 3D que seja o mais próximo possível dos dados coletados pelo Kinect. Neste processo, passaremos por uma etapa de criação de um modelo facial 3D em si, que não necessariamente será colorido. Deste modo, não é esperado que o resultado seja satisfatório, em termos de aparência. É possível relacionar uma cor a um objeto 3D, mas objetos do mundo real não são formados apenas por cores sólidas. Para resolver este problema é necessária uma técnica de aplicação de textura ao modelo 3D, onde esta textura deve ser uma imagem referente ao rosto do usuário. Esta técnica de aplicação de textura é composta por duas etapas:

- Uma etapa de parametrização do modelo 3D sem cores para pixels em uma imagem 2D;
- Uma etapa de criação e preenchimento desta imagem para o qual o modelo foi mapeado.

Ou seja, primeiro é realizado um mapeamento, conhecido como Mapeamento UV, do modelo 3D para uma imagem hipotética, com dimensões padrão que não vão mudar, no caso deste trabalho as dimensões são de 512x512 pixels. Após isso é criada a imagem em si, preenchendo com valores RGB os pixels que foram mapeados a algum ponto no modelo 3D. Como o modelo 3D é um *morphable model*, todos os resultados que podem ser gerados pelo sistema construído neste trabalho são deformações (sem adição ou remoção de pontos) de um modelo base. Logo, se o mapeamento for realizado com o modelo base, este mapeamento também servirá para os modelos finais gerados pelo sistema. Sendo assim, este mapeamento "modelo-imagem" só deve ser realizado uma única vez, e poderá ser utilizado sempre para qualquer deformação do modelo base.

Em seguida, para cada deformação diferente (correspondente a um resultado de uma pessoa diferente), uma nova imagem deverá ser criada, e então ela já estará automaticamente parametrizada para o modelo deformado correspondente. Enfim, a criação desta imagem é feita a partir dos pontos da Nuvem Cheia correspondente, basicamente buscando o ponto 3D correspondente ao pixel 2D e copiando seus valores RGB para o pixel. A aplicação deste processo de parametrização do modelo 3D e de preenchimento da imagem de textura é mostrado na seção 3.6.

### 2.4.1 Parametrização de Superfícies 3D

O processo de parametrização de superfície do modelo médio é conhecido como Mapeamento UV (*UV Mapping*). O mapeamento UV é um processo de projeção de uma imagem 2D na superfície de um modelo 3D e permite que polígonos 3D sejam coloridos de forma mais realista, com as cores provenientes dos pixels de uma imagem (MULLEN, 2009). As letras "U" e "V" representam as coordenadas do plano da imagem a ser mapeada, pois as letras "X", "Y" e "Z" já são usadas para representar as coordenadas do espaço 3D do polígono.

## 2. 5 Formatos de Armazenamento de Dados: Formato OBJ

Arquivos *Object* (REDDY, 2016), também conhecidos como *Wavefront OBJ*, são arquivos criados para serem utilizados pela aplicação de visualização de figuras geométricas *Advanced Visualizer*, desenvolvida pela empresa *Wavefront Technologies*. Este tipo de arquivo foi criado com o objetivo de guardar informações de figuras geométricas, tais como pontos, linhas, polígonos, superfícies, curvas, etc.

Este trabalho se utiliza fortemente deste tipo de arquivo, pois é uma das melhores maneiras encontradas de representar nuvens de pontos 3D, modelos poligonais e ainda ter a possibilidade de descrever dados de texturas para os modelos. A maioria dos outros formatos de descrição de figuras geométricas encontradas não possui uma forma simples de embutir dados de texturas num mesmo conjunto de informações, como o formato OBJ faz. A organização mais simples de um objeto no formato OBJ é composta pelas seguintes partes:

- Dados de vértices;
- Faces, compostas pelos vértices antes citados;
- Atributos de renderização de materiais.

### 2.5.1 Material Library File (.mtl)

Arquivos de definição de materiais (*Material Library Files*) (BOURKE, 2016) contém uma ou mais definições de material, cada uma com um mapeamento de cores, textura e reflexões de cada material definido. Estas definições são aplicadas em superfícies e vértices de objetos definidos nos arquivos OBJ. Arquivos MTL são salvos em formato ASCII e possuem a extensão ".mtl". Este tipo de arquivo difere de outros tipos desenvolvidos pela *Alias|Wavefront* justamente no ponto que se trata de definir mais de um material, pois outros formatos normalmente permitem apenas uma definição por arquivo.

Em um arquivo MTL, cada nova definição de material consiste em uma declaração *newmtl*, que atribui um nome ao material e marca o início da definição de um novo material. Após esta declaração, seguimos declarando as cores do material e o mapa de textura, ou seja, o arquivo de imagem propriamente dito (BOURKE, 2016). As definições que seguem a declaração "*newmtl*" não precisam necessariamente seguir esta ordem. A Tabela 1 exemplifica as estruturas dos arquivos OBJ e MTL necessárias para a visualização de um objeto com os dados descritos no arquivo OBJ e seus dados de iluminação e textura descritos no arquivo MTL.

| | |
|---|---|
| mtllib capsule.mtl<br>v 0.0 0.0 0.0<br>v 0.0 1.0 0.0<br>v 1.0 0.0 0.0<br>...<br>vt 0.000000 0.000000<br>vt 0.010000 0.000000<br>vt 0.020000 0.000000<br>...<br>usemtl material0<br>f 1/1 2/2 3/3<br>... | newmtl material0<br>Ka 1.000000 1.000000 1.000000<br>Kd 1.000000 1.000000 1.000000<br>Ks 0.000000 0.000000 0.000000<br>Tr 1.000000<br>illum 1<br>Ns 0.000000<br>map_Kd capsule0.jpg |

Tabela 1: Exemplo de organização básica de um polígono em arquivo OBJ. À esquerda temos o conteúdo do arquivo ".obj", que descreve os dados dos vértices (v), das faces (f) e do mapeamento dos pixels do arquivo de textura com relação aos vértices (vt). À direita temos o conteúdo de um arquivo auxiliar de extensão ".mtl", que descreve os dados de iluminação, transparência e finalmente o arquivo que contém a imagem a ser utilizada como textura do objeto.

# 3 Reconstrução Facial 3D

Esta seção descreve todo o processo de reconstrução abordado neste trabalho, desde a aquisição dos dados dos sensores do Kinect até a aplicação final de textura no modelo deformado. Tais dados serão, deste ponto em diante, referenciados como “pontos” ou “nuvem(ns) de pontos”. Também está descrito nesta seção a maneira pelos quais foram obtidos os pontos de interesse da face do usuário. Para uma ideia geral, podemos analisar a Figura 3 com o fluxograma dos principais passos realizados neste trabalho.

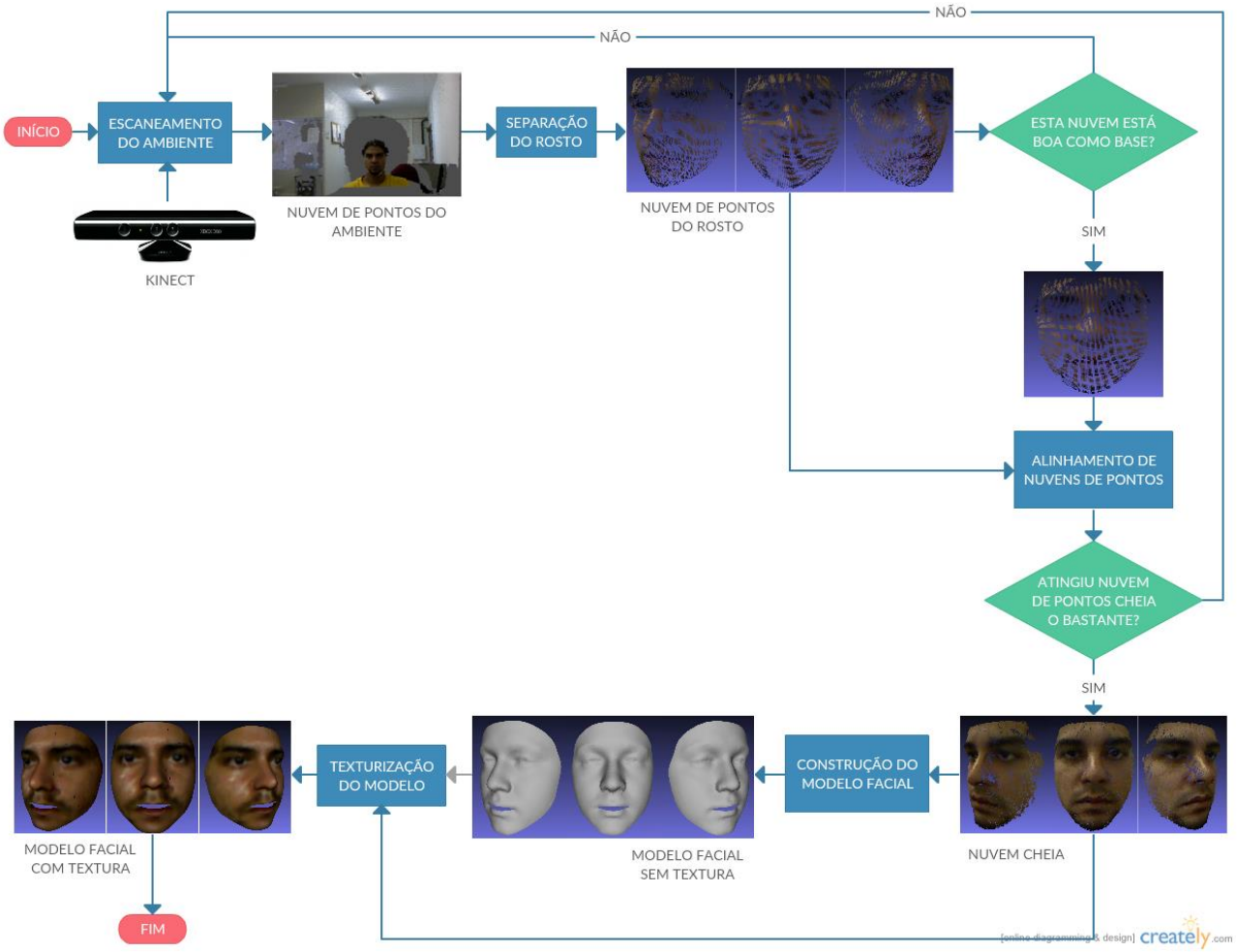


Figura 3: Fluxograma básico dos passos que compõem este trabalho

O método abordado neste trabalho é composto basicamente por duas etapas distintas: **aquisição de dados** e **manipulação de dados**.

O primeiro passo trata-se da aquisição primária dos dados brutos dos sensores do Kinect, isto é, o Kinect se encarrega de escanear o cenário a partir de seu sensor infravermelho e então a biblioteca PCL dispõe de métodos para recepção da nuvem de pontos proveniente do Kinect.

Em seguida, chegamos no passo em que temos que separar a face do resto do cenário, pois a nuvem de pontos do cenário escaneado representa justamente o cenário completo ao qual o Kinect está apontado. Então há necessidade de fazermos cortes na nuvem de pontos do cenário para que restem apenas os pontos referentes ao rosto do usuário. Este passo faz uso de outra fonte de dados do Kinect, a câmera RGB. Com a imagem RGB somos capazes de gerar uma série de pontos 2D

(pixels) que descrevem as fronteiras da face do usuário e seus principais pontos de interesse (características faciais), como olhos, nariz e boca. Deste modo são realizadas correspondências entre os pixels da imagem e os pontos da nuvem, e então podemos escolher apenas os pontos da nuvem que estão relacionados à face detectada na imagem. Ao final destes passos, teremos como saída uma nuvem de pontos referente ao rosto do usuário. Como esta nuvem é apenas um fragmento da nuvem completa do cenário escaneado pelo Kinect, temos em mãos uma nuvem com uma baixa resolução (poucos pontos). Para amenizar este problema, repetimos os passos anteriores um certo número de vezes, aconselhando que o usuário mude a direção do rosto e, ao final de todos os escaneamentos, esperamos obter uma nuvem de pontos do rosto agora com uma resolução maior (muitos pontos). Chamaremos esta nuvem de "Nuvem Cheia". Após este passo, está terminada a etapa de aquisição de dados.

De posse da nuvem cheia, entramos na etapa de manipulação de dados. Aqui a nuvem cheia obtida é utilizada como entrada para um método que tem como função deformar um *morphable model* (BRUNTON *et al*., 2014) de modo que este alcance um formato mais próximo do formato da nuvem cheia.

Após isso, obtemos um modelo facial próximo dos dados reais coletados, porém sem nenhuma cor. Este problema foi resolvido com a aplicação de textura ao modelo resultante, tendo como textura uma imagem que representa a face do usuário. Este problema é subdividido em outros dois problemas menores: realizar uma parametrização da superfície do *morphable model* para uma imagem de textura e criar uma imagem a partir da nuvem cheia do usuário. A parametrização de superfície pode ser realizada apenas uma vez para qualquer imagem a ser utilizada como textura (seção 3.6), e a criação da imagem de textura se aproveita do trabalho realizado na parametrização. Após todos estes passos, obteremos uma reconstrução facial no mínimo razoável, realizando testes de qualidade automatizados e também subjetivos, levando em consideração análises pessoais de convidados sobre a qualidade das reconstruções.

## 3.1 O Sensor Microsoft Kinect

Como já mencionado anteriormente, o sensor utilizado neste trabalho é o Microsoft Kinect, ilustrado na Figura 4. Este aparelho não é apenas um sensor, mas uma combinação de sensores variados, a saber:

- Uma câmera RGB padrão, que provê imagens 2D com resolução de 640x480 pixels a uma frequência máxima de 30 Hz;
- Um sensor de profundidade, composto por um projetor e um receptor de raios infravermelhos. O projetor lança vários raios infravermelhos no local onde está apontado, estes raios são refletidos pelos objetos presentes e capturados pelo receptor. Deste modo, para cada raio capturado, são calculadas as diferenças entre o correspondente lançado e então temos os valores de profundidade;
- Conjunto de microfones, usados para captação e reconhecimento de voz e supressão de ruídos externos. Estes sensores não serão utilizados neste trabalho;
- Motor de movimento, usado para modificar verticalmente o ponto de visão do Kinect. Também não será utilizado neste trabalho.

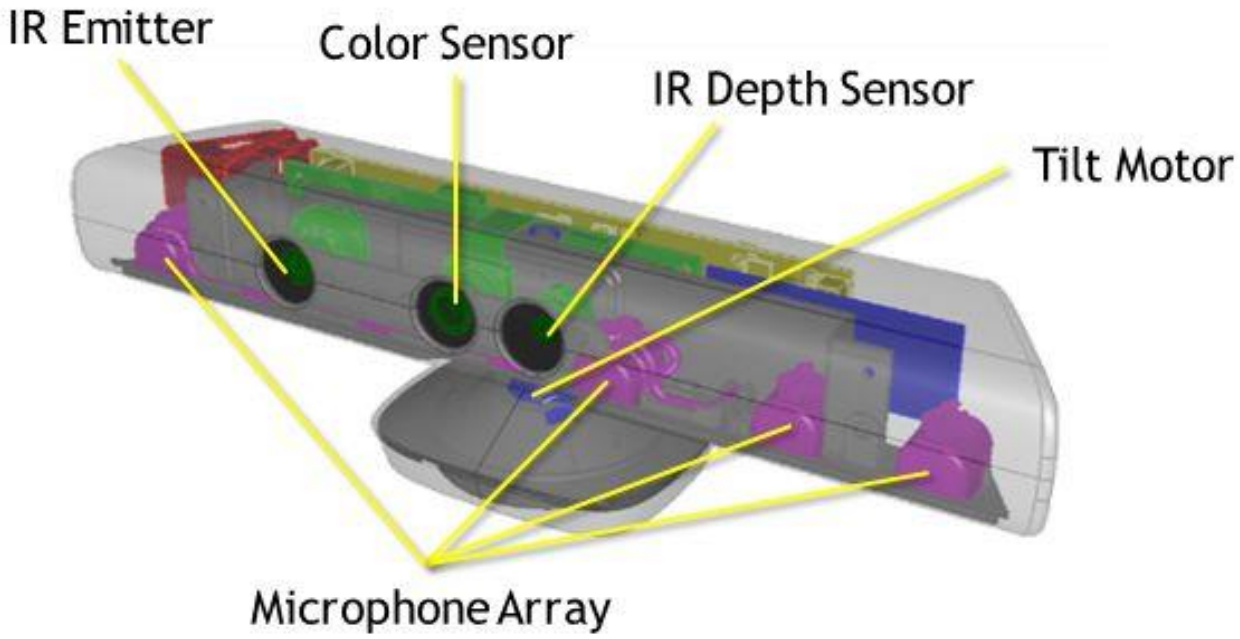


Figura 4: Ilustração dos sensores presentes no Kinect (MICROSOFT, 2016).

## 3. 2 Captura de Dados

Os sensores infravermelho do Kinect, em conjunto com os drivers da biblioteca PCL proporcionam uma representação gráfica, em formato de nuvem de pontos, de todos os objetos do local ao qual o Kinect está apontado. Ao mesmo tempo, o sensor RGB do Kinect também captura imagens da mesma cena, o que será de grande utilidade posteriormente.

Segundo informações disponibilizadas pela própria fabricante, o sensor de profundidade do Kinect necessita de uma distância mínima de 80cm para realizar captura de dados. Isto faz com que o ângulo de visão do sensor infravermelho seja bastante alto, retornando uma nuvem de pontos exageradamente grande de todo o cenário, com muitos pontos desnecessários para este trabalho, como ilustrado na Figura 5. Para resolver este problema inicial, é preciso realizar cortes na nuvem de pontos, para que restem apenas os pontos correspondentes à face da pessoa em questão, faz-se necessário o uso das imagens retornadas pelo sensor RGB, processo explicado a seguir.

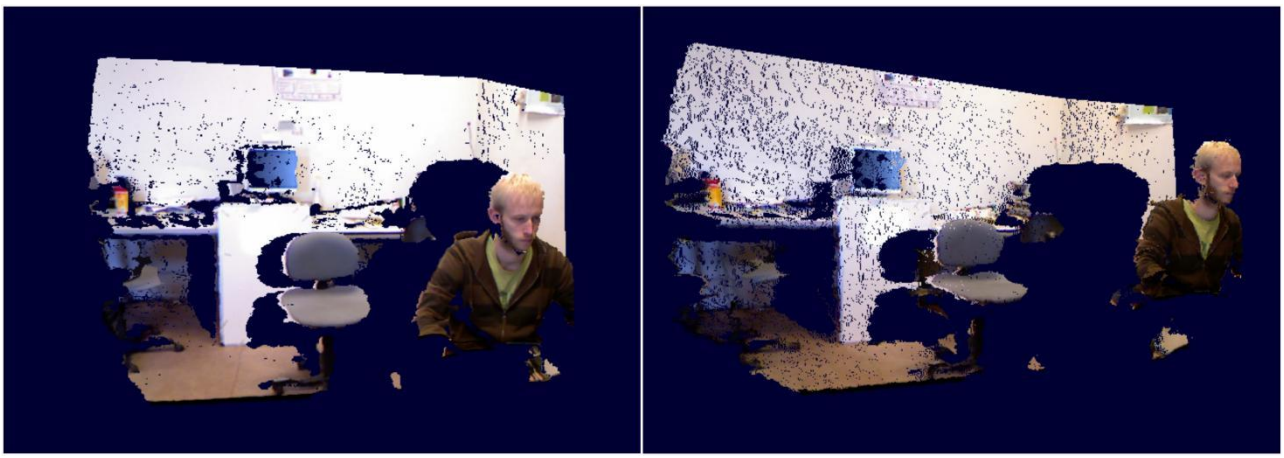

Figura 5: Exemplo de nuvem de pontos do cenário retornada pelo Kinect (BURRUS, 2016)

## 3. 3 Cortes das Nuvens de Pontos Capturadas

Com a imagem RGB podemos utilizar a biblioteca FaceTracker para realizar a detecção da face e das características faciais da pessoa em questão. O método de detecção da face provido pelo FaceTracker é capaz de detectar faces em vários ângulos de visão, desde o ângulo frontal padrão até algo em torno da pose totalmente lateral. Porém, ao se distanciar muito do ângulo frontal, o método de detecção tende a perder um pouco a qualidade, fazendo com que a face detectada seja representada como um "fragmento de face". Então, após testes variados com o Kinect, foi observado que os ângulos mais apropriados entre o eixo de visão do usuário e o sensor, para o uso deste método, variam entre 0 e 45 graus. Valores além de 45 graus não apresentam garantia de boas detecções, como representado pela Figura 6.

Figura 6: Exemplos de detecção pelo FaceTracker de ângulos de visão variados.

Este método de detecção das características faciais retorna uma série de pontos, que representam os contornos da face. Estes pontos são ligados para criar um polígono, cuja área interna é igual a área da imagem que contém a face. Os pontos que representam as sobrancelhas são elevados para também cobrir a área da testa, ilustrado pela Figura 7. Ao final deste processo, temos como saída uma área de pixels referente à face do usuário na imagem 2D.

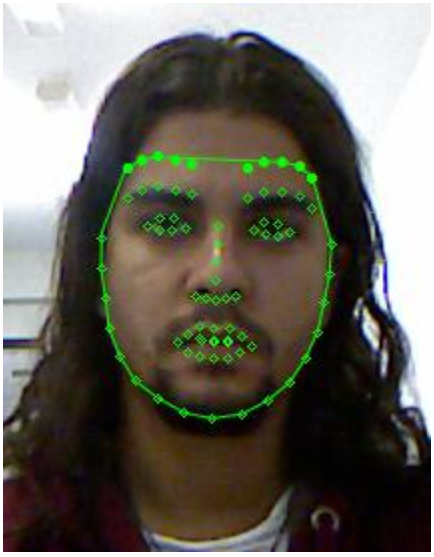

Figura 7: Ilustração do processo realizado com os pontos retornados pelo FaceTracker. Os pontos referentes às sobrancelhas são elevados de um valor proporcional à distância entre os olhos detectados. Após isto são utilizados os pontos que formam o contorno da face para criar um "polígono da face".

Tendo detectada a área da imagem correspondente a face da pessoa, pode-se realizar correspondências entre os pixels da imagem do rosto com os pontos da nuvem referentes a esses pixels. Deste modo, com a ajuda de módulos da biblioteca PCL, finalmente temos uma nuvem de pontos com apenas os pontos referentes ao rosto da pessoa. Este processo é ilustrado nas figuras Figura 8 a e Figura 8 b.

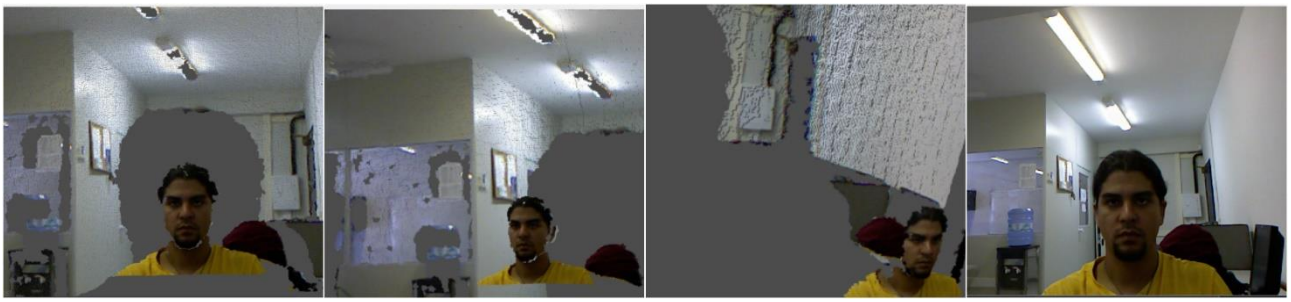

Figura 8 a: Nuvem de pontos do ambiente, antes de sofrer cortes. A última imagem é a imagem 2D referente ao mesmo momento de captura da nuvem do ambiente.

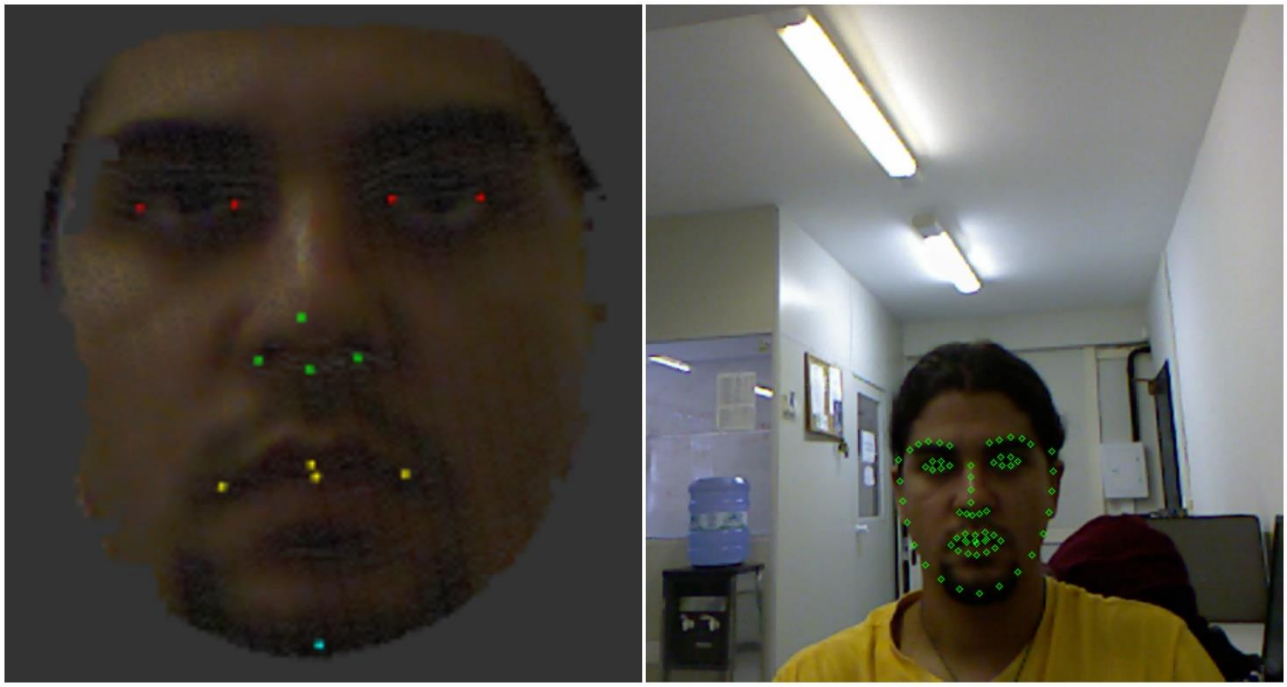

Figura 8 b: Nuvem de pontos do rosto, após sofrer cortes de pontos desnecessários. À direita temos a imagem 2D referente ao mesmo momento de captura, mas com a face detectada, o que possibilitou o corte dos pontos.

Este processo é realizado para todas nuvens de pontos capturadas pelo Kinect. Como o Kinect não é um sensor com o foco em realizar escaneamentos de alta qualidade, quase sempre obtemos nuvens de pontos com muitos ruídos e/ou áreas do rosto não tão fieis ao modelo real (rosto do usuário no mundo real). Deste modo, foi realizado um processo de alinhamento de nuvens de pontos, com o intuito de suavizar os ruídos das nuvens retornadas pelo Kinect. Tal processo será explicado a seguir.

## 3. 4 Alinhamento das Nuvens de Pontos

Neste processo, seguimos o intuito de tentar suavizar o máximo possível os ruídos provenientes dos *scans* do Kinect. Sabemos que, como o Kinect não é muito preciso em seus escaneamentos, teremos resultados diferentes a cada atualização recebida do aparelho, mesmo que não haja alteração alguma no cenário observado. Deste modo, o processo realizado para suavização consiste em salvar uma certa quantidade de nuvens de pontos retornadas pelo Kinect e alinhá-las de modo que fiquem todas numa mesma posição, encaixadas umas às outras. Este processo consiste em duas etapas:

- Uma etapa de aplicação da técnica do Registro Rígido: neste momento são registradas duas nuvens de pontos de diferentes ângulos de visão do rosto do usuário. Para a aplicação do registro rígido entre nuvens de pontos, é necessário passar para o algoritmo alguns pontos que são sabidamente correspondentes nas duas nuvens. No caso, foram utilizados pontos referentes a partes dos olhos, do nariz, boca e queixo, destacados na Figura 8 b. A partir destes pares de pontos e da escolha de uma nuvem como base, a técnica de registro rígido é aplicada à outra nuvem, fazendo com que ela se alinhe com a base, minimizando as distâncias entre os pares de pontos destacados em ambas as nuvens. Porém, por melhor que seja, esta técnica ainda deixa distâncias entre as nuvens que podem ser diminuídas ainda mais. O resultado deste alinhamento pode ser visualizado pela Figura 9.
- Uma etapa refinamento no registro, com o algoritmo ICP: neste momento o ICP realiza, para cada ponto de uma nuvem, transformações (combinações de rotação e translação) para tentar diminuir ainda mais as distâncias entre os pontos das duas nuvens. Este algoritmo compõe a segunda etapa pois necessita que as nuvens de pontos estejam tão próximas quanto possível. Caso elas não estejam muito próximas, a otimização do algoritmo tende a piorar o alinhamento, ao invés de melhorá-lo. O resultado deste alinhamento também pode ser visualizado pela Figura 9.

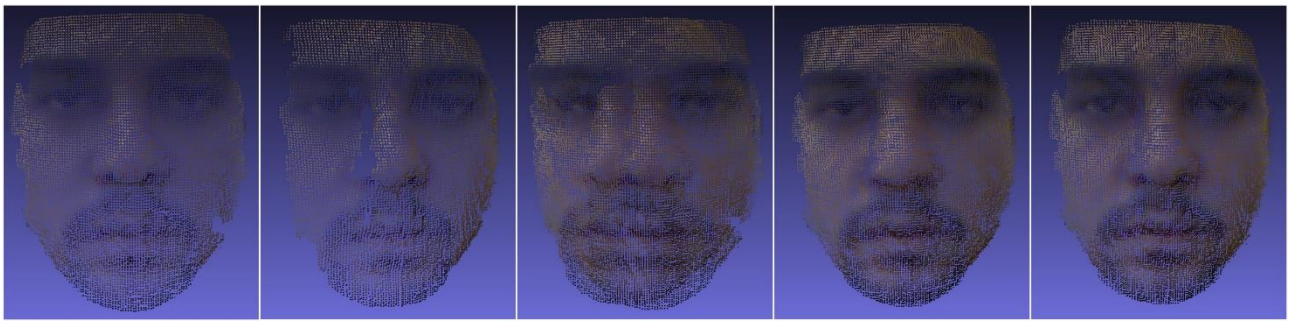

Figura 9: Momentos importantes do alinhamento: Nuvem de pontos base e uma nuvem proveniente de outro *scan*, seguidas da nuvem que representa as duas alinhadas sem utilização de recursos de registro. Após esta, temos a nuvem resultado do alinhamento por Registro Rígido e o alinhamento final pelo algoritmo ICP. Neste último caso o alinhamento alcançou uma qualidade maior e temos uma face mais nítida.

Ao fim deste processo teremos uma única nuvem, porém com muito mais pontos do que uma nuvem de pontos normalmente retornada pelo aparelho e com grande parte dos ruídos suavizados. Esta nuvem, que será o resultado de vários *scans*, é chamada de “nuvem cheia”. As Figura 10 ilustra este processo de alinhamento e suavização.

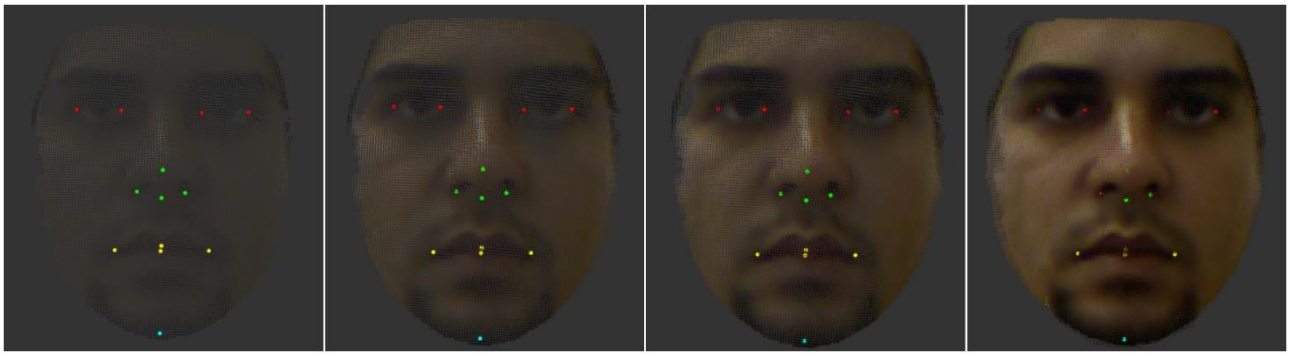

Figura 10: Processo de criação da "nuvem cheia", fazendo com que a nuvem de pontos básica retornada pelo Kinect (primeira imagem) se torne mais nítida e com suavização nas falhas (última imagem). Pontos destacados são algumas características faciais importantes para a realização do Registro Rígido, detectadas pelo FaceTracker.

## 3.5 Uso do Morphable Model

Após a etapa de suavização e criação da nuvem cheia, partimos para o próximo processo, que é o de manipulação do *morphable model*, ou simplesmente "modelo médio", que é o modelo facial 3D que representa uma média de várias faces escaneadas por outros aparelhos, diferentes do Kinect, com maior precisão.

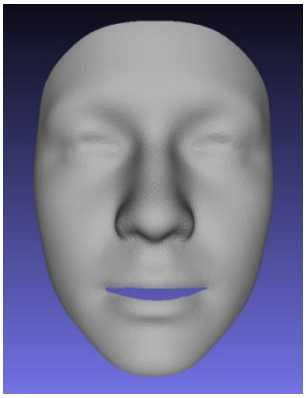

Figura 11: Morphable Model utilizado como base para reconstrução facial

Nesta etapa é aplicado o método proposto por (BRUNTON *et al*., 2014), onde são feitos os alinhamentos e deformações necessários no modelo médio para que se torne o mais parecido possível com a nuvem cheia do usuário. O modelo é primeiramente alinhado com a nuvem cheia usado o método de registro rígido. Como este método requer pares de pontos como entrada, temos que passar como entrada os pontos correspondentes na nuvem e no modelo. Tais pontos são justamente os pontos de características faciais citados anteriormente. Após este alinhamento, são realizadas deformações na forma do modelo, de modo que este se ajuste o máximo possível com a nuvem cheia passada como entrada. A saída é o *morphable model* deformado para representar a face do usuário. A Figura 12 ilustra os resultados esperados.

Figura 12: Com a Nuvem Cheia como parâmetro para a deformação do Modelo Médio, gerando assim o Modelo Resultado.

Mas ainda temos um problema visível, o modelo médio – e consequentemente o modelo final deformado – não possuem cores, diferente da nuvem cheia. Para resolver este problema, foi utilizado um processo de aplicação de texturas, para que o modelo deformado possa se tornar um modelo final texturizado.

## 3. 6 Aplicação de Textura ao Modelo Final

Nesta etapa temos que criar uma imagem que deverá estar mapeada como textura do modelo deformado anteriormente, dando cores a ele. Esta imagem deve conter o rosto da pessoa que foi escaneada pelo Kinect na etapa de aquisição de dados. Assim, para realizarmos a texturização é necessário:

- Realizar parametrização de superfície entre o modelo médio e uma imagem hipotética de dimensões fixadas (neste caso será 512x512 pixels);
- Criar uma imagem do rosto do usuário com as mesmas dimensões;
- Aplicar esta imagem como textura ao modelo final.

### 3.6.1 Parametrização da Superfície do Morphable Model

O processo de parametrização de superfície do modelo médio (mapeamento UV) envolve mapear pixels numa imagem 2D hipotética a pontos no polígono 3D para depois atribuir os valores das cores de um triângulo nesta imagem aos valores num triângulo no polígono (MURDOCK, 2008). Ou seja, para cada ponto do modelo médio, haverá um pixel referente na imagem, e para cada triângulo 3D no polígono haverá uma pequena área de pixels 2D na imagem. Este processo de parametrização é realizado com facilidade por alguns softwares de manipulação de objetos e cenários 3D. O software utilizado para apoiar esta etapa foi o Blender (http://blender.org), que, dentre várias outras, apresenta uma funcionalidade que, ao receber um modelo 3D como entrada, retorna um arquivo SVG (*Scalable Vector Graphics*) com todos os vértices e arestas 3D mapeados para um plano 2D. Este arquivo representa todos os pixels, de uma possível imagem de textura, que terão seus valores RGB atribuídos para os polígonos do modelo deformado. Este processo é ilustrado na Figura 13.

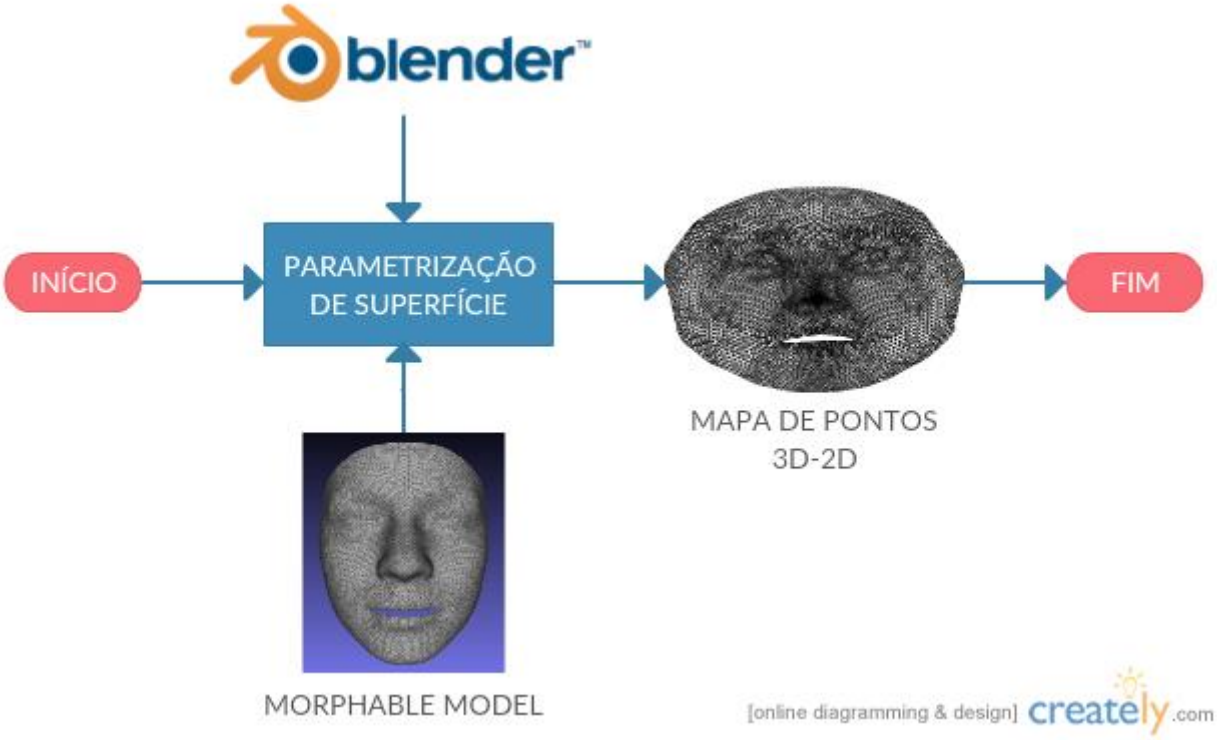


Figura 13: Fluxograma do processo de mapeamento UV realizado pelo Blender para o Morphable Model utilizado neste trabalho.

Tendo em mãos este mapeamento, o modelo pode sofrer quaisquer deformações nas coordenadas XYZ de seus pontos, desde que não sejam adicionados (ou retirados) pontos. Tendo em vista que, ao final da etapa de deformação do modelo médio, o modelo deformado terá os mesmos pontos que o modelo médio, apenas com alterações em suas coordenadas XYZ, o modelo deformado seguirá o mesmo mapeamento realizado para o modelo médio. Deste modo, a imagem – que deve se encaixar na área mapeada na Figura 13 – poderá ser utilizada como textura do modelo deformado. Ou seja, todo o processo de parametrização de superfície do *morphable model* deve ser feita apenas uma vez e reaproveitada para todas as deformações subsequentes de todos os usuários, desde que o *morphable model* seja o mesmo. Esta etapa de parametrização pode ser realizada antes mesmo de qualquer outra etapa, inclusive as etapas de captura de dados dos sensores, pois as deformações futuras utilizarão apenas o resultado da parametrização. Resta então criarmos a imagem a ser utilizada como textura do modelo deformado.

### 3.6.2 Criação e Aplicação da Imagem de Textura

A imagem de textura deve conter o rosto do usuário como ilustrado anteriormente pela Figura 1 a, e para criá-la foi necessário um algoritmo que precisa da Nuvem Cheia como entrada, pois é dela que serão retirados os valores de intensidade de cores necessários para cada pixel da imagem. Poderíamos ter utilizado o *frame* RGB de algum dos momentos da criação da nuvem cheia, mas nenhum deles poderia conter todos os lados da face do usuário em apenas uma imagem. O processo de criação da imagem é ilustrado pela Figura 14.


Figura 14: Fluxograma básico do processo de criação da imagem de textura com o mapa de pontos e a Nuvem Cheia como entradas e base para obtenção dos valores RGB dos pixels.

O algoritmo de criação da imagem é composto pelos seguintes passos:

1 Para cada pixel da imagem mapeada, aqui chamado de P, devemos encontrar qual triângulo 2D – que representa uma das faces do modelo médio – ele se encontra;

2 O ponto P pode ser definido por meio de coordenadas baricêntricas (MÖBIUS, 1827) a partir dos três pontos que representam o triângulo que P está inserido – aqui chamados de P1, P2 e P3 – da seguinte maneira:

3 A partir do ponto P e dos três pontos que representam o triângulo que o pixel está inserido – aqui chamados de P1, P2 e P3 – é possível calcular as coordenadas baricêntricas de cada P a partir das seguintes equações:

$$x = \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 \qquad y = \alpha_1 y_1 + \alpha_2 y_2 + \alpha_3 y_3 \qquad \alpha_1 + \alpha_2 + \alpha_3 = 1$$

Onde $x$ e $y$ são as coordenadas cartesianas do ponto P e os pares $x_1$ e $y_1$, $x_2$ e $y_2$, $x_3$ e $y_3$ são as coordenadas cartesianas referentes aos pontos P1, P2 e P3. As coordenadas baricêntricas de P – os fatores $\alpha_1$, $\alpha_2$ e $\alpha_3$ – são as os comprimentos dos segmentos que unem o ponto P às arestas do triângulo passando pelos pontos P1, P2 e P3. Podemos visualizar melhor na Figura 15.

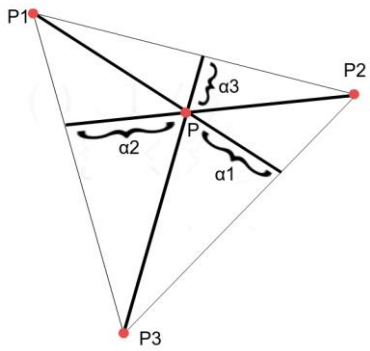


Figura 15: Representação visual das coordenadas baricêntricas do ponto P.

Com estas equações obtemos os valores de $\alpha_1$, $\alpha_2$ e $\alpha_3$ para o ponto P no triângulo formado pelos pontos P1, P2 e P3;

4 A partir do mapeamento entre o modelo e a imagem, recuperamos os pontos M1, M2 e M3 do modelo deformado correspondentes aos pontos P1, P2 e P3, formando um novo triângulo, agora 3D. Neste, temos que recuperar o ponto imaginário M correspondente ao pixel P da imagem. Para fazer isso são utilizados os mesmos cálculos anteriores utilizando os mesmos valores de $\alpha_1$, $\alpha_2$ e $\alpha_3$. Deste modo encontramos os valores das coordenadas XYZ do ponto imaginário M correspondente ao ponto P;

5 Ao obter o ponto imaginário M, e sabendo que o modelo está alinhado com a nuvem cheia, realizamos buscas pelos pontos mais próximos ao ponto M na nuvem cheia. Tais buscas são, geralmente, realizadas utilizando ferramentas de busca implementadas pelas próprias bibliotecas de manipulação de pontos 3D. Neste trabalho, a biblioteca PCL apresenta excelentes módulos de busca utilizando *K-d Trees* (BENTLEY, 1975);

6 Dos pontos mais próximos, são extraídos seus valores de intensidade de cores e calculadas médias. Isto é, a partir de um conjunto de, por exemplo, 20 (vinte) pontos vizinhos, são obtidos 20 valores de intensidade RGB. É calculada uma média aritmética destes valores e então o resultado é finalmente aplicado ao pixel P.

Desta forma são preenchidos os valores de cores necessários para cada pixel da imagem a ser utilizada como textura do modelo deformado, formando assim uma imagem com formato semelhante ao da Figura 1 a. Como já descrito anteriormente, o modelo deformado é apenas uma modificação das coordenadas XYZ dos pontos existente no modelo médio inicial, isto é, ele possui os mesmos pontos que o modelo médio, apenas com valores XYZ diferentes. Não somente isso, o modelo deformado, por ser criado a partir de uma cópia do modelo médio, também possui a mesma parametrização de superfície. Deste modo, basta que exista uma imagem com as dimensões sugeridas na parametrização – e que esteja preenchida na área parametrizada – para que seja realizada a aplicação de textura a partir do arquivo OBJ. Isto é, ao término do processo de preenchimento da imagem a aplicação de textura está também terminada.

# 4 Metodologia Experimental

Esta seção descreve a maneira como foram realizados todos os experimentos para validação do que foi proposto inicialmente por este trabalho. Além de descrever detalhes técnicos sobre bibliotecas e ferramentas utilizadas para apoiar a construção do projeto computacional, arquitetura básica do sistema e, também, apresentar resultados dos experimentos de reconstrução realizados com voluntários.

## 4. 1 Bibliotecas e Arquitetura

As principais ferramentas utilizadas neste trabalho foram:

- *Open Source Computer Vision* (OpenCV): biblioteca que fornece, entre outras coisas, funcionalidades de visão computacional em tempo real;
- *Point Cloud Library* (PCL): biblioteca de algoritmos focada em processamentos de nuvens de pontos e formas geométricas 3D;
- *FaceTracker*: Ferramenta criada com base no OpenCV, com o objetivo de detectar faces e características faciais;
- Software de criação e deformação do *Morphable Model*.

Além de ferramentas, este projeto apresenta uma arquitetura simples em módulos com diferentes objetivos, com a seguinte disposição (representação visual na Figura 16):

- **Cloud IO**: módulo de leitura e escrita de nuvens de pontos em arquivos, geralmente nos formatos OFF e OBJ, mas apresentando também funções para leitura e escrita em arquivos PCD;
- **Cloud Operations**: módulo que provê métodos que realizem diversas manipulações com nuvens de pontos, como criar uma nova nuvem, adicionar pontos a uma nuvem existente, somar duas nuvens, realizar algum tipo de registro em duas nuvens, etc.;
- **Detector**: módulo responsável por integrar a biblioteca FaceTracker e utilizar sua capacidade de detecção de faces e características faciais 2D;
- **Grabber**: módulo principal do projeto, onde ocorre a interface entre o Kinect e o software, realizando a captura dos dados em tempo real e os deixando disponíveis para manipulação;
- **Scan3D**: módulo chave para o projeto, onde reúne em si todas as funcionalidades providas pelos módulos “Cloud IO” e “Cloud Operations”, oferecendo um objeto poderoso e muito útil para o desenvolvimento deste trabalho.

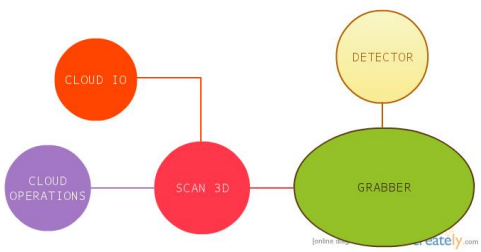


Figura 16: Arquitetura básica do sistema proposto neste trabalho

### 4.1.1 OpenCV

OpenCV (OPENCV, 2016) é uma biblioteca *open-source* de algoritmos focados em visão computacional e *machine learning*. Foi construída com o objetivo de prover uma estrutura básica para aplicações de visão computacional e para acelerar seu uso em produtos comercializáveis, lançados sob a licença BSD.

Esta biblioteca contém mais de 2500 algoritmos otimizados, os quais incluem um conjunto tanto de algoritmos clássicos quanto de algoritmos em "estado da arte", de visão computacional e *machine learning*. Tais algoritmos podem ser utilizados para aplicações de reconhecimento facial, identificação de objetos, classificação de ações humanas em vídeos, extrair modelos 3D de objetos, produzir nuvens de pontos 3D a partir de câmeras estéreo, unir imagens para produzir uma imagem de maior resolução de um cenário, encontrar imagens similares em bancos de imagens, remover olhos vermelhos de fotos tiradas com flash, seguir movimentos oculares, etc. O OpenCV possui uma grande comunidade de usuários (mais de 47 mil usuários), e é amplamente utilizada por empresas, grupos de pesquisa e órgãos governamentais.

O OpenCV é escrito em C++ e atualmente possui interfaces em C, C++, Python, Java e MATLAB e é suportado nos sistemas Windows, Linux, Android e Mac OS. Por objetivar aplicações de visão computacional em tempo real, o OpenCV se utiliza de instruções em baixo nível – quando disponíveis – para alcançar o maior desempenho possível. Atualmente estão sendo desenvolvidas interfaces também para CUDA e OpenCL. Atualmente esta biblioteca é mantida pelo grupo Itseez (http://itseez.com/).

### 4.1.2 PCL

A PCL (RUSU & COUSINS, 2011) é uma biblioteca *open-source* de algoritmos para processamento de nuvens de pontos e de formas geométricas 3D, e também para visão computacional tridimensional. Esta biblioteca é composta de algoritmos para estimar pontos de interesse, reconstrução de superfícies, alinhamento de pontos, ajuste de modelos, etc. É escrita em C++ e lançada sob licença BSD e este trabalho foi concebido integrado com a versão 1.7 desta biblioteca.

Tais algoritmos são utilizados, por exemplo, no campo da robótica, para filtrar valores discrepantes de conjuntos de dados ruidosos, alinhar duas nuvens de pontos 3D, segmentar partes relevantes de um cenário, extrair pontos chave, organizar objetos baseado suas aparências geométricas, criar superfícies a partir de nuvens de pontos e visualizá-las.

A PCL possui suporte nativo a interfaces 3D, possibilitando a aquisição e processamento de dados provenientes de aparelhos como as câmeras 3D da PrimeSense, o Kinect da Microsoft, ou o sensor XTionPRO da Asus. Este suporte é o item vital para a construção deste trabalho.

### 4.1.3 FaceTracker

FaceTracker é uma biblioteca criada para detecção de faces e características faciais. Esta apresenta o uso de técnicas probabilísticas de agrupamento como SCMS (*Subspace Constrained Mean-Shifts*) (CHENG, 1995), de estimação de função densidade de probabilidade como KDE (*Kernel Density*

*Estimate*) (ROSENBLATT, 1956), e técnicas de *Machine Learning* para fazer com que um modelo parametrizado – formado por um conjunto de pontos de interesse – seja encaixado e ajustado a uma imagem de modo que seus pontos correspondam a partes desta imagem (SARAGIH *et al*., 2009).

A união de todas estar técnicas resulta no que é conhecido como ASM (*Active Shape Modeling*). ASM são modelos estatísticos (como o utilizado neste trabalho) com formatos pré-selecionados, que sofrem deformações iterativamente até que alcancem um formato de modelo numa imagem (COOTES *et al*., 1995). As deformações dos modelos são limitadas pela forma modelo PDM (*Point Distribution Model*), para que ocorram apenas como visto em um conjunto de exemplos de treinamento. A forma do objeto (face) é representada por um conjunto de pontos. A técnica ASM tem como objetivo corresponder um modelo a uma nova imagem.

A versão desta biblioteca utilizada neste trabalho encontra-se em http://facetracker.net/ e é atualmente mantida por Kyle McDonald (http://kylemcdonald.net/).

### 4.1.4 Software de criação e deformação do Morphable Model

Desenvolvido por (BRUNTON *et al*., 2016), este software faz parte de um esforço de criação para propósitos de pesquisa não-comercial e tem como propósito carregar um modelo de polígonos estatístico facial e realizar deformações até que este se encaixe a uma nuvem de pontos ou uma malha de triângulos. Para tal, é necessário prover alguns pontos de interesse da face como entrada para o software. Tais pontos são providos na etapa de reconhecimento das características faciais na metodologia deste trabalho.

## 4. 2 Experimentos

Foram realizadas reconstruções faciais em uma seleção de cerca de 15 voluntários com fisionomias variadas, resultando em reconstruções boas e ruins, dependendo do formato facial de cada voluntário. Os voluntários se submeteram aos experimentos sugeridos neste trabalho, primeiramente com a parte de aquisição de dados, onde passaram por vários escaneamentos diante do sensor Kinect e posteriormente com a parte de reconstrução facial. Após esta parte, foi realizada uma avaliação de qualidade da reconstrução facial, que é composta por:

- Uma avaliação computacional, onde é calculado um mapa de distâncias entre a nuvem cheia (construída na etapa de aquisição de dados pelo Kinect) e o modelo final (resultante da deformação do modelo médio). Esta avaliação deve resultar em um novo arquivo de imagem com o mesmo formato que o arquivo utilizado como textura para o modelo final, porém com os dados de intensidade ajustados para mostrar as distâncias entre os pontos da nuvem e o modelo final. Este arquivo é preenchido de tal forma que mostra visualmente se a reconstrução ficou ou não fiel aos dados coletados, colorindo de azul as áreas onde os pontos da nuvem estão a uma distância próxima de 0 (zero) do modelo, e de intensidades cada vez mais avermelhadas as áreas que os pontos estão a distâncias próximas ou maiores que um valor definido (neste trabalho este valor é definido em 1cm, ou seja, o limite máximo aceitável que um conjunto de pontos pode estar do modelo final é de 1 centímetro). Um exemplo deste mapa de distâncias pode ser visualizado na Figura 17.

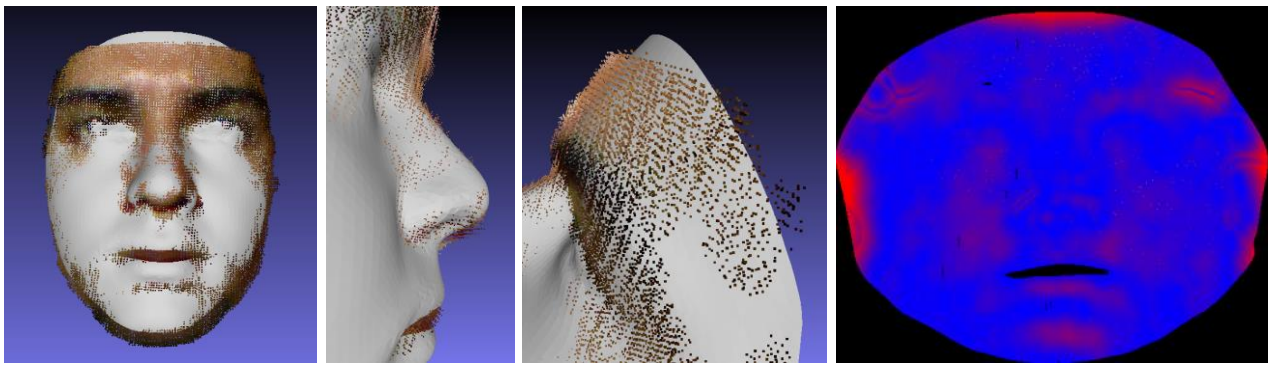

Figura 17: Mapa que representa a distância euclidiana entre o modelo final e a nuvem cheia, onde as áreas azuis representam pontos que estão muito próximos do modelo e as áreas vermelhas representam pontos que estão distantes (máximo 1cm)

- Uma avaliação subjetiva, onde os voluntários são convidados a dar suas opiniões num pequeno questionário com perguntas relacionadas à qualidade da reconstrução realizada. Este questionário foi criado com a ferramenta de criação de formulários do Google, o Google Forms, e se encontra no endereço http://goo.gl/forms/WLJOuxXiy5.

## 4. 3 Resultados e Discussões

Após realizados os experimentos com os voluntários, podemos realizar análises e discussões sobre os resultados, apresentados a seguir. Os resultados apresentados estão dispostos da seguinte maneira:

- **Nuvem Base**: nuvem com a densidade de pontos de apenas 1 (um) *scan*, utilizada como base para os alinhamentos dos próximos *scans*;
- **Nuvem Cheia**: nuvem de pontos formada pelo alinhamento de *scans* de vários ângulos diferentes da face do usuário;
- **Modelo Deformado**: resultado do processo de deformação do Modelo Médio utilizado neste trabalho;
- **Modelo Final**: resultado da texturização do Modelo Deformado, utilizando um arquivo de textura da face do usuário;
- **Arquivo de Textura**: imagem utilizada como textura para a criação do Modelo Final;
- **Mapa de Distâncias**: imagem que mostra as áreas do Modelo Deformado que ficaram mais (ou menos) próximas da Nuvem Cheia (que representa mais fielmente a face real do usuário).

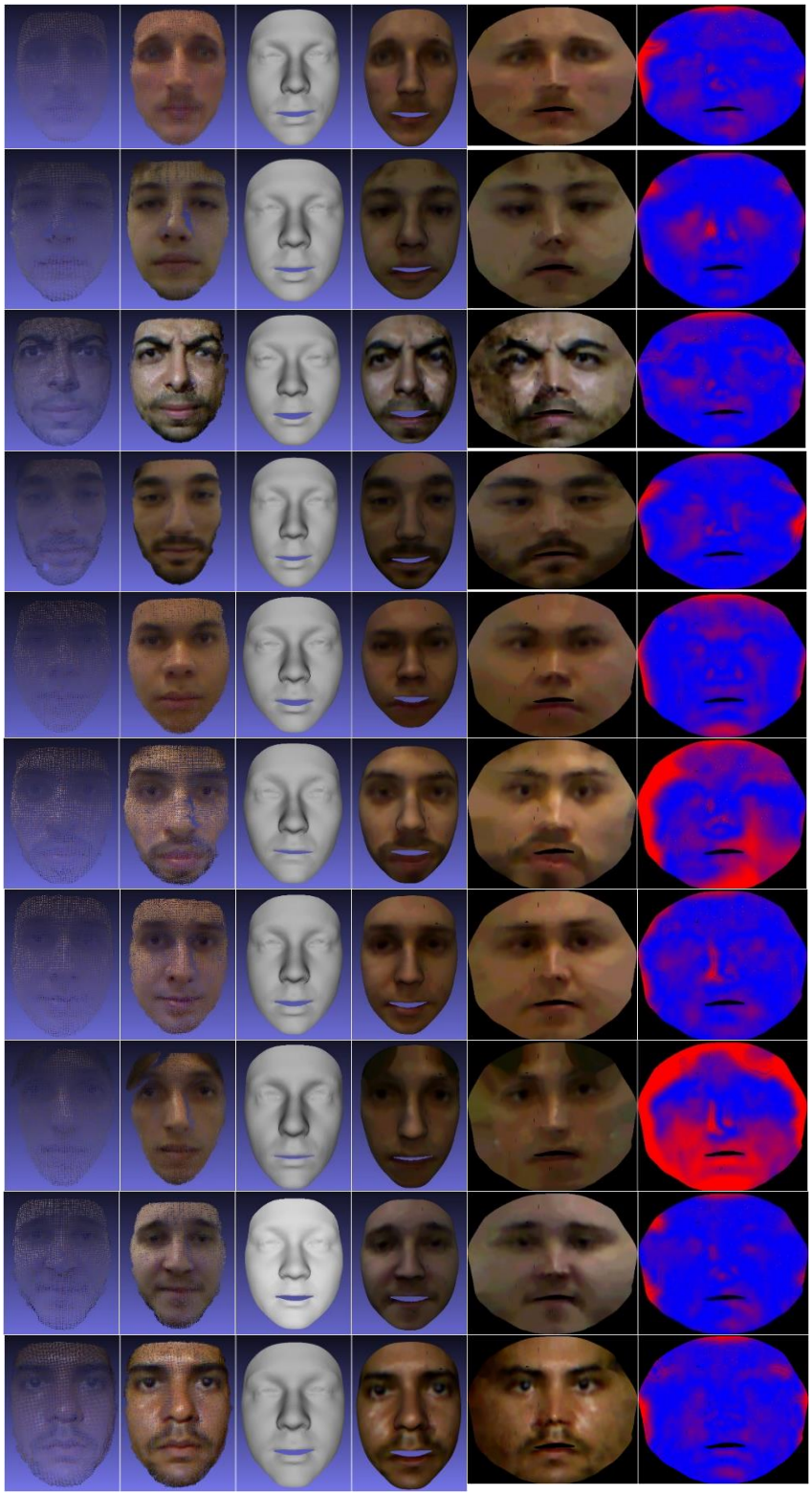

Figura 18: Resultados dos experimentos realizados com voluntários do sexo masculino

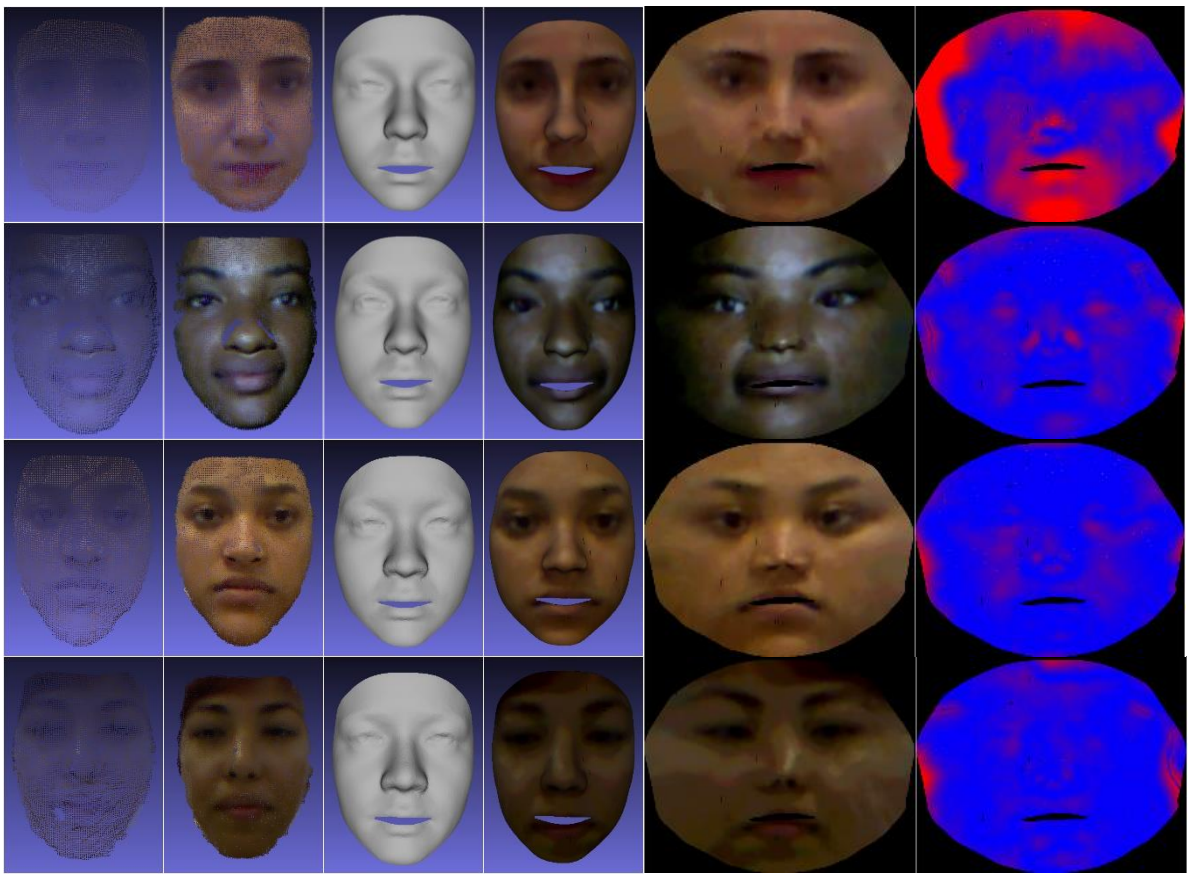

Figura 19: Resultados dos experimentos realizados com voluntários do sexo feminino

Tendo como partida os resultados mostrados nas figuras Figura 18 e Figura 19, podemos observar que a maioria dos exemplos teve bons resultados na avaliação automática, isto é, tendo o mapa de distâncias com quase todos os pixels com valores RGB próximos a (0, 0, 255) (em tons de azul). Mesmo os experimentos que apresentaram áreas vermelhas consideravelmente grandes possuem detalhes curiosos com relação à área dos olhos, onde ambos estão bem próximos dos dados de suas Nuvens Cheias. Logo, podemos concluir que o processo de deformação do modelo é bom nesta região da face, mesmo que apresente problemas em outras regiões. Isto possivelmente é proveniente de um treinamento anterior com faces que não eram muito próximas das faces de alguns voluntários deste trabalho.

Um outro ponto interessante a se discutir é a diferença visível entre os resultados com e sem a presença de texturas, principalmente nas avaliações subjetivas. Os resultados das reconstruções sem texturas, apesar de geometricamente iguais aos resultados texturizados, não trazem a intuição de serem boas reconstruções, a não ser que seja feita uma comparação visual entre os pontos da nuvem cheia e o modelo deformado, como mostrado na Figura 17. Isto nos dá uma noção de que, mesmo com apenas mudanças visuais não-geométricas, a qualidade da reconstrução pode mudar (aumentar ou diminuir, dependendo da avaliação do usuário). Isto foi observado nas respostas dos voluntários quanto a qualidade da reconstrução antes e depois da aplicação da textura da face. As avaliações dadas pelos voluntários foram respostas às seguintes perguntas, ilustradas nas figuras Figura 20 a, Figura 20 b, Figura 21 a e Figura 21 b:

- Qual a sua opinião sobre a qualidade da reconstrução antes da aplicação da textura ao modelo?
- Qual a sua opinião sobre a qualidade da reconstrução após a aplicação da textura ao modelo?

- Após ver o mapa de distâncias (distância euclidiana calculada automaticamente entre a nuvem cheia e o modelo resultado) e o arquivo de textura da face, sua opinião mudou?
- Caso positivo para a pergunta anterior, qual é a sua nova opinião sobre a qualidade da sua reconstrução facial?

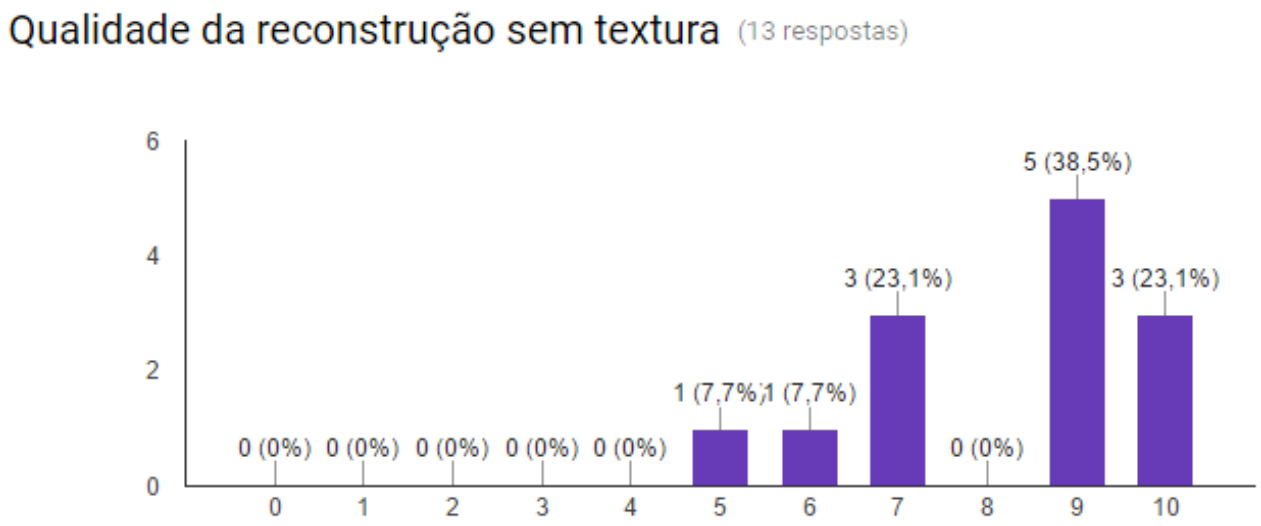


Figura 20 a: Respostas obtidas com relação a qualidade da reconstrução facial ainda sem aplicação de textura.

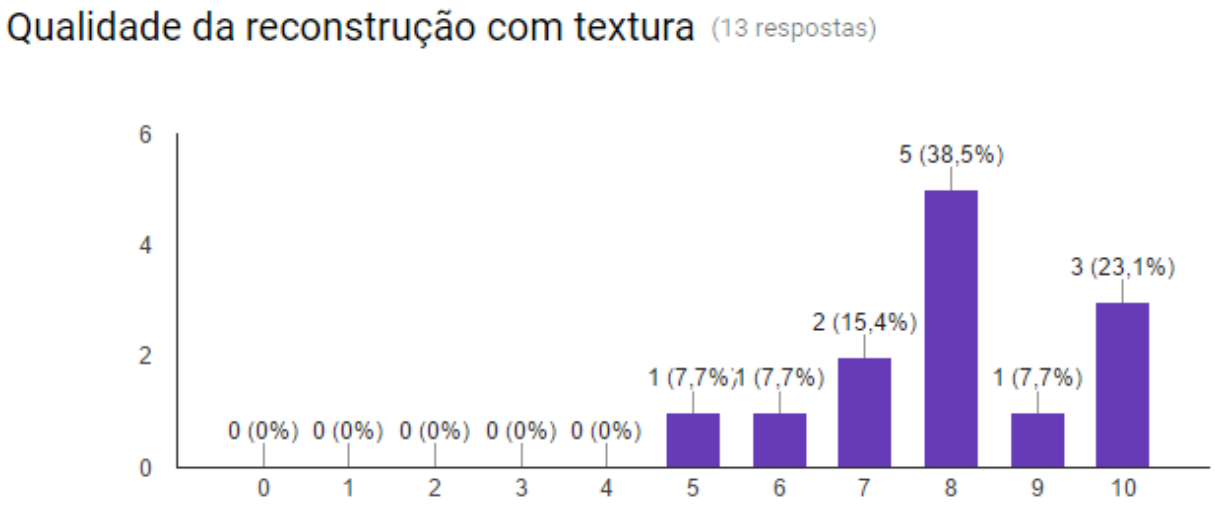


Figura 20 b: Respostas com relação a qualidade da reconstrução após a aplicação das texturas.

As respostas duas primeiras questões revelam que as opiniões com relação as qualidades das reconstruções com e sem texturas são diferentes, o que era esperado. Porém, ao analisarmos as médias de pontuação de cada pergunta, chegamos a um valor próximo de 8 pontos em ambas, isto é, a qualidade das reconstruções, tanto com texturas quanto sem texturas, foi avaliada como muito boa, segundo os usuários que responderam ao questionário.

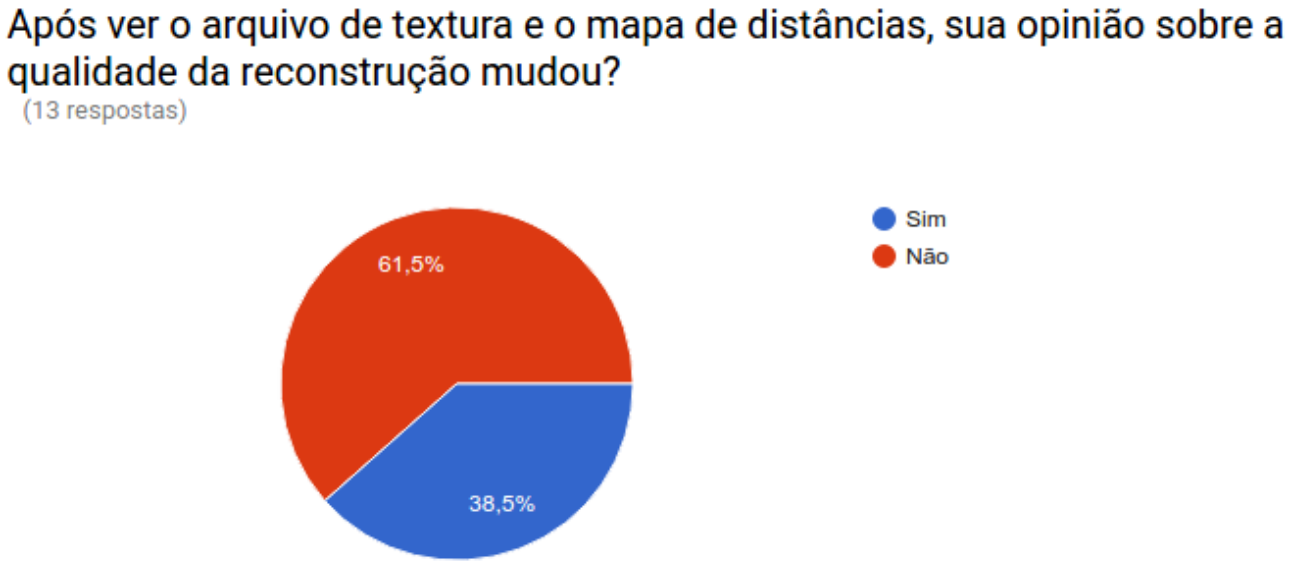


Figura 21 a: Gráfico do percentual de voluntários que, ao se depararem com o mapa de distâncias proveniente da avaliação automática, mudaram de ideia quanto a qualidade da reconstrução de seus testes.

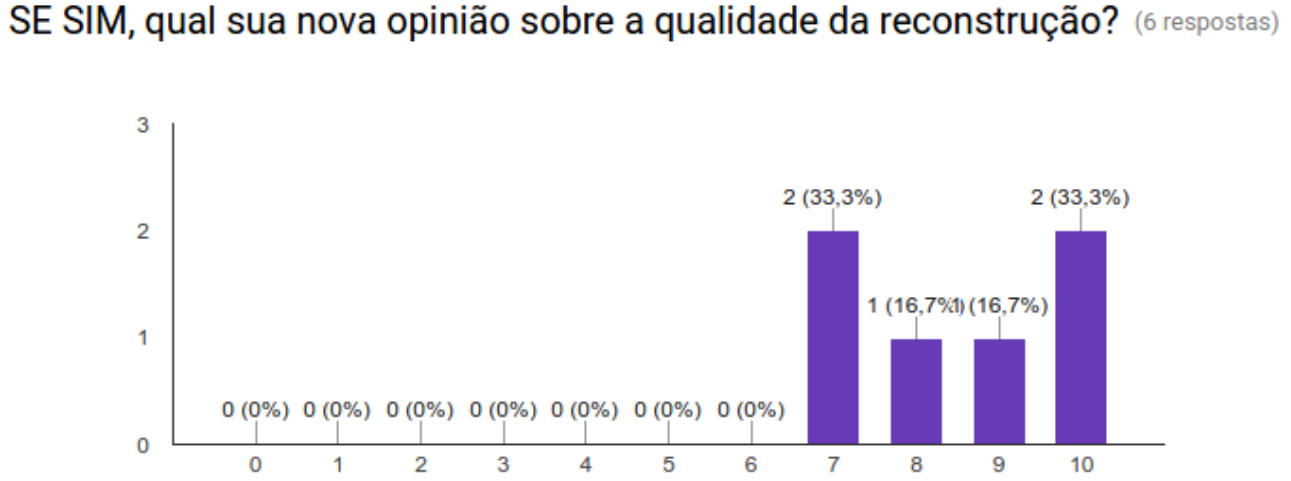


Figura 21 b: Novas respostas dos voluntários que mudaram de ideia quanto a qualidade de seus testes.

Também podemos observar que um percentual considerável de voluntários mudou de ideia quanto a qualidade da sua reconstrução após se depararem com o mapa de distâncias e com o arquivo de textura usado para criação do modelo texturizado. Ou seja, antes de entrarem em contato com a avaliação automática de qualidade de reconstrução, estes voluntários tinham uma opinião sobre a qualidade de sua reconstrução, e após isso – possivelmente por perceberem que sua avaliação de qualidade não batia com a avaliação automática – resolveram aumentar ou diminuir suas notas. Além disso, ao avaliarmos a média destas notas, chagamos ao valor de 8.5 pontos, um valor um pouco acima da média encontrada nas duas primeiras questões.

Além dos experimentos realizados e apresentados por este trabalho através de imagens, há um outro experimento realizado e gravado em vídeo, que pode ser visualizado no endereço https://youtu.be/2L9SC_BQ4-Q.

# 5 Conclusão

Como proposto inicialmente, este trabalho apresenta um protótipo de sistema que se dispõe a construir um modelo facial 3D a partir de nuvens de pontos capturadas do sensor Kinect. Este sistema permite que usuários comuns, sem nenhum conhecimento avançado de operação de sistemas gráficos, possam utilizá-lo sem maiores problemas e em pouco tempo, e assim podendo criar seus próprios avatares 3D.

Utilizar os métodos de registro de nuvens de pontos foi fundamental para este processo, sendo essenciais na etapa de criação da nuvem cheia e consequentemente, na etapa de preenchimento da imagem de textura, sendo possível procurar pontos próximos de uma nuvem mais densa, formando assim imagens de textura de qualidade.

## 5.1 Trabalhos Futuros

Como o sistema apresentado neste trabalho é um protótipo, melhorias e extensões podem ser feitas após sua conclusão. Alguns exemplos são:

- Realizar integração entre este trabalho e o sistema de alinhamento e deformação do modelo médio, que até o momento, como são sistemas completamente separados, está sendo feito apenas por vias de chamadas de sistema operacional;
- Construir um esquema de testes automático para descobrir, e avisar sobre, uma distância mínima, ou ideal, entre o sensor e o usuário;
- Além de otimizações possíveis no código-fonte, a fim de evitar gastos computacionais desnecessários.

Em se tratando de possíveis extensões (ou outros usos de todo ou parte deste sistema), já foram propostas algumas ideias no capítulo 1, como simulações de envelhecimento facial e produção visual. Há também formas mais lúdicas de se aproveitar este trabalho – tendo em vista que este trabalho é focado apenas em conjuntos de dados com expressões neutras – como jogos digitais que trabalham com animações faciais que seguem as expressões feitas pelos próprios usuários. Além, é claro, do possível uso em jogos online, onde cada jogador poderia ter seu próprio avatar mais próximo de sua face real, facilitando a diferenciação e unicidade de cada um.

# Referências


1. Zollhöfer M., Martinek M., Greiner G., Stamminger M., Süßmuth J. “Automatic reconstruction of personalized avatars from 3D face scans”. Journal of Visualization and Computer Animation, Volume 22, 195-202 (2011)
2. Ashburner J., Friston K. “Human Brain Function”, Part II, Section 1, Chapter 2 (2004)
3. Celes W., Cerqueira R., Rangel J. L. “Introdução a Estruturas de Dados” (2004)
4. Foley J. D., van Dam A., Feiner S. K., Hughes J. F. “Computer Graphics: Principles and Practice in C” (1995)
5. Bussab W. O., Morettin P. A. “Estatítica Básica”. 5ª Edição (2002)
6. Yin L., Wei X., Sun Y., Wang J., Rosato M. “A 3D Facial Expression Database For Facial Behavior Research”. International Conference on Automatic Face and Gesture Recognition, 211-216 (2006)
7. Brunton A., Salazar A., Bolkart T., Wuhrer S. “Review of Statistical Shape Spaces for 3D Data with Comparative Analysis for Human Faces”. Computer Vision and Image Understanding, 128:1-17 (2014)
8. Tang L. A., Huang T. “Automatic construction of 3D human face models based on 2D images”. Proceedings of Image Processing’96, pages 467–470 (1996)
9. Kemelmacher-Shlizerman I., Basri R. “3D Face Reconstruction from a Single Image using a Single Reference Face Shape”. IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (2) 394–405 (2011)
10. Breuer P., Kim K. I., Kienzle W., Blanz V., Schölkopf B. “Automatic 3D Face Reconstruction from Single Images or Video”. Automatic Face Gesture Recognition (FG'08). 8th IEEE International Conference, pp. 1–8 (2008)
11. Li H., Sumner R. W., Pauly M. “Global Correspondence Optimization for Non-Rigid Registration of Depth Scans”. Eurographics Symposium on Geometry Processing, Volume 27, Number 5 (2008)
12. Scheidegger C. E., Fleishman S., Silva C. T. “Triangulating Point Set Surfaces with Bounded Error”. Proceedings of the third Eurographics symposium on Geometry processing (SGP '05), Article 63 (2005)
13. Saragih J. M., Lucey S., Cohn J. F. “Face Alignment through Subspace Constrained Mean-Shifts”. International Conference of Computer Vision (ICCV) (September, 2009)
14. Mullen, T. “Mastering Blender”. 1st ed. Indianapolis, Indiana: Wiley Publishing, Inc. (2009)
15. Murdock, K. L. “3ds Max 2009 Bible”. 1st ed. Indianapolis, Indiana: Wiley Publishing, Inc. (2008)
16. Rosenblatt, M. "Remarks on Some Nonparametric Estimates of a Density Function". The Annals of Mathematical Statistics, Volume 27, Issue 3, 832-837 (1956)

17. Cheng, Y. "Mean Shift, Mode Seeking, and Clustering". IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 17, Issue 8, 790-799 (August 1995)

18. Fitzgibbon, A. W. "Robust registration of 2D and 3D point sets". Image and Vision Computing Volume 21, Issue 13, 1145-1153 (2003)

19. Zhang, Z. "Iterative point matching for registration of free-form curves and surfaces". International Journal of Computer Vision (Springer) 13 (12): 119–152 (1994)

20. Pomerleau F., Colas F., Siegwart R. "A Review of Point Cloud Registration Algorithms for Mobile Robotics". Foundations and Trends in Robotics 4 (1): 1–104 (2015)

21. Besl P. J., McKay N.D. "A Method for Registration of 3-D Shapes". IEEE Trans. on Pattern Analysis and Machine Intelligence (Los Alamitos, CA, USA: IEEE Computer Society) 14 (2): 239–256 (1992)

22. Li H., Weise T., Pauly M. “Example-based facial rigging”. ACM Trans. Graph. 29 (4) 32:1–6 (2010)

23. Reddy M. “Obj Specification as used by Wavefront”. Acessado em 16/03/2016 em http://www.martinreddy.net/gfx/3d/OBJ.spec

24. Bourke P. “MTL material format (Lightwave, OBJ)”. Acessado em 16/03/2016 em http://paulbourke.net/dataformats/mtl/

25. Microsoft. “Kinect for Windows Sensor Components and Specifications”. Acessado em 17/03/2016 em https://msdn.microsoft.com/en-us/library/jj131033.aspx

26. Möbius A. F. “The Barycentric Calculus” (1827)

27. Bentley J. L. “Multidimensional binary search trees used for associative searching”. Communications of the ACM 18 (9): 509 (1975)

28. Rusu R. B., Cousins S. “3D is here: Point Cloud Library (PCL)”. IEEE International Conference on Robotics and Automation (ICRA) (2011)

29. OpenCV. “About OpenCV”. Acessado em 11/04/2016 em http://opencv.org/about.html

30. Lopes E. C. “Detecção de Faces e Características Faciais”. Trabalho Individual II. Programa de Pós-Graduação em Ciência da Computação. Pontifícia Universidade Católica do Rio Grande do Sul (2003)

31. Zhang C., Zhang Z. “Boosting-Based Face Detection and Adaptation” (2010)

32. Burrus N. “Kinect RGB Demo v0.5.0”. Acessado em 14/04/2016 em http://nicolas.burrus.name/index.php/Research/KinectRgbDemoV5?from=Research.KinectRgbDemoV4

33. Cootes T. F., Taylor C. J., Cooper D. H., Graham J. "Active shape models - their training and application". Computer Vision and Image Understanding (61): 38–59 (1995)