RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Uni-FACEF Centro Universitário Municipal de Franca
Vol. 13 Nro. 1 2023

PHOTOFACER3D: um protótipo de software para criação de um modelo de rosto 3D com reconhecimento facial

Lucas Amaral Oliveira
Graduado em Engenharia de Software – Uni-FACEF
lucas.amaral.olivera@gmail.com

Nathan Mendes Maturano
Graduado em Engenharia de Software – Uni-FACEF
nathanmedesn2m@gmail.com

Daniel Facciolo Pires
Docente do Uni-FACEF
daniel@facef.br

Resumo
O reconhecimento facial tem como fundamento a definição dos pontos nodais(pontos de marcas faciais) para facilitar na modelagem 3D, que por sua vez reconhece estes pontos e modela o avatar de uma forma mais exata. Este trabalho tem como objetivo criar um protótipo de software para a modelagem de um avatar que poderá ser utilizado em diferentes aplicações, tais como jogos, filmes e dublês digitais. A metodologia utilizada para a execução do trabalho consiste na criação de documentos presentes na engenharia de software, bem como na implementação dos procedimentos da modelagem 3D, tais como a definição dos contornos do rosto, os pontos nodais, para então a criação do avatar 3D. Como resultados, obteve-se um modelo 3D, porém em poucos polígonos. O projeto permite que usuários possam poupar tempo na construção de avatares, e os encoraja a utilizar a solução proposta em seu dia a dia.

Palavras-chave: Modelagem 3D, Reconhecimento Facial, Visão Computacional, Pontos Nodais.

Abstract
Facial recognition is important for defining nodal points that aid in 3D modeling, which in turn recognizes these points and models the avatar more accurately. The goal of this work is to model an avatar that can be used in various applications such as games, movies, and digital doubles. The methodology used for performing this task involves capturing images, after which the prototype proceeds to the modeling procedures, including defining facial contours and nodal points, before moving on to the 3D avatar. The procedures resulted in a 3D model, albeit with few polygons. The project successfully achieved simplified and efficient 3D modeling, saving users a significant amount of time and encouraging them to use the solution in their daily lives.

Keywords: *3D Modeling. Face Recognition. Computer Vision. Nodal Points.*

1 Introdução

É notável a crescente necessidade do uso e aplicação do reconhecimento facial para diversos fins, como a detecção de fraudes, a segurança cibernética, os aeroportos e controle de fronteiras, os serviços bancários, a área de saúde, entre outros. Particularmente, a cidade dos autores deste trabalho, Franca-SP, tem um time de basquete local. Em dias de jogos no ginásio municipal, a entrada dos sócios-torcedores é realizada por reconhecimento facial.

Os avatares desempenham um papel importante na identificação, personalização, privacidade, expressão criativa, representação, comunicação, *branding* e entretenimento em ambientes digitais. Eles permitem a criação de representações realistas, interativas e envolventes de rostos humanos e personagens, contribuindo para diversas áreas, tais como, da arte, ciência e tecnologia.

Como problema de pesquisa, questiona-se se há dificuldades iniciais na utilização do reconhecimento facial e da modelagem 3D para a criação de avatares. Em seguida propor a criação de um protótipo de software para modelagem de um avatar a partir do uso de reconhecimento facial, que gera modelos em 3D.

Para o uso da fotogrametria em modelagem 3D, foram utilizadas diversas tecnologias, como por exemplo, a ferramenta *RStudio* foi utilizada, tendo como base a foto de um dos desenvolvedores. Na expectativa de se obter um modelo 3D partindo das fotos utilizadas, alguns obstáculos foram encontrados, sendo discutidos posteriormente neste artigo.

Para um melhor entendimento do problema foi realizada uma pesquisa sobre as tecnologias que estão em atividade atualmente, como Revopoint, Polycam, Keentools, entre outros. Após a pesquisa foram definidas as estratégias de desenvolvimentos e testes do protótipo.

Metodologicamente, este trabalho inicia com uma revisão teórica, exploratória, bibliográfica, de temas na área de reconhecimento facial, modelagem 3D, visão computacional e exemplos de aplicações existentes atualmente. Em seguida, foram desenvolvidos alguns documentos da área de engenharia de software, como o de requisitos, os diagramas da UML, Caso de Uso e Máquina de Estado, bem como os *backlogs* da metodologia *Scrum*, e ainda o diagrama *BPMN*.

2 Referencial Teórico

Nessa seção serão vistos conceitos essenciais para a construção do projeto de software e uma pesquisa de mercado para analisar métodos que soluções semelhantes utilizam para geração de malha automatizada.

2.1 Representação Digital Tridimensional

Um modelo tridimensional, também conhecido como modelo 3D, é uma representação digital de um objeto ou cena em três dimensões. Ou seja, é uma representação computacional que descreve um objeto em três eixos: altura, largura e profundidade (ROYRIGSS, 2011).

Os modelos 3D são amplamente utilizados em diversas áreas, como arquitetura, engenharia, design de produtos, animação, jogos, entre outras. Eles podem ser criados a partir de softwares de modelagem 3D, escaneamento 3D ou outras técnicas de captura de dados.

Uma das formas de salvar um modelo 3D mais popular é o formato .obj, que é um formato de arquivo utilizado para armazenar modelos tridimensionais (3D) em computação gráfica. Esses arquivos geralmente contêm informações sobre a geometria, texturas, texturas normais e outras propriedades de um modelo 3D (ROYRIGSS, 2011).

Os arquivos .obj foram criados em 1980 e atualmente podem ser abertos e editados em softwares de modelagem 3D, sendo suportados por uma grande variedade de programas de computador, tanto comerciais Maya (ALIAS, 1998), 3ds Max (AUTODESK, 1996) quanto gratuitos, como Blender (TON, 1994) , entre outros.

Explicando em mais detalhes, .obj nada mais é que um dicionário interpretado pelo computador, como vemos na Figura 1, na qual o ‘v’ representa um vértice e sua posição; ‘vt’ representa a textura/a cor do vértice, ‘vn’ o mapa normal do vértice, que é basicamente como que aquele vértice deve reagir a luz, ‘g’ que representa o grupo, a malha na qual se encontram os vértices (um obj pode ter mais de um grupo) e ‘f’ representa uma face, que nesse caso seria um polígono triangular, utilizando os vértices (-147,-152 , -155), com suas coordenadas X, Y, Z; e suas respectivas texturas e normals (Como o vértice deve reagir a luz) para construí-lo.

Figura 1 - arquivo .obj aberto no bloco de notas

```
f -147/-147/-147 -152/-152/-152 -155/-155/-155
g Model001_22_head_0.5_75_75_head_head neck upper
usemtl Model001_Material003
v 0.01841415 1.792252 0.09704287
vn 0.2231895 -0.1253395 0.2740371
vt 0.422607 1.060547
```

Fonte- SEA206,2019

Por esta razão, optou-se por utilizar a extensão .obj neste artigo, sendo o software Blender escolhido para a abertura e visualização destes arquivos.

2.2 Visão computacional

Visão computacional é uma área de estudo da ciência da computação que lida com a interpretação de informações visuais a partir de dados de imagens ou vídeos. Utilizando de técnicas de geometria, transformação e processamento de imagens e estimação de movimento, ela consegue reconhecer objetos desde rostos numa foto até montar um objeto num espaço 3D (SZELISKI,2011).

Isso por causa de padrões geométricos que a máquina nota nas imagens providas a ela. Razões como essa tornam a visão computacional um atrativo para trabalhar com segurança por meio da biometria, já que um bom sistema pode diferenciar rostos e dizer a quem ele pertence, utilizando 80 pontos nodais do rosto, pontos específicos da face humana, para medir e identificar diferenças, aplicando então o reconhecimento facial (ZHANG, 2017).

2.3 Reconhecimento facial

Reconhecimento facial é mais do que detecção facial. A detecção diz respeito apenas a dizer se tem um rosto naquela foto e onde está, enquanto o reconhecimento analisa os detalhes do rosto para determinar uma pessoa a partir de uma imagem digital ou de um vídeo (ORVALHO, 2019).

Reconhecer faces é uma tecnologia que permite a melhoria de sistemas de segurança, seja por ser utilizada como verificação de pessoas, quanto por identificar emoções e padrões para evitar crimes(KASPERSKY, 2023).

Isso significa que o reconhecimento pode ser atrapalhado pela qualidade da câmera, iluminação do rosto e orientação da cabeça, precisando que o usuário se ajuste ou realize mais fotos em outros ângulos para compensar.

Mas mesmo com suas limitações, a capacidade do reconhecimento facial de coletar pontos nodais (pontos de marcas faciais) e utilizá-los para distinguir pessoas o torna um atrativo para principalmente a área de segurança, auxiliando a biometria.

Existem vários pontos nodais que podem ser utilizados para distinguir pessoas, com algumas distinções entre especialistas de quais são essenciais. Ping Lou define, por exemplo, 68 pontos. Dentre eles, vale ressaltar: ponto do queixo, da maçã do rosto, subnasal e os cantos externos e internos do olho.(ZHANG, LUO, LOY, TANG, 2017)

Como esses pontos são uma constância para todo rosto, se o protótipo for capaz de reconhecê-los, poderia recontextualizá-los para medir a altura do sujeito, semelhante às técnicas de fotogrametria.

2.4 Fotogrametria

Ao tirar múltiplas fotos em diferentes ângulos, podemos determinar com mais clareza as dimensões do sujeito das fotos, conseguindo informações como altura, largura e comprimento. Aplicando essa escalabilidade num ambiente 3D e texturizando o modelo gerado é o que chamamos de fotogrametria. Em pessoas é aplicada mais na área de medicina, sendo mais popular para determinar o tamanho de uma área geográfica (ADENILSON, 2018).

A definição de Fotogrametria a proposta pela *American Society of Photogrammetry* em 1979, como sendo: "*Fotogrametria é a arte, ciência e tecnologia de obtenção de informação confiável sobre objetos físicos ambiente através de processos de gravação, medição e interpretação de e por meio imagens fotográficas e padrões de energia eletromagnética radiante e outras fontes*".

Por suas características, essa técnica que originalmente foi criada para estudos geológicos já foi exportada para diversas áreas, como criação de Projetos de estradas, Arqueologia, Robótica, Digitalização de apartamentos para vendê-los, e até medicina.

Apesar de ser ótima para construir réplicas virtuais didáticas de órgãos humanos, ela não é adequada para criar uma pessoa 3D, pelo menos quando se utiliza poucas câmeras.

Isso porque fotogrametria utiliza a semelhança das fotos para determinar o tamanho do objeto, mas com qualquer variação ou inconsistência causadas por respirar já pode desalinhar o processo, gerando um modelo 3D incorreto, o que é especialmente relevante quando se utiliza apenas uma câmera, uma vez que a quantidade ideal de fotos recomendada para este processo é de 50 (3DFLOW, 2017).

2.5 *Scanners 3D*

O Scanner 3D é um hardware que utiliza recursos de câmeras de alta resolução para captar imagens. Para auxiliar na captura mais precisa destas imagens podem ser utilizados recursos como luz estruturada e feixe de laser. Estas duas formas citadas anteriormente possuem a função de facilitar na captura e envio de dados em imagem (PRINTIT, 2021).

Um bom *scanner* pode poupar tempo de trabalhos indesejados com redimensionamentos ou correções indesejadas. Respeitando o versionamento e as normas de cada aparelho, o escaneamento obtido será mais preciso.

2.6 Aplicativos similares à proposta do trabalho

Nessa seção serão abordadas as principais técnicas e ferramentas utilizadas pela concorrência, para realizar tarefas semelhantes, que normalmente podem ser categorizados como softwares de fotogrametria ou *scanner* 3d.

2.6.1 Revopoint

Revopoint é uma companhia focada em escaneamento 3d. Seu último produto, Revo RANGE, permite escanear objetos grandes de perto (até 800 milímetros) o que permite uma precisão melhor que outros *scanners* profissionais, especialmente para pessoas (REVOPOINT, 2023).

Ele é pequeno e possui softwares e compatibilidade com o celular. Utiliza duas lentes para tirar fotos e dizer a distância dos objetos baseado nas diferenças entre as duas imagens.

Ainda sim ele não possui nenhum processo de otimização extra para pessoas é feito, e precisa de um hardware extra para fazer.

2.6.2 ZKTeco

Outro *Scanner 3d*, O ZKTeco se diferencia do antecessor pela sua tecnologia de Luz Estruturada, que permite aplicar o reconhecimento facial em 3D. Essa tecnologia permite que o brilho do ambiente e outros fatores externos sejam desconsiderados pelo sistema, já que ele possui 30.000 pontos laser na face do usuário (ZKTECO, 2021).

Apesar disso, ele não é compatível com a proposta do projeto, esse hardware é utilizado para segurança biométrica, e nesse quesito ele é surpreendente, já que ele consegue distinguir fotos de pessoas e pessoas de verdade com sua tecnologia de infravermelho.

Apesar de criar um modelo 3D e poder acessá-lo para conferir se é o mesmo usuário, por questões de segurança não é permitido baixar o modelo e modificar, já que se ele estiver diferente da pessoa; ou essa pessoa vai ser barrada pelo sistema ou o modelo vai ser revertido baseado na pessoa real.

2.6.3 Large Pose 3D Face Reconstruction

Software desenvolvido por estudantes britânicos, esse programa consegue criar um modelo 3D de um rosto utilizando apenas uma única foto. Para isso, eles criaram uma *Convolutional Neural Network* (*Rede Neural Convolucional* ou *CNN)* e passaram várias fotos 2D e seus respectivos modelos 3D, treinando a AI (AARON, 2017).

O estudante Aaaron S. Jackson se empenhou na criação de um sistema que utiliza marcas faciais para criar a malha.“*Mostramos como a tarefa relacionada da localização de marcos faciais 3D pode ser incorporada à estrutura proposta e ajudar a melhorar a qualidade da reconstrução*”[..] “*Demonstramos que nossa CNN trabalha com apenas uma única Imagem facial 2D*”

Ela foi criada em 2017 com o propósito de reconstrução facial para usos de investigação forense, e se desempenhou melhor que 3DDFA e EOS, softwares que estavam sendo comparados no estudo.

Existia uma demo online do software, mas infelizmente o projeto foi descontinuado recentemente e sua última versão contém problemas e especificações que o tornam menos acessíveis e impreciso.

2.6.4 Meshroom

Meshroom permite fazer modelos 3D por meio de uma série de fotos, como outros aplicativos de fotogrametria. Seu diferencial se dá por ser um aplicativo completamente gratuito, e por permitir configurar cada passo do processo, desde como as fotos serão processadas até como a malha será renderizada (ALICE, 2022).

Ela é semelhante ao *scanner*, mas considerando que o melhor jeito de gerar um modelo humano assim é tirar múltiplas fotos ao mesmo tempo, é mais acessível ter várias câmeras do que ter vários *scanners*.

Isso o torna um sistema bastante complexo, que pode punir iniciantes, já que fica por conta do usuário definir a ordem do processo, podendo gerar em ordem incoerreta. Usando um sistema de ponto de nuvens, ela une os pontos conseguidos por meio das fotos em uma única malha.

2.6.5 PolyCam

Usando tecnologia de LIDaR, Polycam consegue transformar uma série de fotos em um objeto 3d. Isso tudo usando apenas a câmera do celular e processando nele, o que o torna bem fácil de utilizar e acessível (POLYCAM, 2023).

O aplicativo é mais tratado como uma rede social, você posta os modelos criados com ele online e as pessoas reagem, possivelmente servindo como portfólio. Mesmo você criando o modelo, para você baixar e utilizá-lo é necessário pagar por uma conta premium.

2.6.6 Keentools Facebuilder

Esse é possivelmente o aplicativo mais próximo do nosso conceito, Facebuilder é um *addon open source* para o Blender, criado pela empresa Kentools, que permite a criação de rostos foto realistas para uso profissional (KEENTOOLS, 2023).

Ele não cria um modelo do zero, na verdade fotos são importadas para o Blender e o próprio usuário ajusta o modelo para ficar enquadrado com as fotos e, eventualmente, com a utilização de várias fotos boas, o modelo fica mais e mais parecido com o rosto que estava sendo replicado.

Uma inteligência artificial está sendo desenvolvida para ajustar as fotos automaticamente, mas esta possui uma baixa precisão. Então o fato de ter que combinar várias fotos manualmente combinado com uma licença cara pode ser vista como um negativo.

2.6.7 Estratégia de Desenvolvimento

Para melhor visualização dos diferenciais de cada solução, um quadro com os pontos únicos de cada software foi feito, visto na Figura 2.

Figura 2 - Tabela de concorrência

	3D Face Reconstruction	Polycam	ZKTECO	Facebuilder	Revopoint	Meshroom
Minimo de Fotos	1	15	5	12	10	30
Metodo	Inteligencia Artificial	Fotogrametria	Scanner	Manual	Scanner	SemiManual/Fotogrametria
Otimização	Rosto Humano	Nenhum	Rosto Humano	Rosto Humano	Proximidade	Nenhum
Hardware	Computador	Celular	Scanner	Computador + Cameras	Scanner + Competador/Celular	Computador + Cameras
Orçamento	Gratuito	Gratuito	R$ 5.443,99	US$499.00	US$799.00	Gratuito
Situação	Descontinuado	Ativo	Ativo	Ativo	Ativo	Ativo
Objetivo	Recontrução Forencica	Digitalizar Objetos	Biometria	Rostos Proficionais	Digitalizar Objetos	Digitalizar Objetos

Fonte - Autores

De posse dessas informações, foi possível entender o mercado e repensar na forma de desenvolver o protótipo. Devido a sua capacidade de gerar um modelo com uma quantidade pequena de fotos, o protótipo utilizará de inteligência artificial como metodologia, com objetivos de gerar rostos humanos realistas.

3 Resultados da Documentação do Protótipo e da Modelagem 3D

3.1 Documentação do Protótipo

A documentação possui uma importância fundamental para o bom desenvolvimento, já que com esta podemos entender os requisitos do projeto, com o Documento de Requisitos, o backlog é importante para saber distribuir as tarefas e medir suas prioridades para o bom andamento do desenvolvimento.

O Documento de Requisitos, visto na Figura 3 foi o ponto de partida para o desenvolvimento do projeto e sua prototipação, pois neste documento são citados os requisitos que compõem o projeto. Um dos principais requisitos é a capacidade de criar modelos a partir de fotos, bem como atualizá-los se necessário ou possível, caso for de interesse do usuário.

O diagrama de Caso de Uso, na Figura 4, mostra as etapas do processo sistêmico de forma clara e detalhada. Ele diz como os atores, nesse caso o usuário, se relaciona com cada requisito do sistema.

Por fim, o BPMN, ou *Business Process Model and Notation*, ajuda a organizar os processos do sistema (TOTVS, 2023). Exemplificado na Figura 5, apresenta uma representação visual que esclarece os papéis desempenhados por cada ator durante a operação do sistema em questão, além das suas relações entre as atividades de diferentes atores.

Isso proporciona uma visão mais abrangente e compreensível da dinâmica de interação entre os diversos participantes no contexto do processo. Essa documentação pode ser verificada no GitHub (OLIVEIRA E MATURANO, 2023).

Figura 3 - Requisitos do sistema

[RF002] Criar Malha

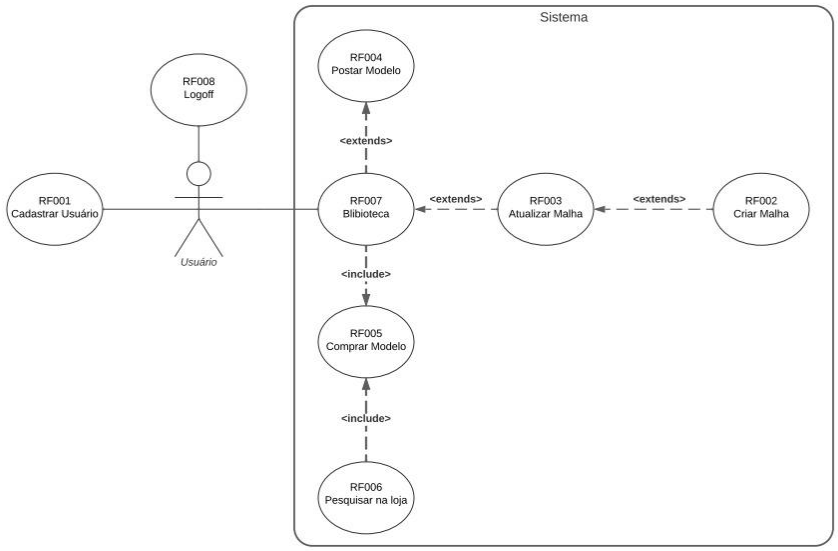
Identificação:	[RF002] Criar malha
Descrição:	Cria uma malha 3d do rosto que for tirado foto.
Prioridade:	þ Essencial
Regra de Negócio:	Não Possui.

[RF003] Atualizar Malha

Identificação:	[RF003] Atualizar Malha
Descrição:	Permite modificar a malha criada para parecer mais com a forma desejada.
Prioridade:	þ Essencial
Regra de Negócio:	Não Possui.

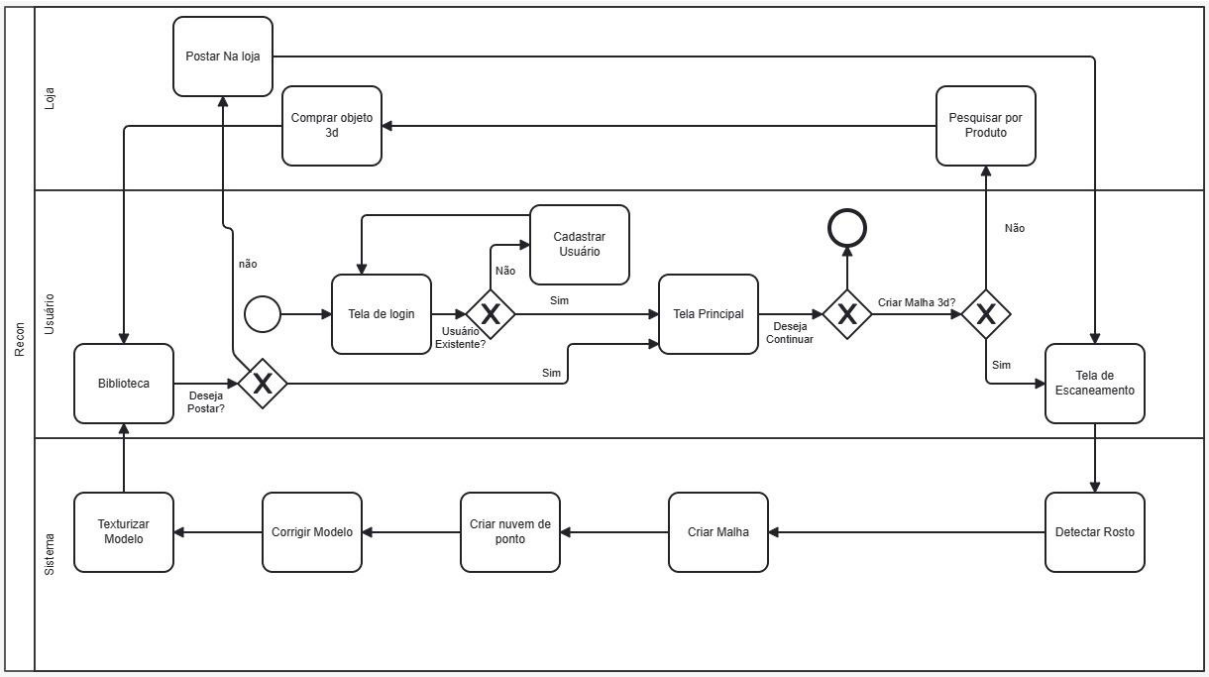
Fontes - Autores

Figura 4 - Diagrama de caso de Uso



Fontes - Autores

Figura 5 - BPMN



Fontes – Autores

3.2 Modelagem 3D

Com o escopo do projeto definido, começou a fase de desenvolvimento. O projeto foi dividido em duas partes, a primeira em gerar um modelo *lowpoly* utilizando apenas as características do rosto do usuário e a segunda sendo um modelo mais detalhado utilizando o primeiro para correções.

3.2.1 Ferramentas

Para a progressão desse protótipo, foi necessário a utilização de alguns softwares externos.

Todos os arquivos criados foram salvos no Google Drive e no OneDrive. Para controle de versões entre os desenvolvedores, o Github foi de grande ajuda.

A criação de diagramas foi feita no lucidchart, e arquivos de textos utilizaram LibreOffice para sua realização. O software Rstudio otimizou a programação em R. Blender foi utilizado no pós-processamento do modelo.

3.2.2 Linguagem

Primeiramente procuramos por linguagens que possuam uma biblioteca de reconhecimento facial,sendo elas, *Python*, *R*, *Javascript*.

Nossa primeira escolha foi *Python* com sua biblioteca *OpenCV* , ele era pioneiro nas pesquisas, sua precisão era um dos maiores atrativos e poderia ser escolhido quais pontos nodais seriam utilizados para o reconhecimento e a posição deles era salva. Apesar disso, a versão disponível da biblioteca *OpenCV* não era compatível com nenhuma versão disponível de *Python*.

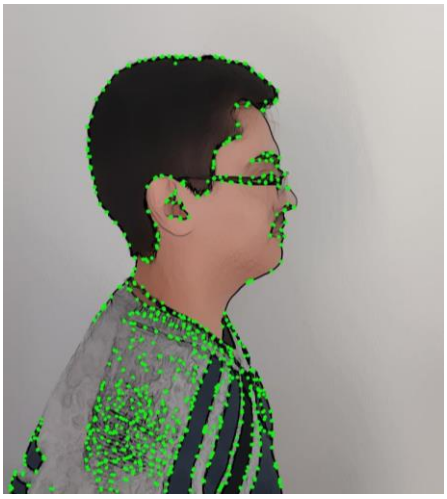
Procuramos então por uma versão em *Javascript*, já que seria fácil de implementar para multiplataformas e seria uma opção leve. Um *bug* com o protótipo fez com que a função *canva,* que supostamente só iria marcar os pontos nodais, ocupasse a tela inteira em vez de só marcar os pontos e sobrepor a câmera.

Paralelamente pesquisamos também por uma biblioteca na linguagem R, já que essa língua é bastante semelhante a *Python*, e nela encontramos outro *OpenCV*. Esse era capaz de reconhecer o rosto, e não apresentou muitos erros, então optamos por utilizá-lo.

3.2.3 Limitações

Apesar da biblioteca ser funcional, ainda havia problemas. Ela possui duas funções para retornar pontos, mas como podemos ver na Figura 6, a primeira função não retornava pontos referentes ao rosto humano, tratavam de objetos como um óculos, a blusa da pessoa o algoritmo encontrava variação no anglo da silhueta maior que o valor dado (nesse caso de 25°).

Figura 6- Pontos da Imagem



Fonte – Autores

A segunda função não podia ser modificada, retornando pontos que a máquina considerou importante, podendo ou não ser parte do rosto.

O reconhecimento facial também não foi ideal. Ele não retorna pontos importantes, como a boca e o nariz, e mesmo pontos de referência que ele reconhecia, como o olho, não retornava a localização do mesmo.

Sendo assim, as únicas informações providenciadas pela biblioteca foram o centro do rosto e a distância do queixo até o final da testa, ficando a nosso cargo descobrir a localização de mais referências.

3.3 Desenvolvimento

Nessa seção será retratado o código utilizado para a criação do protótipo desenvolvido, mostrando como as aplicações foram utilizadas para realizar determinada tarefa.

3.3.1 Entradas

O usuário só necessita de fornecer 2 imagens, uma de frente e outra de lado, para que o sistema possa realizar o processo inteiro. Para nossos testes, foram usadas imagens que já estavam salvas no computador, mas o programa tem tanto a opção de utilizar um arquivo salvo quanto a de tirar uma nova foto no momento e utilizá-la, conforme ilustrado na Figura 7.

Figura 7 - Entradas do Usuário

```
rosto1<-ocv_read("C:/Users/arnal/OneDrive/Imagens/frente.png")
rosto2<- ocv_picture()
```

Fonte - Autores

3.3.2 Tratamento de imagens

Para que o protótipo tenha uma maior precisão e consiga retornar os pontos necessários, será preciso que uma cópia das imagens fornecidas sejam alteradas para atender as necessidades.

A Figura 8 exemplifica os principais filtros: o filtro de estilização, utilizado para aumentar o contraste, detecção de borda que retorna todas as bordas encontradas em branco e o vazio entre elas como nada e a detecção de rosto, utilizada para descobrir o tamanho do rosto e a posição dele na imagem.

Figura 8 - Tratamento de imagem

```
contraste1<- ocv_stylize(rosto1)
ocv_write(contraste1,"C:/Users/arnal/OneDrive/Imagens/seila.jpg")
dark1<-ocv_edges(contraste1)
ocv_write(dark1,"C:/Users/arnal/OneDrive/Imagens/Face.jpg")

circulo1<-ocv_face(contraste1)
```

Fonte - Autores

3.3.3 Detecção de pontos

Pontos que não foram retornados na detecção facial e que são essenciais para o rosto humano são encontrados utilizando uma série de técnicas.

Um exemplo está na Figura 9. Neste caso, mesmo não retornando a localização dos olhos, ele reconheceu que existia um ali e marcou na imagem, então foi possível determinar o centro do olho com essa marcação.

Além da detecção, é possível que o sistema retorne os pontos com uma certa imprecisão. Utilizando a imagem das bordas, podemos pegar o pixel branco mais perto daquele ponto para aumentar a precisão, que foi feito na Figura 10 para corrigir as orelhas.

Os pontos da face de lado precisam de um cuidado especial, já que a detecção de face não funciona de lado. Mas o nariz e o topo da cabeça podem ser encontrados por procurar respectivamente o pixel branco mais ao lado e mais alto.

Figura 9 - Centro do Olho

```
raiox<-0
raioy<-0
for (x in esqLong1:dirLong1) {
  for(y in topoy1:baixoy1){
    cor<-color.at(olho, x, y,z=1)
    raioy<- x-centrox
    raiox<- y-centroy
      if((cor[1]=0.1)&&(cor[2]<=azul[2])&&(cor[2]<=0.3)&&((0.65<=cor[3])&&(cor[3]<=azul[3]))){
        if(sqrt((raiox*raiox)+(raioy*raioy))<(raio-50)){
          olho_vetx<-append(olho_vetx,x)
          olho_vety<-append(olho_vety,y)
          print(cat(x,y))
```

Fonte - Autores

Figura 10 - Ajustar Orelhas

```
#encontra o primeiro valor que não é preto, nesse caso é uma orelha
for (x in esqLong1:centrox) {
  cor<-color.at(img, x, centroy,z=1)
  if(cor!=0){
    esqLong1<-x
    break
  }
}
#encontra o primeiro valor que não é preto, nesse caso é uma orelha
for (x in dirLong1:centrox) {
  cor<-color.at(img, x, centroy,z=1)
  if(cor!=0){
    dirLong1<-x
    break
  }
}
```

Fonte - Autores

Com esses pontos encontrados, os pontos restantes podem ser encontrados utilizando essas duas referência, e os pontos encontrados na imagem de frente, levando em conta a distância entre eles estabelecida.

3.3.4 Geração do Modelo

Após a definição de todos os pontos, são utilizados os pontos X e Y retirados da imagem de frente como o X e o Y do vértice.

Para escrever o Z do vértice, pode ser utilizado o X de um dos pontos encontrados na imagem de lado, desde que ele seja o mesmo ponto encontrado na imagem de frente, ilustrado na Figura 11.

Figura 11- Vértice do Modelo

```
#escrever o modelo
sink("C:/Users/arnal/OneDrive/Imagens/Rosto.obj")
#queixo
cat(c('v', (centrox/100), (orelha_topo_y/-100),(min(topoz)/100),'\n'))#1
cat(c('v', (centrox/100),(baixoy1/-100),(max(baixoz)/100),'\n'))#2
#testa centro(vou ter que fazer testa lado depois)
cat(c('v', (centrox/100),(topoy1/-100),(min(topoz)/100),'\n'))#3
cat(c('v', (centrox/100),(topoy1/-100),(max(topoz)/100),'\n'))#4
#estremidades laterais
cat(c('v', (esqLong1+15/100), (centroy/-100),(median(centroz)/100),'\n'))#5
cat(c('v', (dirLong1-15/100),(centroy/-100),(median(centroz)/100),'\n'))#6
#nariz
cat(c('v', (centrox/100),(nariz_y/-100),(nariz_z/100),'\n'))#7
```

Fonte - Autores

É recomendado anotar qual vértice pertence a qual lugar para a geração de faces, visto que a posição dele no arquivo vai ser usado para definir qual vértice pertence a qual face. Um exemplo se encontra na Figura 12.

Figura 12 - Faces do Modelo

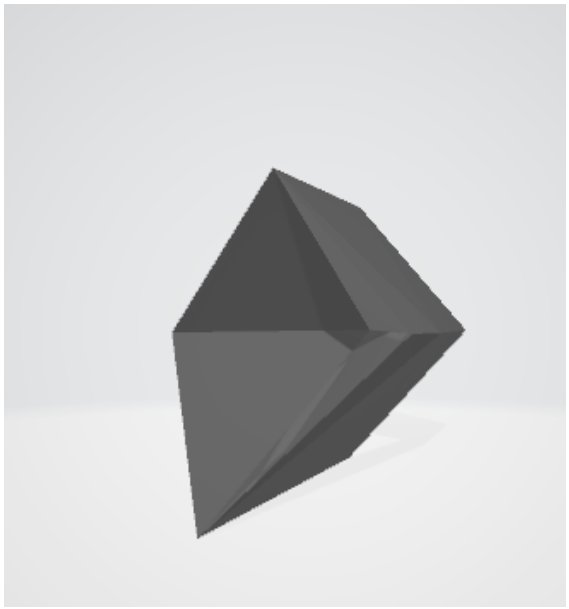
```
#face olho
cat('f 12 20 22 14 \n')
cat('f 13 21 22 14 \n')
cat('f 17 19 22 21 \n')
cat('f 19 22 20 16 \n')
cat('f 50 16 18 29\n')
cat('f 51 12 15 28\n')
```

Fonte - Autores

3.4 Resultados

Como visto na Figura 13, o modelo gerado não se assemelha a uma pessoa. Isso se deve a alguns erros no script que está sendo desenvolvido e por uma quantidade baixa de pontos detectados.

Figura 13 - Resultado do primeiro teste



Fonte – Autores

A detecção de borda, por exemplo, na parte direita do rosto, o código falhou em distinguir o limite do rosto e o início da parede. Para solucionar esse contratempo, verificamos que a mesma biblioteca de detecção de bordas também oferecia uma ferramenta de aumentar o contraste da imagem, que gerava um bom resultado quando combinado, corrigindo assim o módulo.

Além disso, nesse primeiro teste foi utilizado fotos com óculos o que dificultou a detecção dos olhos na vista de frente, e impossibilitou que o rosto fosse detectado de lado.

Isso provou um grande problema não só por limitar a quantidade de pontos que poderíamos conseguir do modelo, mas já que o diâmetro do rosto foi determinado de lado, não foi possível escalar a imagem de lado para condizer com a imagem de frente.

Então é partindo do pressuposto que a foto de lado não só está na mesma distância da câmera de frente como também na mesma altura, o que gera imprecisões se o nariz numa foto estava 10 pixels para cima da outra. Futuros testes ocorreram sem acessórios para aumentar a precisão do sistema, recomendamos que nossos usuários também façam isso.

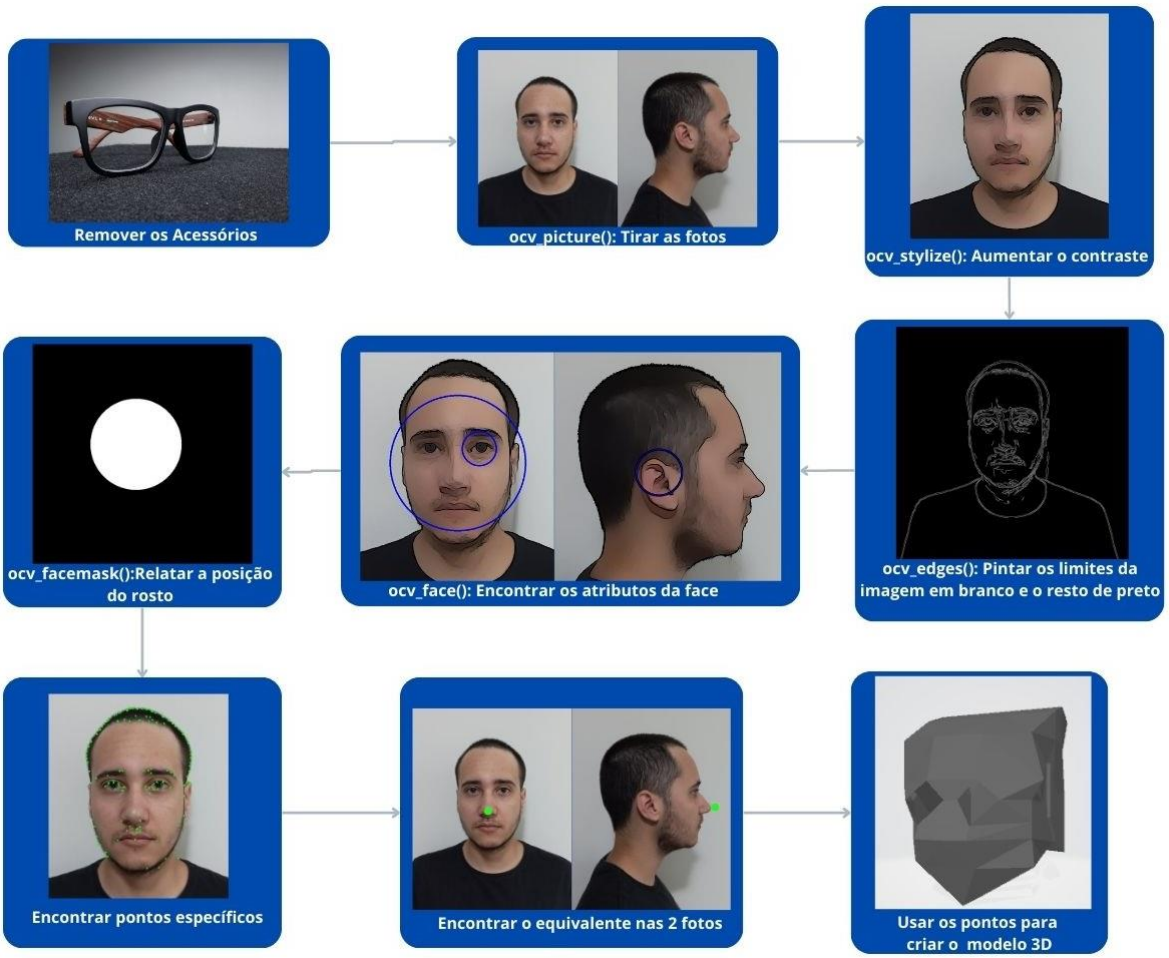
Logo, percebemos que será necessário melhorar o método de como o protótipo é gerado, para retornar resultados mais aceitáveis.

A Figura 14 ilustra os procedimentos de como o software executa. Inicialmente, quaisquer acessórios removíveis devem ser retirados para aprimorar a precisão ao capturar a fotografia.

Após o processamento da imagem do usuário, esta passará por modificações destinadas a intensificar o contraste, resultando em contornos faciais mais nítidos e reduzindo a ênfase em detalhes superficiais.

Dessa forma, a detecção de contornos pode ser aplicada, destacando de maneira clara as características faciais essenciais para análises subsequentes.

Figura 14 - Fluxograma dos processos do programa



Fonte – Autores

Portanto, empregamos funções para identificar a localização geral do rosto e outras características faciais. No entanto, mesmo que essas características sejam detectadas, a biblioteca OpenCV() requer a realização de um processo adicional para determinar o centro dessas características e suas dimensões.

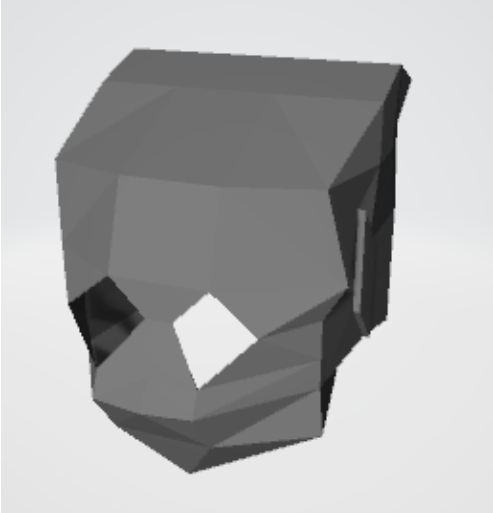
Utilizando os pontos fornecidos pelo sistema como base, é possível ampliar essa identificação. Muitos sistemas de visão computacional oferecem funcionalidades para recuperar pontos, embora bibliotecas não especializadas em reconhecimento facial possam gerar pontos não pertinentes à estrutura do rosto. Para mitigar esse problema, recorremos à simetria, a fim de localizar pontos não inicialmente detectados.

Isso envolve reconhecer características simétricas no rosto, permitindo a inferência de pontos que possam não ter sido diretamente detectados. Essa abordagem ajuda a aprimorar a precisão da detecção de pontos faciais mesmo em bibliotecas menos específicas para essa tarefa, tornando o processo mais robusto e confiável.

Após a conclusão desses processos nas duas imagens, utilizamos o eixo Y (os pontos se encontram em uma altura semelhante na imagem de frente e de lado), a orientação da imagem e outros fatores relevantes para identificar pontos equivalentes nas duas perspectivas. Registrando as coordenadas X e Y na imagem frontal e aproveitando a coordenada X da imagem lateral como Z, criamos um arquivo de extensão .obj contendo esses pontos. Ao estabelecer relações entre esses pontos, conseguimos gerar um modelo 3D.

Com dedicação ao aprimoramento do código, conseguimos produzir um modelo mais fiel às características humanas, identificando pontos distintos como os olhos, nariz e orelhas, como ilustrado na Figura 15.

Figura 15 - Resultado do segundo teste



Fonte – Autores

Esse modelo agora incorpora 90 vértices. Embora seja nove vezes maior em relação ao modelo anterior, essa quantidade ainda é relativamente baixa para alcançar níveis realistas de detalhe em modelos tridimensionais.

4 Discussão da solução proposta

A proposta abrange a concepção de um protótipo voltado à geração precisa de modelos 3D. Não obstante sua aptidão nesse contexto, ao avaliar tanto os resultados apresentados quanto as imagens empregadas no procedimento, constata-se uma discrepância entre eles.

Vale ressaltar que a obtenção automatizada de modelos 3D, sem a necessidade de aplicação de técnicas de redução de ruído, configura-se como uma ocorrência raramente observada.

Dessa maneira, empregaram-se variados métodos com o objetivo de aprimorar a malha e conferir maior fidelidade à pessoa que serviu de modelo exemplar, como resposta a esse desafio.

4.1 Testes

Os testes são partes fundamentais do processo de validação e verificação. Eles são usados para avaliar a funcionalidade do software, identificar problemas e garantir que ele funcione corretamente em diferentes cenários e condições. Assim foram realizados os testes e procedimentos citados abaixo.

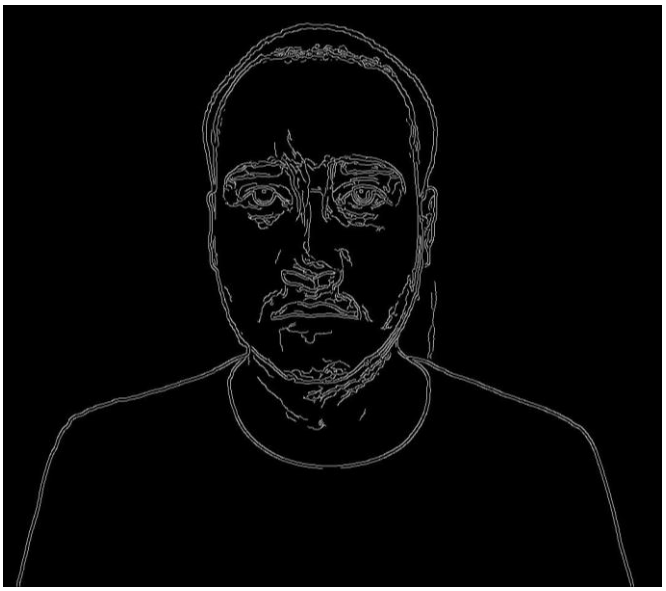
Para verificar se foi feito o sistema com as funcionalidades certas, foi utilizado Documento de Requisitos e verificando se todos os artefatos foram implementados de acordo com os requisitos e as normas apresentadas ao projeto.

A validação será um processo constante, mudando a visão do projeto e de como realizá-lo, podendo desenvolver junto com a maturidade dos desenvolvedores. Sempre que formos implementar algo, será verificado todos os métodos possíveis para realizar inovações e usar aquele que se provar mais adequado para a situação, se possível.

Com uma foto de frente e de lado dos dois desenvolvedores, o resultado esperado para o teste do primeiro protótipo é um modelo com poucos polígonos e sem textura. O segundo protótipo deve retornar a um modelo mais detalhado, com textura, mas utilizar mais fotos dos devs para evitar complicações éticas.

Para encontrar os pontos com mais facilidades, as imagens utilizadas passaram por um filtro de detecção de borda, visto na Figura 16, onde os limites são pintados de branco e o resto de preto. Dessa forma podemos percorrer pela imagem e procurar por valores de 1 nela, ao contrário de procurar o tom específico da pele, que pode variar de usuário.

Figura 16 - Detecção de borda



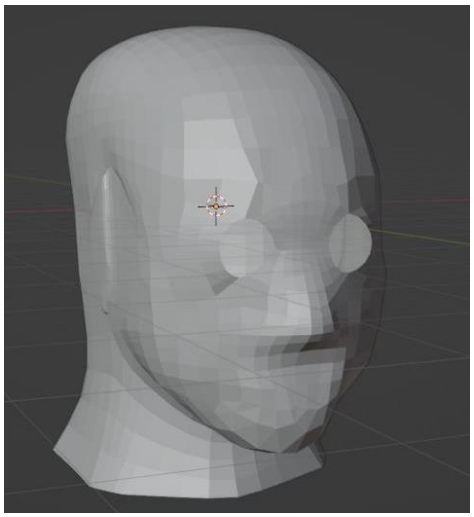
Fonte - Autores

4.2 Pós-Processamento

O modelo foi transferido para o software Blender, no qual cada superfície passou por um processo de subdivisão, resultando em um aumento no número de vértices. Adicionalmente, procedeu-se à aplicação de um alisamento no modelo, conferindo-lhe uma forma mais arredondada.

A Figura 17 evidencia que, mediante um período de 20 minutos no software, ao incrementar a densidade poligonal, houve uma significativa melhoria na qualidade do modelo gerado.

Figura 17 - Pós-Processamento do modelo



Fonte – Autores

Ao investir um maior lapso temporal na etapa de texturização, tornou-se mais evidente a identificação da origem do modelo, conforme ilustrado na Figura 18.

4.3 Texturização

Nossa aplicação não incorpora a geração automática da textura do modelo, essas imagens e coordenadas de texturas foram geradas no blender; devido à complexidade desse processo, que é influenciada por diversos fatores.

Primeiramente, a imagem de textura requer a unificação de todas as imagens capturadas para a construção do modelo, consolidando-as em uma única imagem e ajustando-as para corresponder à proporção do modelo, como ilustrado na Figura 19.

O segundo e último motivo reside no fato de que, a fim de aplicar essa textura ao modelo 3D, cada vértice necessita de, pelo menos, uma coordenada de textura na formatação Wavefront OBJ. Estes pontos de coordenadas possuem valores compreendidos entre 0 e 1, que correspondem a uma porcentagem da imagem, embora a geração destes valores não tenha sido realizada.

Figura 18 - Último modelo com textura

Fonte – Autores

Figura 19 - Textura do modelo final

Fonte – Autores

5 Conclusão

O problema central abordado por este artigo é a dificuldade enfrentada por softwares automáticos de geração de modelos 3D ao criar malhas 3D para rostos humanos.

Uma característica importante desse protótipo é a sua acessibilidade. Ele foi projetado de forma a ser facilmente utilizado por qualquer pessoa que necessite criar modelos 3D de rostos humanos, independentemente do nível de habilidade em modelagem.

A abordagem de simplificar as inserções do usuário, limitando-as a apenas duas fotos, demonstra uma compreensão clara das necessidades dos usuários com pouca ou nenhuma experiência em modelagem 3D. Isso reduz a curva de aprendizado e facilita o processo de criação, tornando a tecnologia mais acessível e amigável.

É importante reconhecer que o sistema possui suas limitações, como qualquer tecnologia. No entanto, ele fornece uma base sólida para a criação de modelos 3D, especialmente no contexto de personagens humanos.

A existência de um ponto de partida com elementos humanos pré-modelados reduz consideravelmente o tempo necessário para construir esses modelos a partir do zero. Isso é particularmente valioso para artistas e designers que desejam se concentrar em detalhes específicos ou em outras áreas criativas, em vez de investir tempo na modelagem básica.

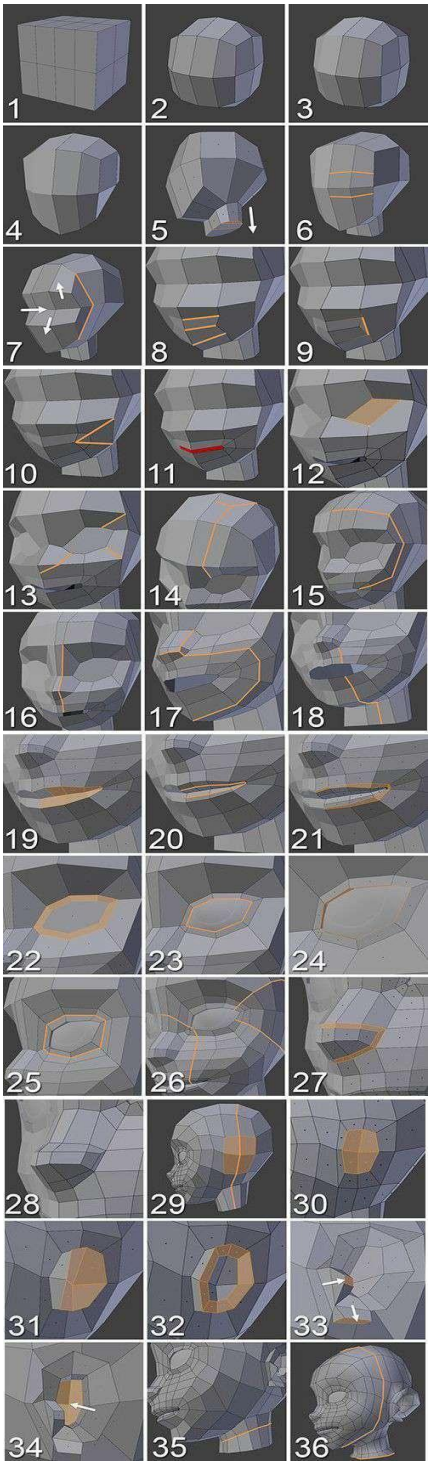
Com um pouco mais de sofisticação deste trabalho, o usuário pode criar um avatar para outras aplicações, como *streamar* na *Twitch* de maneira anônima, ou criar um auto retrato estilizado em 3D, podendo até imprimir o mesmo e criar uma estátua do usuário ou um molde para manequim, ou ainda melhorar um CGI realizado em um ator.

O protótipo descrito neste artigo pode facilitar e popularizar a modelagem em 3D, gerando uma redução no custo e tempo da produção de avatares. A Figura 20 retrata todos os passos necessários para a geração do modelo de forma manual.

A proposta do protótipo é fazer com que o usuário passe do primeiro para o trigésimo sexto passo sem a necessidade de passar pelos intermediários e reduzindo o tempo de geração do modelo 3D.

Neste caso, o usuário só precisaria de enviar algumas fotos e o sistema automaticamente gerará um modelo baseado nelas, o que o torna uma forma de modelagem 3D bastante acessível e fácil de usar numa era em que todo mundo tem um celular e um dos maiores diferenciais é a qualidade da câmera.

Figura 20 - Passos para a criação do modelo



Fonte - OREILY, 2020

Referências

3DFLOW(s.d.). 3DF Zephyr Free. Disponível em: https://www.3dflow.net/3df-zephyr-free/. Acesso em: 06 set. 2023

[ARXIV.ORG]. Large Pose 3D Face Reconstruction from a Single Image via Direct Volumetric CNN Regression. Disponível em: https://arxiv.org/abs/1703.07834. Acesso em: 05 jun. 2023.

BERNAL, Gabriel; BERNAL, Gustavo. Registro de Presença Acadêmico Através de Reconhecimento Facial.

CASA DAS CIÊNCIAS. Fotogrametria. Disponível em: https://rce.casadasciencias.org/rceapp/art/2019/073/. Acesso em: 05 jun. 2023.

[CS.CMU.EDU]. 3D Object File Formats. Disponível em: https://www.cs.cmu.edu/~mbz/personal/graphics/obj.html. Acesso em: 05 jun. 2023.

D1WQTXTS1XZLE7.CLOUDFRONT.NET. IA_FT_UNICAMP_visaoComputacional-libre.pdf. Disponível em: https://d1wqtxts1xzle7.cloudfront.net. Acesso em: 05 jun. 2023

GIOVANINI, Adenilson. Fotogrametria: o que é e para que serve.

HARTLEY, Richard; ZISSERMAN, Andrew. An Introduction to Computer Vision. Prentice Hall, 2003.

KASPERSKY. What Is Facial Recognition? Disponível em: https://www.kaspersky.com.br/resource-center/definitions/what-is-facial-recognition. Acesso em: 05 jun. 2023.

KEENTOOLS.IO. FaceBuilder for Blender. Disponível em: https://keentools.io/products/facebuilder-for-blender. Acesso em: 05 jun. 2023.

OLIVEIRA, Lucas; MATURANO, Nathan. Repositório para a disponibilização de documentos do Trabalho de Conclusão de Curso - Artur e Larissa. Disponível em: https://github.com/LuckyAmaral/TCC/tree/main/TccCode

ORVALHO, V., (2019) Reconhecimento facial, Rev. Ciência Elem., V7(4):073DOI http://doi.org/10.24927/rce2019.073 Acesso em: 06 set.2023

REVOPOINT3D.COM. Range. Disponível em: https://revopoint3d.com. Acesso em: 05 jun. 2023.

[RESEARCHGATE]. Untitled Document. Disponível em: https://www.researchgate.net. Acesso em: 05 jun. 2023.

SERASA. O que é reconhecimento facial e como funciona esse mecanismo? Disponível em: https://www.serasa.com.br/carteira-digital/blog/o-que-e-reconhecimento-facial-e-como-funciona-esse-mecanismo/. Acesso em: 05 jun. 2023.

SOLEM, Jan Erik. Programming Computer Vision with Python. O'Reilly Media, 2012.

SZELISKI, Richard. Computer Vision: Algorithms and Applications. Springer, 2011.

TOTVS. BPMN: entenda o que é a modelagem de processos de negócios, como fazer e sua importância! Disponivel em: https://www.totvs.com/blog/gestao-industrial/bpmn/ Acesso em: 06 out 2023

UPTODOWN. Meshroom para Windows - Baixe gratuitamente na Uptodown. Disponível em: https://pt.meshroom.com/windows. Acesso em: 05 jun. 2023.

ZHANG, Zhanpeng; LUO, Ping; LOY, Chen Change; TANG, Xiaoou. Facial landmarks detection based on 68 points, 2017.

ZKTECO. Introduction of 3D structured light facial recognition_4Jun,2021. Disponível em: https://zkteco.com.br. Acesso em: 05 jun. 2023.