UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

FERRAMENTA DE INDEXAÇÃO DE FACES HUMANAS EM VÍDEOS

LEONARDO LEAL OLIVEIRA

BLUMENAU
2018

LEONARDO LEAL OLIVEIRA

FERRAMENTA DE INDEXAÇÃO DE FACES HUMANAS EM VÍDEOS

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Dalton Solano dos Reis, Mestre - Orientador

BLUMENAU
2018

FERRAMENTA DE INDEXAÇÃO DE FACES HUMANAS EM VÍDEOS

Por

LEONARDO LEAL OLIVEIRA

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Dalton Solano dos Reis, Mestre – Orientador, FURB
Membro:	Prof. Aurélio Faustino Hoppe, Mestre – FURB
Membro:	Profa. Joyce Martins, Mestre – FURB

Blumenau, 12 de julho de ano 2018

Dedico este trabalho à minha mãe, que me criou com muito amor e carinho.

AGRADECIMENTOS

À minha mãe, que me criou com o maior carinho do mundo, enfrentando todo tipo de dificuldade que a vida trouxe pelo caminho.

Ao meu orientador, Dalton Solano dos Reis, pelo apoio e orientação durante a realização deste trabalho.

Aos meus professores, que ao longo do curso me apoiaram e transmitiram seus ensinamentos.

Aos meus colegas de curso, que me apoiaram durante o aprendizado e desafios do curso.

A todos que direta ou indiretamente me ajudaram a atingir essa conquista.

Tu te tornas eternamente responsável por aquilo que cativas.

Antoine de Saint-Exupéry

RESUMO

Este trabalho apresenta o desenvolvimento de uma ferramenta que realiza detecção e reconhecimento facial em vídeos. O fluxo da ferramenta é dividido em duas fases: a da indexação dos vídeos e a do uso dos dados extraídos durante e após a indexação. A indexação analisa cada um dos vídeos contidos na biblioteca, frame a frame, detectando regiões dos frames que possuam faces humanas. Com as faces humanas detectadas é utilizado um classificador com a técnica Local Binary Patterns Histograms (LBPH) para reconhecer e agrupar quem são as pessoas detectadas nos vídeos. Posteriormente as informações das pessoas encontradas, em quais vídeos e em quais partes de cada vídeo são disponibilizadas para que o usuário possa efetuar buscas e filtragens dentro de sua coleção de vídeos. É possível acompanhar a indexação em tempo real. Durante a reprodução do vídeo são desenhados retângulos ao redor das faces, juntamente com o nome da pessoa reconhecida, configurado pelo usuário. A implementação foi realizada pensando no reaproveitamento do trabalho, resultando em uma biblioteca e uma aplicação visual. A biblioteca possui o conjunto de classes para encapsular e facilitar a aplicação de detecção e reconhecimento facial em outros trabalhos. Já a aplicação visual faz uso da biblioteca e apresenta os resultados de forma gráfica para o usuário final. Foram realizados testes da experiência do usuário, do reaproveitamento da biblioteca por outros desenvolvedores e da escalabilidade da ferramenta em relação a uso de memória e performance. Os testes foram satisfatórios e mostraram que os usuários tiveram sucesso na maior parte das tarefas realizadas na ferramenta. A separação do trabalho em ferramenta de teste e biblioteca com o conjunto de lógicas encapsuladas também se mostrou proveitoso, visto que foi possível que outros desenvolvedores reutilizam facilmente esse trabalho em outros projetos.

Palavras-chave: Reconhecimento facial. Detecção facial. Visão computacional. Inteligência artificial.

ABSTRACT

This work presents the development of a tool that performs facial recognition and detection in videos. The tool's flow is divided into two phases: the indexing of videos and the use of extracted data during and after indexing. The indexing analyzes each of the videos contained in the library, frame by frame, detecting regions of the frames that have human faces. With the detected human faces, a classifier, with the technique Local Binary Patterns Histograms (LBPH), is used to recognize and group who the people detected in the videos are. Subsequently the information of the people found, which videos and which parts of each video they in is made available so that the user can search and filter within their video collection. It is possible to track the indexing in real time. During the visualization of the video, rectangles are drawn around the faces, along with the name of the recognized person, configured by the user. The implementation was performed with the reuse of the work, resulting in a library and a visual application. The library has the set of classes to encapsulate and facilitate the application of detection and facial recognition in other works. The visual application, however, makes use of the library and presents the results graphically to the end user. Tests of the user experience, the reuse of the library by other developers and the scalability of the tool in relation to memory usage and performance were performed. The tests were satisfactory and showed that users were successful in most tasks performed on the tool. The separation of the work in test tool and library with the set of encapsulated logics also proved itself useful, since it was possible that other developers could easily reuse this work in other projects.

Key-words: Facial recognition. Facial detection. Computer vision. Artificial intelligence.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Segundo Nisselson, Hunter-Syed e Shah (2017, p. 1), em 2022 existirão cerca de 45 bilhões de câmeras no mundo: “Onde houver um crescimento em câmeras, haverá imensas oportunidades de negócios em captura, análise e interpretação de dados visuais” (tradução nossa). Isso aponta como a quantidade de vídeos irá crescer, trazendo consigo a necessidade de interpretação e reconhecimento das informações contidas nesses vídeos de uma forma computacional e automatizada.

Com essa grande quantidade de vídeos, gera-se a necessidade de poder interpretá-los e catalogá-los de forma a facilitar buscas dentro de uma coleção. Uma forma de catalogar os vídeos é pelas faces de pessoas neles contidas, visto que o reconhecimento facial é um processo de identificação amplamente utilizado pelos seres humanos por conta da rapidez no reconhecimento de um indivíduo (DINIZ; SILVA; ALENCAR, 2016).

Como apontam Passarinho, Salles e Sarcinelli Filho (2015), detectar e acompanhar a movimentação de faces de pessoas são tarefas importantes em inúmeras aplicações, além de efetuar o reconhecimento e diferenciação das faces detectadas. O reconhecimento de faces é um dos estudos da visão computacional, que, segundo Rosa (2010), estuda o desenvolvimento de técnicas de interpretação de imagens para alguma aplicação útil, como detecção de movimento e reconhecimento de objetos.

Dentre as várias técnicas existentes para detecção de faces, Orts (2016) as divide em dois grandes grupos: as técnicas de identificação de faces e as de verificação de faces. A diferença é que uma faz a busca da face em um banco de faces reconhecidas, procurando por uma ocorrência de igualdade; a outra, de verificação, faz uso de técnicas para tentar verificar a igualdade de faces utilizando extração de características faciais (ORTS, 2016).

Com base nesses fundamentos, este trabalho apresenta o desenvolvimento de uma ferramenta para reconhecimento de faces em coleções de vídeos. Serão utilizados conceitos de visão computacional e reconhecimento de padrões para extrair características faciais e facilitar a catalogação dos indivíduos identificados nos vídeos.

1.1 OBJETIVOS

O objetivo deste trabalho é implementar uma ferramenta para indexar uma coleção de vídeos com base nas faces humanas.

Os objetivos específicos são:

a) disponibilizar uma ferramenta reaproveitável e multiplataforma com a tecnologia dotNet;

b) disponibilizar uma coleção com as faces reconhecidas em cada vídeo.

1.2 ESTRUTURA

Este trabalho está dividido em quatro capítulos. O primeiro capítulo apresenta os objetivos e a motivação para o desenvolvimento do trabalho. O segundo capítulo trata da fundamentação teórica, explicando os principais conceitos e técnicas utilizadas no desenvolvimento da ferramenta. No terceiro capítulo são descritos a arquitetura do trabalho através de diagramas, o detalhamento da implementação da ferramenta e os resultados obtidos nos testes realizados. Por fim, são apresentadas as conclusões e limitações do trabalho, assim como sugestões para extensões e trabalhos futuros.

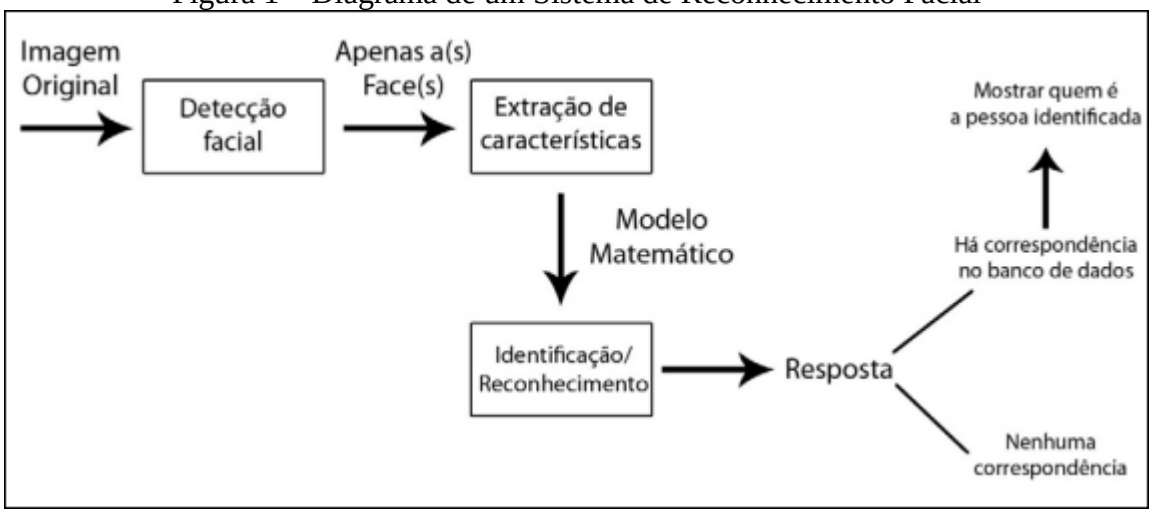
2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo explorar o principal assunto utilizado para a realização deste trabalho e três trabalhos correlatos. A seção 2.1 aborda o reconhecimento facial. Por fim, a seção 2.2 apresenta os trabalhos correlatos.

2.1 RECONHECIMENTO FACIAL

Sistemas que utilizam reconhecimento facial são capazes de receber como entrada uma imagem ou vídeo, identificar as faces presentes, diferenciá-las matematicamente via técnicas de estatística e, se necessário, compará-las com faces já reconhecidas que estejam armazenadas em um banco (BRAGA, 2013). A Figura 1 ilustra um possível sistema de reconhecimento facial. Para obter melhores resultados, pode ser realizado um pré-processamento na imagem, antes de ser feita a detecção facial. O pré-processamento visa diminuir ruídos e reduzir a taxa de erro da detecção facial.

Figura 1 – Diagrama de um Sistema de Reconhecimento Facial



Fonte: Braga (2013).

Na detecção facial é popularmente utilizada a biblioteca OpenCV, que possui a implementação do algoritmo Viola-Jones. O algoritmo Viola-Jones tenta encontrar em uma imagem características que são compatíveis com uma determinada classe a ser detectada. Essa classe pode ser definida como o formato de uma face, no contexto de detecção facial. A biblioteca OpenCV é uma biblioteca desenvolvida pela Intel no ano 2000 para aplicação em sistemas de tempo real envolvendo visão computacional e processamento de imagens (DINIZ et al. 2013).

Para que seja possível diferenciar faces umas das outras e agrupá-las quando forem da mesma pessoa, são utilizadas técnicas estatísticas para extrair características únicas de cada face. A técnica mais popular é a de análise das componentes principais (PCA). PCA é uma

técnica matemática de redução de dimensionalidade, baseada em extrair componentes principais de um espaço multidimensional. É utilizada em reconhecimento de padrões para eliminar redundâncias de informação e mesmo assim manter as principais características de cada padrão (BRAGA 2013).

Após a extração das características e montagem de um modelo matemático para cada face detectada, vem a fase do reconhecimento. O reconhecimento pode ser dividido em dois modos: verificação (autenticação) ou identificação. No modo de verificação o sistema verifica se uma pessoa é realmente quem ela está dizendo ser, como uma pessoa portando um passaporte e apresentando-o como seu documento de identidade. A verificação é um processo realizado de um para um, onde a face de entrada é comparada com uma face esperada. Já o modo de identificação é um modo onde a face de entrada é comparada com todas as faces do banco de dados para buscar uma correspondência. Como a identificação é um processo de um para vários, ela tende a ser mais custosa computacionalmente (BRAGA 2013).

2.2 TRABALHOS CORRELATOS

A seção 2.3.1 descreve o trabalho de Koch (2012), que desenvolveu um protótipo para reconhecimento de faces e identificação de usuários de páginas e aplicativos web. A seção 2.3.2 apresenta um método de fusão de dois algoritmos para um *tracking* de faces mais robusto, proposto por Jiang et al. (2016). Por fim, a seção 2.3.3 aborda o Video Indexer, aplicação da Microsoft para extração de *insights* de vídeos, incluindo reconhecimento de faces.

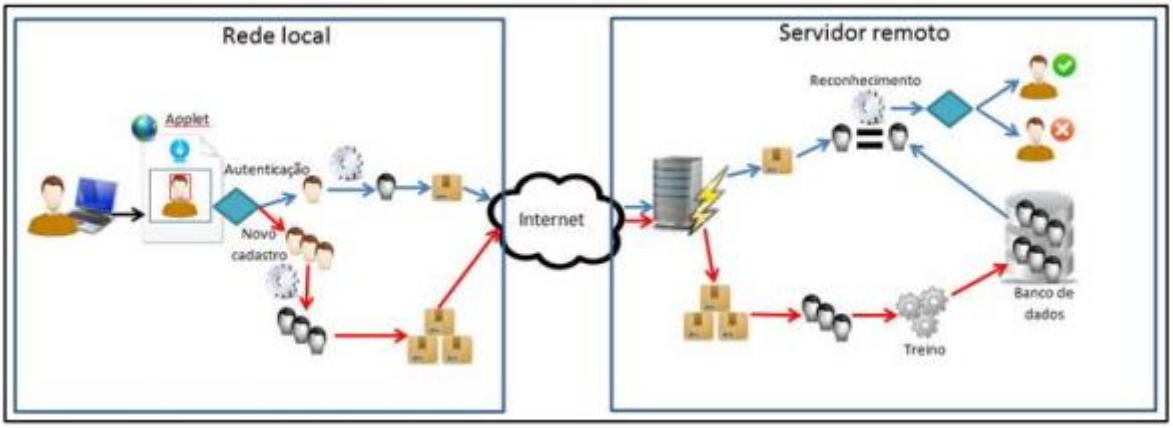
2.2.1 Visão computacional para reconhecimento de faces aplicado na identificação e autenticação de usuários na web

Koch (2012) especificou e implementou a identificação e autenticação de usuários por meio do reconhecimento facial em aplicativos, focado em web. O protótipo utiliza um *applet* Java no navegador para detectar e capturar a face do usuário, então faz o processamento e a transferência para o servidor, que é responsável por extrair as informações, via *socket*. O armazenamento das informações é feito numa base de dados Oracle. Para a identificação, é feita a comparação dos dados da face apresentada com os dados de faces conhecidas pelo banco; sendo a face de um usuário reconhecida, os dados são retornados para o *applet*, redirecionando para uma página de boas-vindas.

Todo o processo de detecção e identificação das imagens das faces é realizado com a API JavaCV, que é a versão Java da API OpenCV. A extração e o reconhecimento das

características das faces foram realizados pela técnica PCA. A Figura 2 apresenta o processo de separação de responsabilidades do modelo de processamento distribuído utilizado. O lado esquerdo da Figura 2 representa a rede local, onde o usuário está acessando a aplicação por um navegador web, que está executando um *applet* para a detecção e captura das faces. A comunicação com o servidor passa pela internet via *socket*. No lado direito da imagem é demonstrado o funcionamento do servidor, que realiza a recepção das informações, o processamento do reconhecimento das faces e a comparação com as faces cadastradas no banco de dados.

Figura 2 – Processo distribuído utilizado pela aplicação de Koch (2012)



Fonte: Koch (2012).

Segundo Koch (2012), os objetivos específicos foram atingidos. A utilização da técnica estatística de análise das componentes principais (PCA) mostrou-se muito eficiente para o reconhecimento de indivíduos por meio de suas faces. Dentro de ambientes controlados, com iluminação moderada, sem excesso de brilho, tendo o indivíduo na posição frontal e com pose neutra foi possível atingir mais de 96% de acerto. Como era possível cadastrar mais de uma foto do mesmo indivíduo, a aplicação se demonstrou tolerante à autenticação com expressões faciais moderadas, desde que haja alguma face já cadastrada com uma expressão semelhante. Uma situação específica de falsos positivos foi a de mulheres que se cadastraram com cabelo solto e tentaram se autenticar com cabelo preso, ou vice-versa. Para essas situações bastou efetuar o cadastro na base de faces com as duas situações, sendo então possível contornar a situação.

Koch (2012) também aponta que a API OpenCV foi de extrema importância para várias tarefas, como o acesso ao dispositivo de vídeo, o pré-processamento e a manipulação das imagens, as funções matemáticas e estruturas de manipulação de matrizes. O autor também apontou que a API apresentou problemas de liberação de memória quando utilizada

com *threads* e em situações onde era utilizado um grande número de imagens ocorreram erros nos métodos `cvCalcEigenObjects()` e `cvEigenDecomposite()`.

2.2.2 A fusion method for robust face tracking

Jiang et al. (2016) propuseram a fusão dos algoritmos Supervised Descent Method (SDM) e Compreesive Tracking Method (CT) para obter um *tracking* de faces mais robusto em vídeos com grande incidência de oclusões e movimentações das faces. A aplicação do SDM e do CT é realizada ao mesmo tempo. O SDM é responsável por corrigir erros de movimentação do CT continuamente durante o *tracking* facial frontal. Entretanto, quando a orientação da face atinge o ângulo ortogonal ao ponto de vista (completamente de lado), resulta em erros no *tracking* realizado pelo método SDM. Nestas situações o CT é utilizado para manter a região rastreada até que o SDM detecte e rastreie a face novamente.

O protótipo foi implementado na linguagem C++ com auxílio da biblioteca OpenCV. Na Figura 3 são demonstrados os resultados dos testes, sendo: resultado esperado em amarelo; CT tracker em azul; SDM tracker em verde; e a fusão proposta em vermelho. O algoritmo SDM utiliza a biblioteca OpenCV que, segundo os autores, é muito adequada para detecção de faces frontais, mas não consegue detectar faces em outros ângulos.

Figura 3 – Resultado dos testes de *tracking*

Fonte: Jiang et al. (2016).

Jiang et al. (2016) realizaram testes de *tracking* em tempo real utilizando vários desafios de literaturas recentes e concluíram que a estratégia da fusão atingiu uma performance encorajadora em termos de eficiência e confiabilidade. Foi possível manter a boa precisão do método SDM para detecção de faces frontais e evitar os problemas de flutuação do método CT. Com a utilização do método CT foi possível manter um *tracking* robusto em situações onde a aparência é alterada completamente do visual normal de uma face, como quando há oclusão da face por algum objeto ou pelas mãos da pessoa.

2.2.3 Video Indexer

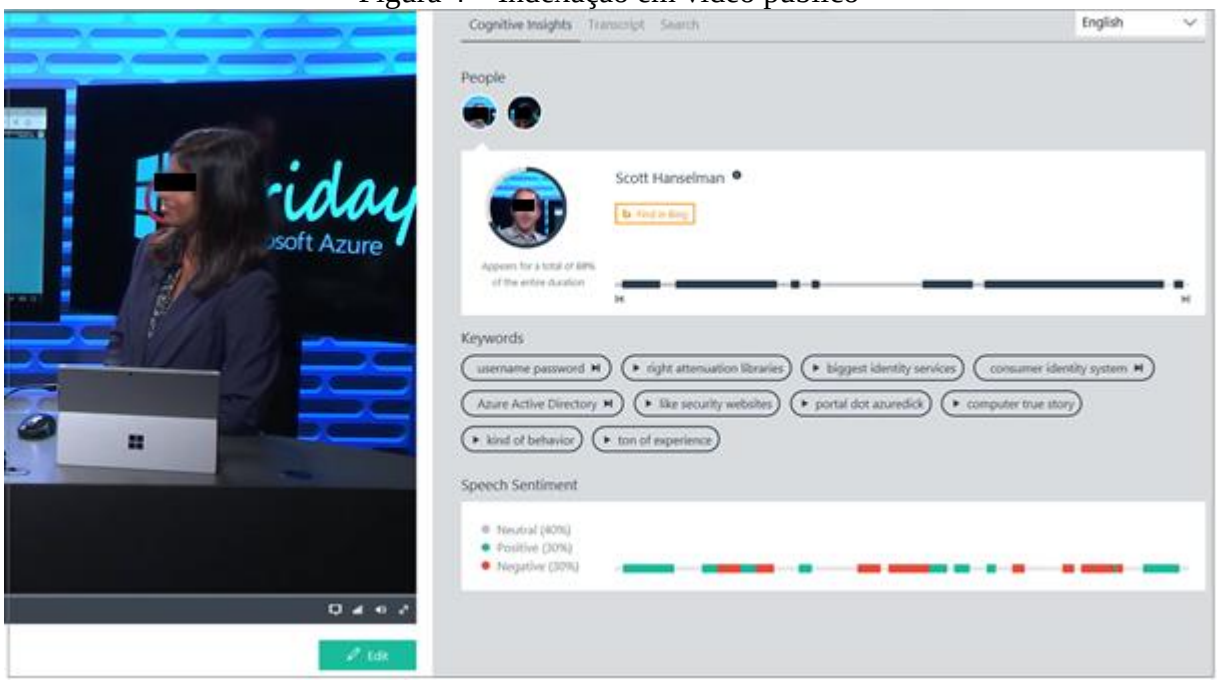
Segundo Kornich e Fowler (2017), o Video Indexer é um serviço em nuvem que possibilita extração de um conjunto de *insights* de vídeos utilizando tecnologias de inteligência artificial. O serviço possibilita: transcrição de áudio; *tracking* e identificação de faces; identificação de voz (quem está falando); reconhecimento visual de texto; separação de sons de fundo e vozes; detecção de cenas; extração de *frames* chave; análise sentimental das falas e expressão facial; tradução dos áudios para um conjunto de idiomas; moderação de conteúdo; extração de palavras chave; anotação do vídeo com base em uma coleção de objetos conhecidos.

O serviço pode tanto ser utilizado diretamente pelo portal ou por desenvolvedores diretamente pela API REST. Kornich e Fowler (2017) sugerem três possíveis cenários onde o serviço pode ser útil. O primeiro cenário é em pesquisas, onde a indexação pelos *insights* facilita a busca por vídeos ou momentos dentro deles em uma biblioteca de vídeos. O segundo cenário é na monetização, como o exemplo de indústrias que controlam publicidade, que saberão que anunciar uma chuteira durante um jogo de futebol será muito mais relevante do que anunciar durante uma competição de natação. Já o terceiro cenário é na experiência do usuário, como na situação de um aluno procurando conteúdo sobre pirâmides que encontra um vídeo de uma hora de duração, com os primeiros 30 minutos falando sobre esferas e os últimos sobre pirâmides. Nesse terceiro cenário o aluno seria beneficiado se o vídeo já fosse apresentado iniciando na posição de 30 minutos, onde começa o assunto de interesse.

Conforme Gada (2017), a tecnologia de *tracking* e identificação de faces compara as faces reconhecidas com um banco de celebridades, para avaliar quais celebridades estão presentes no vídeo. É possível adicionar uma *label* para as faces que não coincidiram com uma celebridade. Essas *labels* são utilizadas posteriormente para reconhecer outros vídeos que possuam a mesma face reconhecida. Na Figura 4 é possível visualizar as faces reconhecidas e os momentos em que cada uma apareceu durante o vídeo. Ao lado direito, são dispostas as

pessoas reconhecidas. A figura foi retirada de uma indexação realizada em vídeo público disponibilizado pela Microsoft.

Figura 4 – Indexação em vídeo público



Fonte: Kornich e Fowler (2017).

Para realizar a indexação, é necessário fazer o upload do vídeo e aguardar o processamento da indexação ser realizado nos servidores do Microsoft Azure, que é o sistema *cloud* da Microsoft. Ao finalizar a indexação, é enviado um e-mail para o usuário. É disponibilizado upload de 40h de vídeo na opção gratuita. O resultado da indexação, com todo o conteúdo reconhecido, pode ser baixado em arquivo `json`. O tamanho dos vídeos está limitado a 50MB e não é possível compartilhar o vídeo diretamente por uma URL, sem efetuar login na aplicação.

3 DESENVOLVIMENTO DA FERRAMENTA

Este capítulo demonstra as etapas do desenvolvimento da ferramenta. Na seção 3.1 são abordados os principais requisitos. A seção 3.2 apresenta a especificação. A seção 3.3 descreve de forma detalhada a implementação. Por fim, a seção 3.4 apresenta os resultados dos testes e sugestões de extensões da ferramenta.

3.1 REQUISITOS

A ferramenta a ser desenvolvida deve:

a) receber um vídeo como entrada e retornar uma coleção com as faces reconhecidas (Requisito Funcional - RF);
b) receber um vídeo e a imagem de uma face como entrada e retornar se a face foi encontrada dentro do vídeo (RF);
c) ter métodos de busca do status atual da detecção facial durante a execução/processamento do vídeo (RF);
d) utilizar bibliotecas de processamento de imagens como OpenCV e Viola-Jones (RF).
e) ser multiplataforma e reaproveitável para uso em outras aplicações (Requisito Não Funcional - RNF);
f) ser implementada na linguagem C#, utilizando a IDE Visual Studio 2017 e utilizando a plataforma dotNet (RNF);
g) utilizar algoritmos de pré-processamento e segmentação para a detecção das faces (RNF);

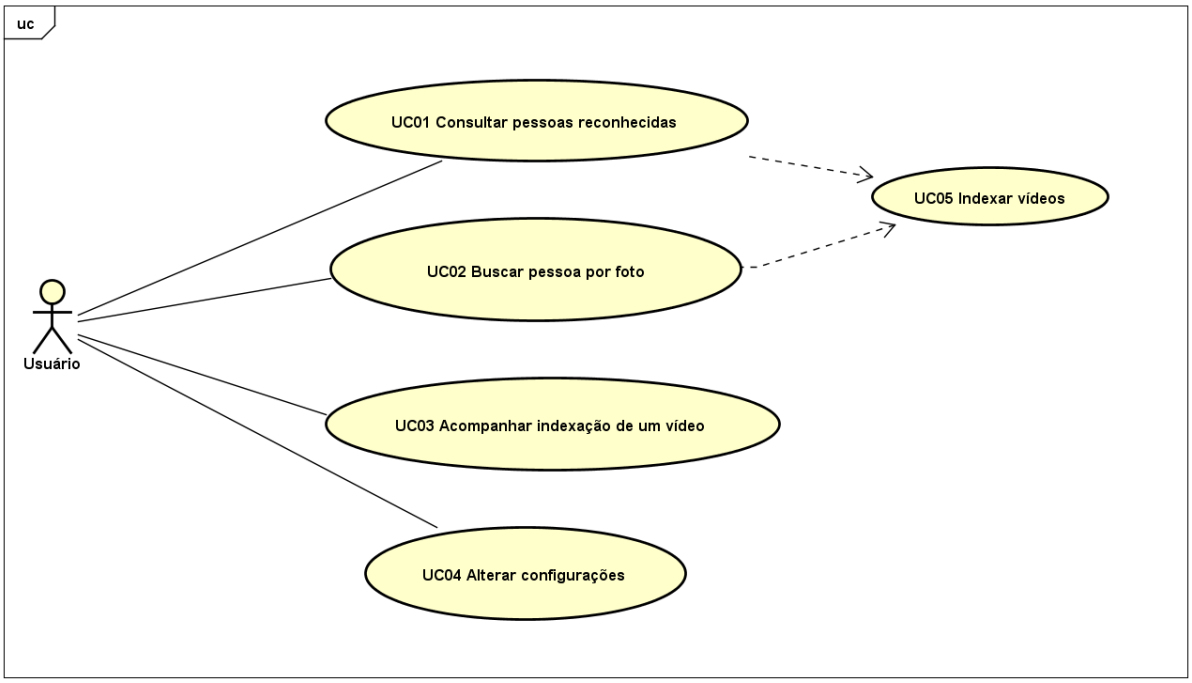
3.2 ESPECIFICAÇÃO

A especificação da ferramenta é apresentada por diagramas UML, utilizando a ferramenta Astah Community 7.1.0/f2c212 Model Version 37. Neste trabalho foram desenvolvidos diagramas de caso de uso e de atividades, apresentados nas seções a seguir.

3.2.1 Diagrama de casos de uso

Esta seção apresenta e descreve o diagrama de caso de uso da ferramenta. Como pode ser visto na Figura 5, a ferramenta possui apenas um ator, que é o `Usuário`, responsável por acessar todas as funcionalidades disponíveis.

Figura 5 – Diagrama de casos de uso



Fonte: elaborado pelo autor.

O `UC01 Consultar pessoas reconhecidas` permite que o usuário consulte as pessoas que já foram identificadas até o momento pela indexação dos vídeos. Esse caso de uso depende da prévia indexação dos vídeos, que pode ter sido feita parcialmente, indicada no `UC05 Indexar vídeos`.

No `UC02 Buscar pessoa por foto` a ferramenta permite que o usuário possa selecionar uma foto da face de uma pessoa e efetuar uma busca dentre as pessoas já reconhecidas. A busca engloba apenas as pessoas já reconhecidas dos vídeos já indexados.

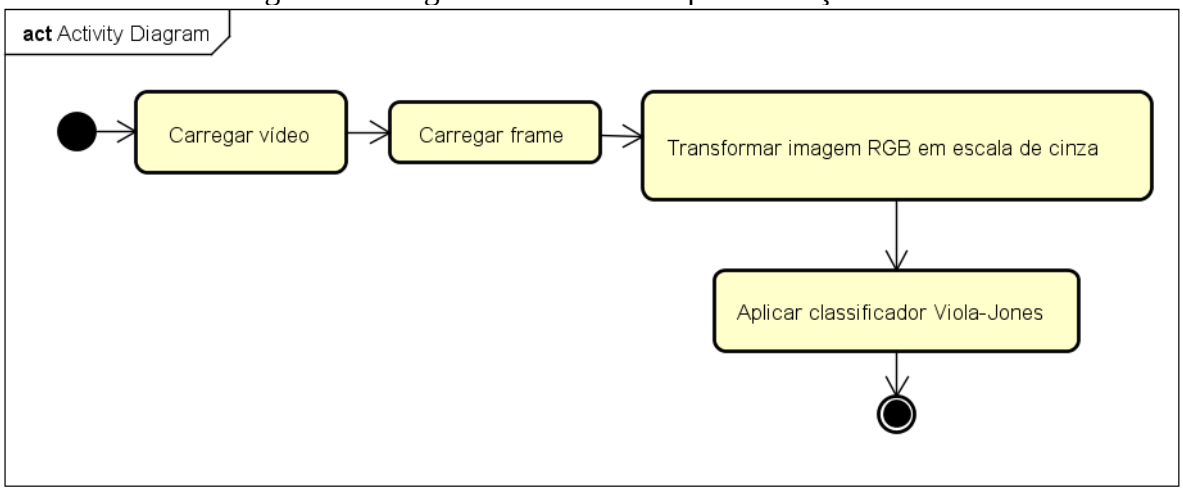
O `UC03 Acompanhar indexação de um vídeo` permite que o usuário visualize retângulos ao redor das faces detectadas, *frame* a *frame*. No canto superior esquerdo de cada retângulo é exibido o nome da pessoa reconhecida.

Por fim, o `UC04 Alterar configurações` permite que o usuário parametrize os algoritmos de detecção e reconhecimento facial utilizados durante a indexação dos vídeos. Também é possível configurar diretórios e extensões dos vídeos, assim como elementos visuais.

3.2.2 Detecção facial

Nesta seção são detalhados os passos para realizar a detecção das faces humanas nos vídeos. A Figura 6 apresenta o diagrama de atividades com as etapas desse processo.

Figura 6 – Diagrama de atividades para detecção facial



Fonte: elaborado pelo autor.

Primeiramente o arquivo de vídeo é carregado utilizando a classe `Capture` da biblioteca EmguCV e o primeiro *frame* é acessado. A biblioteca EmguCV encapsula a biblioteca OpenCV, que é a versão original em C e C++. O Quadro 1 apresenta o código fonte para o carregamento do vídeo e o acesso ao primeiro *frame*. Para o processamento dos próximos *frames*, é utilizado uma *thread* que utiliza o mesmo método para pegar o primeiro *frame*. O método *QueryFrame* da biblioteca EmguCV, sempre retorna o próximo *frame* do vídeo.

Quadro 1 – Carregamento do vídeo e primeiro *frame*

```
Capture = new Capture(Video.FilePath);
CurrentFrame = new Image<Bgr, Byte>(Capture.QueryFrame().Bitmap);
```

Fonte: elaborado pelo autor.

Em seguida a imagem é transformada em escala de cinza e o classificador é aplicado. A transformação para escala de cinza é requisito para o classificador. O classificador utilizado é a classe `CascadeClassifier` da biblioteca EmguCV. O método de detecção retorna um vetor de retângulos com as regiões onde foram detectadas faces humanas.

O classificador Viola-Jones também pode ser utilizado para detecção de outros padrões de imagens. No momento da criação do objeto, é definido um arquivo XML com informações de pré-treinamento para a classificação de um tipo de objeto. Neste trabalho foram utilizados arquivos padrão disponibilizados juntamente com a biblioteca OpenCV, sendo que o arquivo default é `haarcascade_frontalface_alt_tree.xml`. No Quadro 2 é apresentado o método de conversão para escala de cinza e a execução do classificador Viola-Jones sobre o *frame* atual.

Quadro 2 – Escala de cinza e classificação

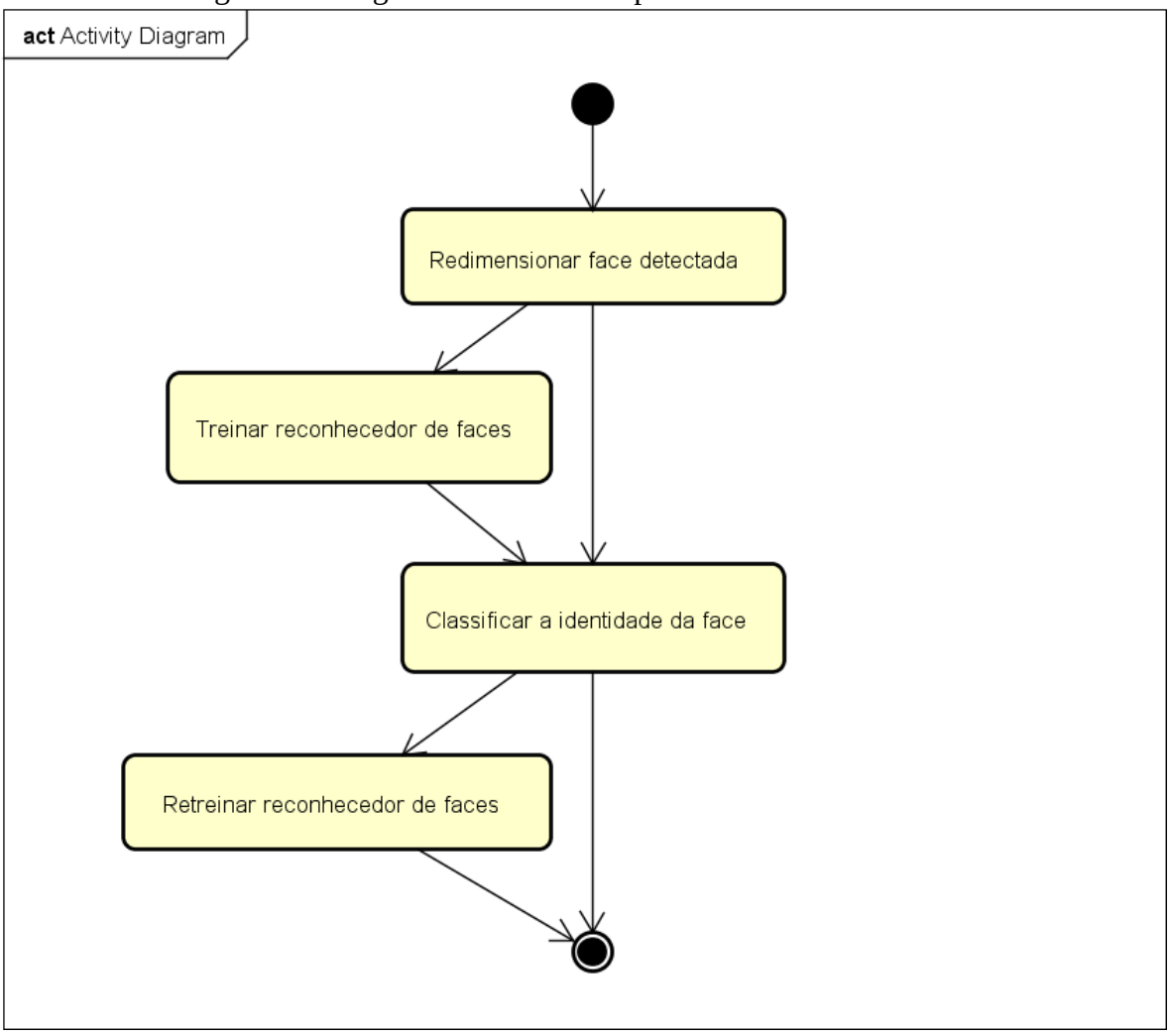
```
Image<Gray, Byte> currentGrayFrame = CurrentFrame.Convert<Gray, Byte>();
Rectangle[] facesDetected =
CascadeClassifier.DetectMultiScale(currentGrayFrame);
```

Fonte: elaborado pelo autor.

3.2.3 Reconhecimento facial

Nesta seção são detalhados os passos aplicados para efetuar o reconhecimento das faces detectadas nos vídeos. O objetivo é distinguir faces diferentes e agrupar faces iguais. Na Figura 7 são apresentadas as etapas desse processo em um diagrama de atividades.

Figura 7 – Diagrama de atividades para reconhecimento facial



Fonte: elaborado pelo autor.

Inicialmente o retângulo com a face detectada é redimensionado para um tamanho único, que pode ser configurado, tendo o padrão em 200x200. Esse redimensionamento é um requisito das classes de reconhecimento facial disponíveis na biblioteca EmguCV. No Quadro 3 é apresentado o código fonte do redimensionamento da face.

Quadro 3 – Redimensionamento da face

```
Image<Gray, Byte> faceRegion = currentGrayFrame.Copy(face);
faceRegion = faceRegion.Resize(DataFile.config.FaceRectangleSize,
DataFile.config.FaceRectangleSize, Emgu.CV.CvEnum.Inter.Cubic);
```

Fonte: elaborado pelo autor.

Dentre as três opções de classes para reconhecimento facial, foi optado pela classe `LBPHFaceRecognizer`. Essa classe necessita que exista pelo menos uma imagem de pré-treinamento antes de efetuar a classificação das faces. Portanto, caso não haja nenhuma face já reconhecida, a primeira face recebe um identificador sequencial de número um e é adicionada ao conjunto de faces reconhecidas utilizado para efetuar o treinamento do reconhecedor. O treinamento inicial com a primeira face detectada é apresentado no Quadro 4.

Quadro 4 – Treinamento do reconhecedor

```
if (TrainImages.Count == 0)
{
    idPerson = TrainLabels.Count + 1;

    TrainImages.Add(faceRegion);
    TrainLabels.Add(idPerson);
    FaceRecognizer.Train(TrainImages.ToArray(), TrainLabels.ToArray());

    SaveFace(CurrentFrame.Copy(originalFace), originalFace, idPerson);
}
```

Fonte: elaborado pelo autor.

Com a face redimensionada e o reconhecedor de faces treinado com pelo menos uma face, é realizada a classificação da identificação da nova face dentro das faces já reconhecidas. Caso a nova face seja identificada dentro das faces já reconhecidas, é retornado o identificador da pessoa. Caso contrário, se for uma pessoa não reconhecida ainda, a nova face é adicionada às faces de treinamento com um novo identificador gerado sequencialmente. O método da classe `LBPHFaceRecognizer` responsável pela classificação das faces é o `Predict`. A execução da classificação e o retreinamento em caso de nova face é apresentado no Quadro 5.

Quadro 5 – Classificação das faces

```
FaceRecognizer.PredictionResult predictionResult =
FaceRecognizer.Predict(faceRegion);

if (predictionResult.Label == 0)
{
    idPerson = TrainLabels.Count + 1;

    TrainImages.Add(faceRegion);
    TrainLabels.Add(idPerson);
    FaceRecognizer.Train(TrainImages.ToArray(), TrainLabels.ToArray());

    SaveFace(CurrentFrame.Copy(originalFace), originalFace, idPerson);
}
else
    idPerson = predictionResult.Label;
```

Fonte: elaborado pelo autor.

3.2.4 Visualização da detecção e reconhecimento facial

Esta seção apresenta como é realizada a apresentação do reconhecimento facial para visualização do usuário. Enquanto o vídeo está sendo visualizado, quando um dos *frames* possui faces detectadas, elas são destacadas com retângulos ao redor de cada uma e com o nome de cada pessoa. Caso a pessoa detectada seja uma nova pessoa, ainda não reconhecida em outros momentos, é dado um nome padrão no formato `Pessoa: n`, onde `n` é um número sequencial.

O Quadro 6 mostra o trecho de código responsável por destacar a pessoa reconhecida desenhando um retângulo ao redor da face e o nome da pessoa reconhecida.

Quadro 6 – Desenho do retângulo e nome da pessoa

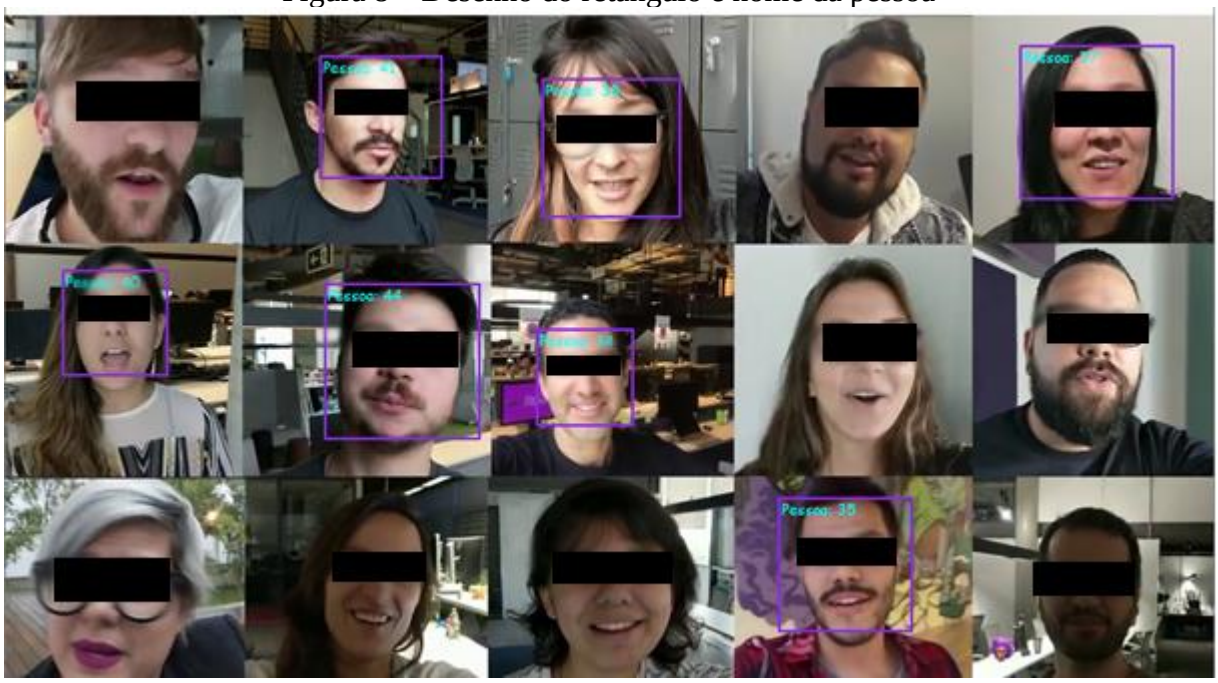
```
foreach (Face face in frame.Faces)
{
    CurrentFrame.Draw(face.Position, new
Bgr(DataFile.config.RectangleColor), DataFile.config.RectangleThickness);

    Graphics g = Graphics.FromImage(CurrentFrame.Bitmap);
    g.DrawString(DataFile.GetPerson(face.IdPerson).Name, new
Font(DataFile.config.FontName, DataFile.config.FontSize), new
SolidBrush(DataFile.config.ColorFont), new Point(face.Position.Location.X,
face.Position.Location.Y));
}
```

Fonte: elaborado pelo autor.

O resultado visual do desenho do retângulo e do nome da pessoa pode ser visto na Figura 8.

Figura 8 – Desenho do retângulo e nome da pessoa



Fonte: elaborado pelo autor.

3.2.5 Persistências

Nesta seção são apresentadas quais são e como são realizadas as persistências do trabalho. Ao todo, são persistidos três arquivos no formato JSON com a biblioteca Json.NET. As configurações da ferramenta são armazenadas no arquivo `config.json` e os campos e valores padrão podem ser visualizados no Quadro 7.

Quadro 7 – Arquivo de configurações

```
{
  "FontName": "Comic Sans MS",
  "FontSize": 12,
  "ColorFont": "Aqua",
  "RectangleColor": "BlueViolet",
  "RectangleThickness": 2,
  "VideosExtensions": "*.mp4",
  "VideosDirectory": "Vídeos/",
  "FacesDirectory": "Temp/",
  "FacesExtension": ".png",
  "HaarCascadeDirectory": "haarcascade/",
  "CascadeClassifier": "haarcascade_frontalface_alt_tree.xml",
  "FaceRectangleSize": 200,
  "ResizeScale": 1.0,
  "LBPH_threshold": 80.0,
  "LBPH_radius": 1,
  "LBPH_neighbors": 8,
  "LBPH_gridX": 8,
  "LBPH_gridY": 8,
  "PeopleFaceSize": 200,
  "VideoThumbSize": 200
}
```

Fonte: elaborado pelo autor.

São salvas informações dos arquivos de vídeo localizados no diretório configurado pelo usuário dentro do arquivo `videos.json`. Como a indexação de um vídeo toma um certo tempo e existe a possibilidade de haver vídeos repetidos dentro do diretório, foi utilizado o algoritmo de geração de *hash* no conteúdo do arquivo. Com a geração da *hash*, não serão indexados arquivos de vídeo com conteúdo repetido. A geração da *hash* é apresentada no Quadro 8.

Quadro 8 – Geração de *hash* para arquivos de vídeo

```
public static string CalculateMD5(string filename)
{
    using (var md5 = MD5.Create())
    {
        using (var stream = File.OpenRead(filename))
        {
            var hash = md5.ComputeHash(stream);
            return BitConverter.ToString(hash).Replace("-",
"").ToLowerInvariant();
        }
    }
}
```

Fonte: elaborado pelo autor.

Cada vídeo armazenado no arquivo `videos.json` possui a *hash*, o nome do arquivo, a *thumb* e a coleção com os *frames* onde existem faces detectadas. Cada *frame* da coleção tem a identificação da face detectada e em qual posição da imagem ela foi detectada. Um exemplo do arquivo é apresentado no Quadro 9.

Quadro 9 – Arquivo de vídeos

```
{
  "ee43ec65b87caedc2a74000ee31be301": {
    "FilePath": "Vídeos/1280x720.mp4",
    "Thumb": {
      "Rows": 200,
      "Cols": 200,
      "NumberOfChannels": 3,
      "CompressionRatio": 0,
      "Bytes": "PDAyPDAyPDAyPDAyPD[...]",
      "Roi": "0, 0, 200, 200"
    },
    "Frames": [
      {
        "Index": 159,
        "Faces": [
          {
            "IdPerson": 3,
            "Distance": 1.7976931348623157E+308,
            "Position": "573, 318, 104, 104"
          }
        ]
      },
[...]
}
```

Fonte: elaborado pelo autor.

O último arquivo armazenado é o `people.json`. Nesse arquivo são armazenadas informações sobre as pessoas detectadas. Existe a possibilidade de uma mesma pessoa ser reconhecida duas vezes e ser considerada duas pessoas diferentes. Para essa situação o usuário pode indicar uma relação entre as duas pessoas; no arquivo essa relação é indicada pela coleção de clones que cada pessoa tem. No Quadro 10 é apresentada a estrutura do arquivo de pessoas.

Quadro 10 – Arquivo de pessoas

```
[
  {
    "Id": 1,
    "Name": "Pessoa: 1",
    "MainPhoto": {
      "Rows": 576,
      "Cols": 576,
      "NumberOfChannels": 3,
      "CompressionRatio": 0,
      "Bytes": "t7e3t7e3tLa4tLa4ubi6ubi6trm5trm5[...]",
      "Roi": "0, 0, 576, 576"
    },
    "Clones": []
  },
[...]
```

Fonte: elaborado pelo autor.

Para efetuar a leitura de um arquivo no formato JSON com a biblioteca Json.NET, basta definir a classe do objeto que foi gravado no arquivo e chamar o método `JsonConvert.DeserializeObject`, passando como parâmetro todo o conteúdo do arquivo lido como texto, como pode ser visto no Quadro 11.

Quadro 11 – Leitura de arquivo JSON

```
if (File.Exists(configFileName))
{
    config =
JsonConvert.DeserializeObject<Config>(File.ReadAllText(configFileName));
}
```

Fonte: elaborado pelo autor.

Já a gravação é realizada transformando o conteúdo do objeto em texto com o método `JsonConvert.SerializeObject` e gravando a string retornada no arquivo desejado. A gravação de arquivo com a biblioteca Json.NET é apresentada no Quadro 12.

Quadro 12 – Gravação de arquivo JSON

```
File.WriteAllText(configFileName, JsonConvert.SerializeObject(config,
Formatting.Indented));
```

Fonte: elaborado pelo autor.

3.2.5.1 Reconhecimento facial

As informações necessárias para a aplicação do reconhecimento facial, após fechar a aplicação e abrir de novo, são extraídas do arquivo de pessoas. São necessárias as faces das

pessoas e um identificador para cada uma delas. Após o arquivo `pessoas.json` ser lido, é realizada a extração dessas informações, que são organizadas no padrão necessário para realizar o treinamento do reconhecimento pela classe `LBPHFaceRecognizer`. O método que extrai essas informações para realizar o treinamento é apresentado no Quadro 13.

Quadro 13 – Carregamento da base de treinamento

```
public static void LoadTrainingData(List<int> TrainLabels,
List<Image<Gray, Byte>> TrainImages)
{
    for (int idx = 0; idx < people.Count; ++idx)
    {
        Person person = people[idx];

        TrainImages.Add(person.MainPhoto.Convert<Gray, Byte>());
        TrainLabels.Add(person.Id);

        for (int idxClones = 0; idxClones < person.Clones.Count;
++idxClones)
        {
            Person clone = person.Clones[idxClones];

            TrainImages.Add(clone.MainPhoto.Convert<Gray, Byte>());
            TrainLabels.Add(person.Id);
        }
    }
}
```

Fonte: elaborado pelo autor.

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

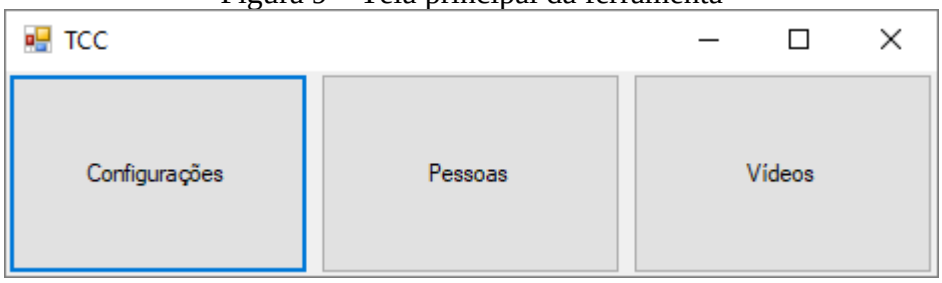
3.3.1 Técnicas e ferramentas utilizadas

A ferramenta foi desenvolvida utilizando a linguagem de programação C# com .NET Framework, versão 4.6.1 no ambiente de desenvolvimento Microsoft Visual Studio Enterprise 2017, versão 15.7.2, no sistema operacional Windows 10 Education. Para a implementação foram utilizadas as bibliotecas EmguCV e Json.NET. EmguCV é uma biblioteca para fazer a ponte entre aplicações .NET e as funções da biblioteca OpenCV, desenvolvida em C/C++. Json.NET é um framework para persistência de arquivos no formato JSON para aplicações .NET.

3.3.2 Operacionalidade da implementação

A tela principal da ferramenta possui três botões para acesso às telas de configurações, de pessoas reconhecidas e de vídeos encontrados. A Figura 9 apresenta a tela principal da ferramenta, onde é possível visualizar os botões descritos.

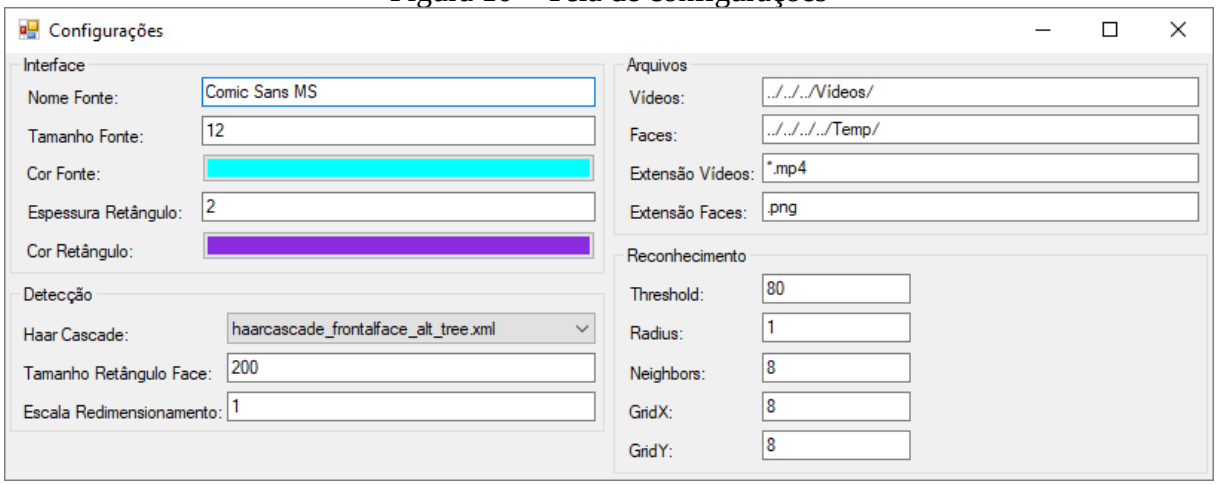
Figura 9 – Tela principal da ferramenta



Fonte: elaborado pelo autor.

O botão `Configurações` fornece acesso à tela de configurações da ferramenta, como pode ser visto na Figura 10. É possível personalizar detalhes visuais da interface, diretórios dos vídeos e parâmetros da detecção e reconhecimento facial.

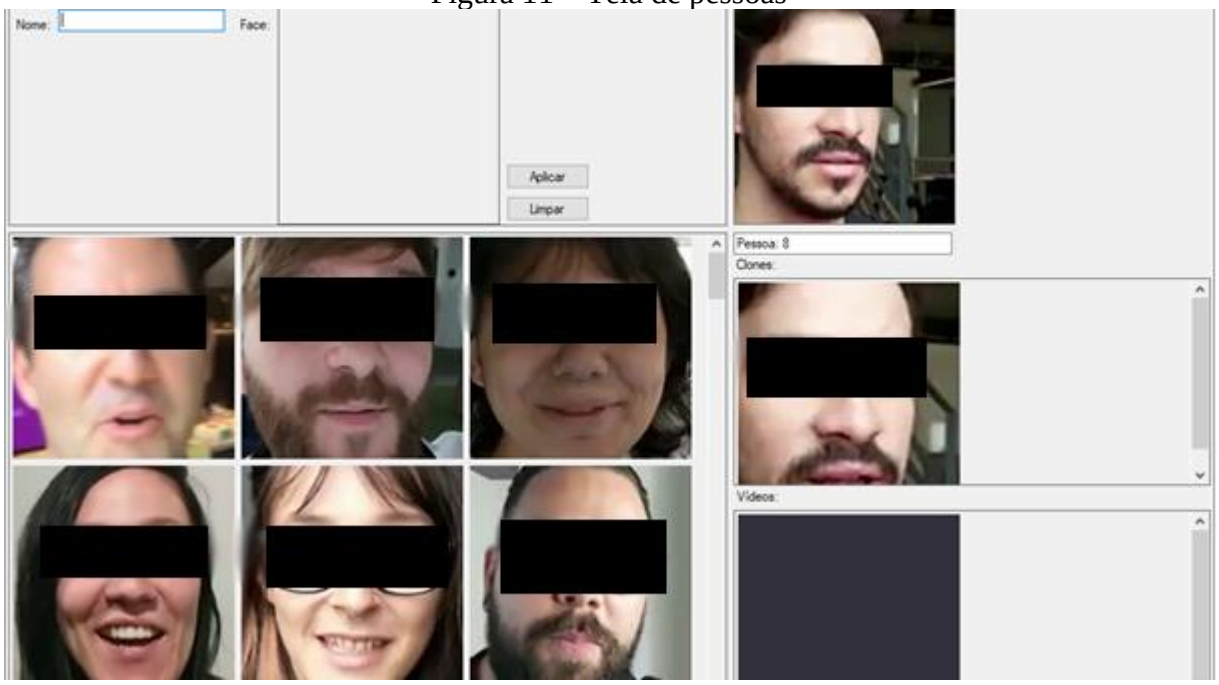
Figura 10 – Tela de configurações



Fonte: elaborado pelo autor.

O botão `Pessoas` abre a tela com as pessoas reconhecidas nos vídeos já indexados. Nessa tela é possível realizar associação/junção de duplicidades/clones (quando a mesma pessoa é identificada duas vezes como sendo pessoas diferentes), dar um nome para cada pessoa, filtrar por nome ou imagem da face e visualizar os vídeos associados a cada pessoa (os vídeos onde a pessoa reconhecida aparece). A Figura 11 apresenta a tela de pessoas após um vídeo ter sido completamente indexado.

Figura 11 – Tela de pessoas

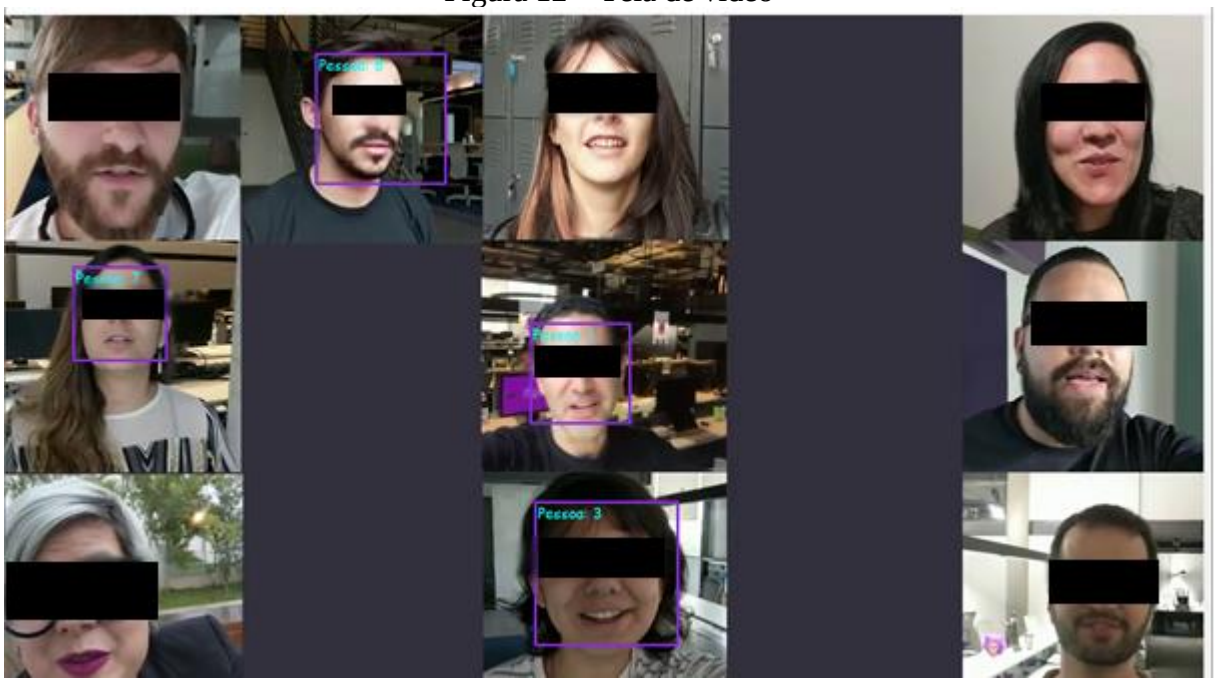


Fonte: elaborado pelo autor.

No painel superior esquerdo é possível filtrar a busca por nome e face de uma pessoa. A face é selecionada arrastando uma imagem do tipo *png* sobre o respectivo quadrado. O painel inferior esquerdo exibe as faces das pessoas detectadas. Quando duas faces forem consideradas diferentes pelo algoritmo, mas o usuário as considerar iguais, é possível arrastar uma face sobre a outra para adicionar um relacionamento entre elas. Esse relacionamento entre faces iguais, é chamado de clone. Todos os clones relacionados a uma pessoa são exibidos me um painel abaixo do nome da pessoa selecionada. Por último, o painel direito exibe a foto da pessoa selecionada e permite efetuar alteração do nome, remover clones e visualizar os vídeos em que a pessoa foi detectada.

Ao dar um duplo clique em um dos vídeos, o vídeo é aberto para visualização, na primeira posição do vídeo onde a pessoa selecionada é reconhecida. Como pode ser visto na Figura 12. A tecla seta direita do teclado avança para o próximo *frame* onde a pessoa é reconhecida.

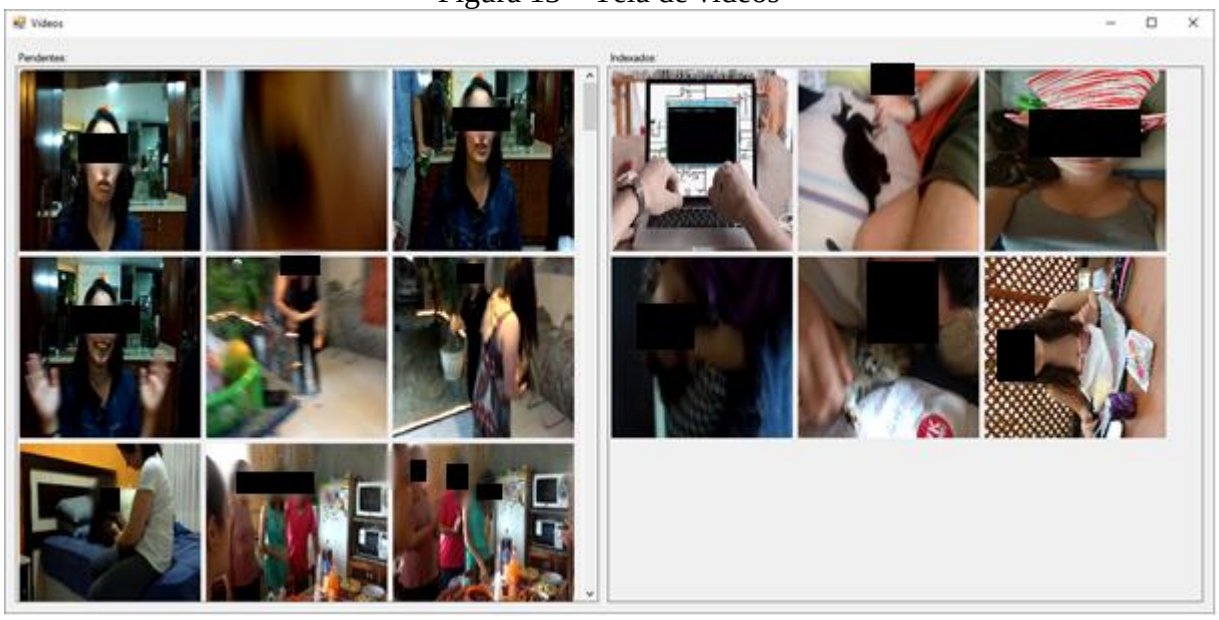
Figura 12 – Tela de vídeo



Fonte: elaborado pelo autor.

Por fim, o botão `Videos` apresenta a tela de vídeos. A Figura 13 apresenta a tela de vídeos. A tela é separada por dois painéis, sendo o da esquerda com os vídeos pendentes de indexação e o da direita de vídeos já indexados. Enquanto a tela de vídeos está aberta, é executada uma tarefa em segundo plano para indexar o próximo vídeo pendente. Com um duplo clique em um dos vídeos é aberta a tela de visualização do vídeo. Durante a reprodução do vídeo a tecla seta direita do teclado avança para o próximo *frame* onde há alguma pessoa reconhecida.

Figura 13 – Tela de vídeos



Fonte: elaborado pelo autor.

3.4 ANÁLISE DOS RESULTADOS

Essa seção apresenta os experimentos realizados com a ferramenta. A seção 3.4.1 mostra os resultados da avaliação da experiência do usuário. Na seção 3.4.2 são apresentados os resultados da avaliação da biblioteca por parte de desenvolvedores. A seção 3.4.3 demonstra os resultados das análises de memória e performance da ferramenta. Os experimentos da experiência do usuário e do desenvolvedor foram realizados através de formulário disponibilizado no Apêndice A. Por fim, a seção 3.4.4 apresenta a comparação entre a ferramenta e os trabalhos correlatos mostrados no capítulo anterior.

3.4.1 Experiência do usuário

O experimento de experiência dos usuários foi respondido por 5 voluntários. A primeira seção do questionário analisou o perfil dos voluntários. Na Tabela 1 é possível visualizar as respostas do perfil dos usuários.

Tabela 1 – Perfil dos usuários

Sexo?	80% masculino 20% feminino
Idade?	80% entre 21 a 25 anos 20% mais de 30 anos
Nível de escolaridade?	60% superior incompleto 40% superior completo
Com que frequência você faz uso de aparelhos que capturam vídeos (celular, câmera digital, webcam etc)?	60% frequentemente 40% às vezes
Com que frequência você precisa buscar por um vídeo em que uma determinada pessoa aparece?	80% às vezes 20% nunca precisou
Você já utilizou alguma ferramenta para auxiliar na indexação de vídeos com base nas pessoas presentes neles?	100% nunca utilizou
Grau de familiaridade com Detecção Facial?	20% nunca ouviu falar 80% conhece, mas nunca utilizou
Grau de conhecimento sobre Reconhecimento Facial?	40% nenhum 60% conhece um pouco

Fonte: elaborado pelo autor.

Como pode ser visto nos dados da Tabela 1, a maioria dos voluntários é do sexo masculino, assim como a maioria também está na faixa estaria dos 21 aos 25 anos. Todos os voluntários já possuem escolaridade superior ou está em andamento. A maior parte dos voluntários utiliza frequentemente aparelhos que geram conteúdo de vídeo digital. Nenhum deles havia utilizado alguma ferramenta de indexação de vídeos com base nas faces das pessoas, mas a maioria deles já passou por situações de precisar buscar um vídeo em que uma determinada pessoa aparece. A maioria deles já ouviu falar de detecção facial, mas nunca

havia utilizado. Por fim, o reconhecimento facial é um tópico que mais da metade conhece um pouco.

A primeira seção de orientações para avaliar a experiência do usuário utilizou um vídeo fornecido, com resultados já esperados. Na Tabela 2 são exibidas as respostas as orientações.

Tabela 2 – Respostas das instruções com vídeo fornecido

Clique no botão "Vídeos" e dê dois cliques no vídeo disponível. É possível visualizar um retângulo ao redor das faces e um nome gerado automaticamente?	100% sim
Volte ao menu principal e clique no botão "Pessoas". É possível visualizar as pessoas reconhecidas no vídeo visualizado?	100% sim
Dê um duplo clique na foto da primeira mulher com óculos. Em seguida dê um duplo clique no vídeo disponível no painel inferior direito. Utilize a seta direita do teclado para avançar os frames. É possível visualizar os frames em que a pessoa selecionada aparece?	100% sim
Quando uma pessoa é identificada duas vezes separadamente (a ferramenta acredita ser pessoas diferentes), é possível realizar o vínculo das duas pessoas, para que sejam consideradas a mesma pessoa. Clique e arraste uma face para cima da outra, para que o vínculo seja realizado. Dê um duplo clique para visualizar o vínculo no painel direito, chamado "Clones". Em seguida dê um duplo clique no vídeo e veja se os frames exibidos contêm as duas pessoas reconhecidas. Foi possível realizar o vínculo corretamente?	100% sim

Fonte: elaborado pelo autor.

Com base nas respostas da Tabela 2, é possível verificar que todos os voluntários conseguiram utilizar a ferramenta e validar os resultados a partir de um vídeo fornecido. Um dos usuários fez uma observação a respeito da taxa de assertividade da detecção facial, sendo que não foram todas as faces que foram detectadas e marcadas com retângulo ao redor do rosto.

Após as instruções com vídeo fornecido, para que o usuário possa se ambientar com a ferramenta, iniciou o segundo passo do questionário. Na segunda fase o usuário poderia utilizar a ferramenta com qualquer conjunto de vídeos de sua preferência. O objetivo dessa seção é verificar se a ferramenta é flexível para outros tipos de vídeos, em outras qualidades, com outras faces, iluminação e outros fatores que podem impactar no seu funcionamento. Na Tabela 3 são exibidas as respostas da seção de uso livre.

Tabela 3 – Respostas das instruções do modo livre

Copie vídeos de sua preferência para o diretório "Vídeos", dentro do diretório "TCC". Se a extensão dos vídeos for diferente de "mp4", ajuste a configuração na tela de "Configurações". Em seguida abra a tela de Vídeos, aguarde seus vídeos serem indexados. Você pode acompanhar a indexação enquanto ela está sendo executada (duplo clique no vídeo).	100% sim
Abra a tela de "Pessoas" e veja se foram identificadas pessoas contidas nos vídeos que você selecionou. Se houverem repetições da mesma pessoa, faça o vínculo delas. Dê duplo clique em uma das pessoas e veja se a visualização do vídeo é exibida diretamente nos momentos em que a pessoa selecionada aparece.	100% sim

Fonte: elaborado pelo autor.

Com base nas respostas da Tabela 3, é possível verificar que a ferramenta se mostrou flexível para ser utilizada de modo livre pelos usuários, com diferentes tipos de vídeos. Por fim, na última seção do questionário, foram realizadas perguntas referentes a usabilidade da ferramenta. A Tabela 4 apresenta as respostas sobre usabilidade.

Tabela 4 – Respostas sobre usabilidade

Das atividades solicitadas, quantas atividades você conseguiu executar sem auxílio?	60% todas 40% a maior parte
De modo geral, você achou a ferramenta intuitiva e fácil de usar?	100% sim
Se você nunca havia utilizado uma ferramenta para auxiliar na indexação de vídeos com base nas faces detectadas, após o uso dessa ferramenta, você voltaria a utilizar?	100% sim
Você acha que uma ferramenta para indexar vídeos com base nas faces das pessoas contidas, auxilia na busca e organização de vídeos?	100% sim

Fonte: elaborado pelo autor.

Baseado nas respostas da Tabela 4, é possível verificar que mais da metade conseguiu executar as tarefas sem auxílio. Um dos usuários observou que conseguiu utilizar a ferramenta facilmente a partir das instruções do questionário, mas apontou que sem as instruções provavelmente demoraria um pouco para entender como a ferramenta funciona.

3.4.2 Experiência do desenvolvedor

Nesta seção são apresentados os resultados da experiência de 2 desenvolvedores ao utilizar a biblioteca disponibilizada. O objetivo do experimento foi avaliar o reaproveitamento da biblioteca desenvolvida por parte de outros desenvolvedores. O reaproveitamento da biblioteca é um fator interessante por permitir a fácil extensão deste trabalho. Na seção 3.4.2.1

é apresentado um tutorial de como utilizar a biblioteca. Inicialmente foi questionado o perfil dos desenvolvedores, cujas respostas podem ser vistas na Tabela 5.

Tabela 5 – Perfil dos desenvolvedores

Sexo?	100% masculino
Idade?	100% entre 21 a 25 anos
Nível de escolaridade?	100% superior incompleto
Com que frequência você faz uso de aparelhos que capturam vídeos (celular, câmera digital, webcam etc)?	50% frequentemente 50% às vezes
Com que frequência você precisa buscar por um vídeo em que uma determinada pessoa aparece?	50% às vezes 50% nunca precisou
Você já utilizou alguma ferramenta para auxiliar na indexação de vídeos com base nas pessoas presentes neles?	100% nunca utilizou
Grau de familiaridade com Detecção Facial?	100% conhece, mas nunca utilizou
Grau de conhecimento sobre Reconhecimento Facial?	100% conhece um pouco

Fonte: elaborado pelo autor.

Como é possível ver na Tabela 5, todos os 2 desenvolvedores são do gênero masculino, estão entre 21 a 25 anos e possuem escolaridade superior incompleta. Um deles faz uso frequente de aparelhos que capturam vídeos, enquanto o outro utiliza apenas às vezes. Um deles reportou que às vezes precisa efetuar busca de um vídeo em que uma determinada pessoa aparece, o outro nunca precisou. Nenhum dos entrevistados havia utilizado ferramentas de indexação de vídeos a partir das faces das pessoas. Os desenvolvedores possuem familiaridade com detecção facial, mas nunca utilizaram na prática. Por fim, o grau de conhecimento sobre reconhecimento facial é médio, sendo que ambos conhecem apenas um pouco do assunto. Na Tabela 6 são mostradas as respostas relacionadas às orientações do questionário.

Tabela 6 – Respostas das instruções do questionário

Abra o Visual Studio 2017 e crie um projeto do tipo "Windows Forms App (.NET Framework)" e compile em "Release"	100% sim
Copie todo o conteúdo extraído do arquivo "TCC.zip" baixado no início do questionário para dentro do diretório "bin\Release" do seu projeto.	100% sim
No Visual Studio, na janela de "Solution Explorer", clique com o botão direito em "References" e em seguida em "Add Reference". No painel esquerdo, selecione "Browse", pressione o botão "Browse..." e navegue para dentro do diretório "bin\Release" de seu projeto. Selecione todas as DLLs disponíveis e pressione "Add".	100% sim
Na janela "Toolbox", clique com o botão direito e pressione "Choose Items...". Na janela aberta, pressione o botão "Browse..." e selecione a DLL "Emgu.CV.UI.dll", que está dentro do diretório "bin\Release" de seu projeto. Pressione OK. Deverá aparecer a opção "ImageBox" dentre as opções da janela "Toolbox" do Visual Studio.	100% sim
Arraste um componente "ImageBox" para a janela "Form1.cs", que é a janela principal da sua aplicação. Na janela "Properties", com o componente "ImageBox" selecionado, altera o valor de "Dock" para "Fill".	100% sim
Abra o arquivo Form1.cs. Crie uma variável membro na classe, do tipo "Recognizer". No construtor da classe "Form1", após o método "InitializeComponent()", adicione a linha "recognizer = new Recognizer(@"Vídeos\Como os clientes Nus mudaram a nossa vida.mp4", imageBox1);". Após o construtor, adicione o trecho de código: "protected override void OnClosed(EventArgs e) { base.OnClosed(e); DataFile.SaveFiles(); }". Seu arquivo Form1.cs deve estar semelhante a imagem abaixo. Em seguida, compile e execute a aplicação.	100% sim
Foi possível visualizar o vídeo durante o processo de detecção e reconhecimento facial?	100% sim

Fonte: elaborado pelo autor.

Com pode ser visto nas respostas da Tabela 6, foi possível reaproveitar as rotinas desenvolvidas e encapsuladas em uma biblioteca neste trabalho, por outros desenvolvedores em outras aplicações. Um dos desenvolvedores fez uma observação em relação a IDE, onde foi orientado criar um projeto utilizando o Visual Studio versão 2017. Segundo o desenvolvedor, foi possível efetuar todos os passos utilizando o Visual Studio na versão 2015.

3.4.2.1 Biblioteca

Para efetuar uso da biblioteca, é necessário ter uma instalação do Visual Studio 2017, com o framework de desenvolvimento em C# e dotNet adicionado. O uso da biblioteca pode ser feito em um novo projeto ou em projetos já existentes. A restrição é de que o projeto seja do tipo `Windows Forms App (.NET Framework)`. A biblioteca disponibilizada está em um pacote compactado com os binários necessários. Este pacote deve ser descompactado dentro do diretório `Release` do projeto que será utilizado.

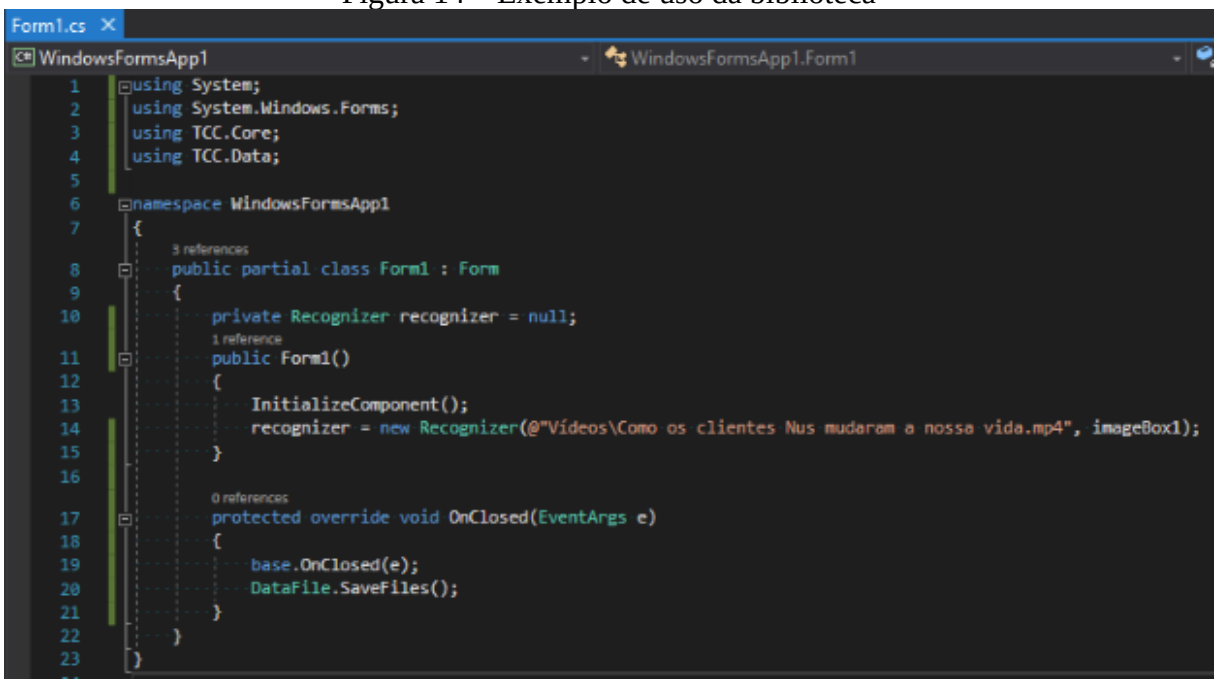
Na janela de `Solution Explorer` é preciso clicar com o botão direito na opção `References` e em seguida em `Add Reference`. No painel esquerdo, o desenvolvedor deve selecionar `Browse`, pressionar o botão `Browse...` e navegar para dentro do diretório `bin\Release` de seu projeto. Por fim, é necessário selecionar todas as DLLs disponíveis e pressionar o botão `Add`.

Na janela `Toolbox`, o desenvolvedor deve clicar com o botão direito e pressionar o botão `Choose Items...`. Em seguida, na janela que abrirá, pressionar o botão `Browse...` e selecionar a DLL `Emgu.CV.UI.dll`, que está dentro do diretório `bin\Release` de seu projeto. Ao pressione o botão `OK`, aparecerá a opção `ImageBox` dentre as opções da janela `Toolbox` do Visual Studio.

Se for necessário acompanhamento da reprodução do vídeo pelo usuário, pode ser arrastado um componente `ImageBox` para dentro do `form` onde a visualização do vídeo é necessária. Na janela `Properties`, com o componente `ImageBox` selecionado, deve-se alterar o valor de `Dock` para `Fill`.

No código fonte do `form` em que foi incluído o componente `ImageBox`, é preciso instanciar um objeto do tipo `Recognizer`. A classe `Recognizer` recebe dois parâmetros, sendo o primeiro o caminho do vídeo a ser indexado e o segundo o `ImageBox` onde será exibido o acompanhamento do vídeo em tempo real. O segundo parâmetro é opcional. A Figura 14 apresenta um exemplo de uso da biblioteca.

Figura 14 – Exemplo de uso da biblioteca

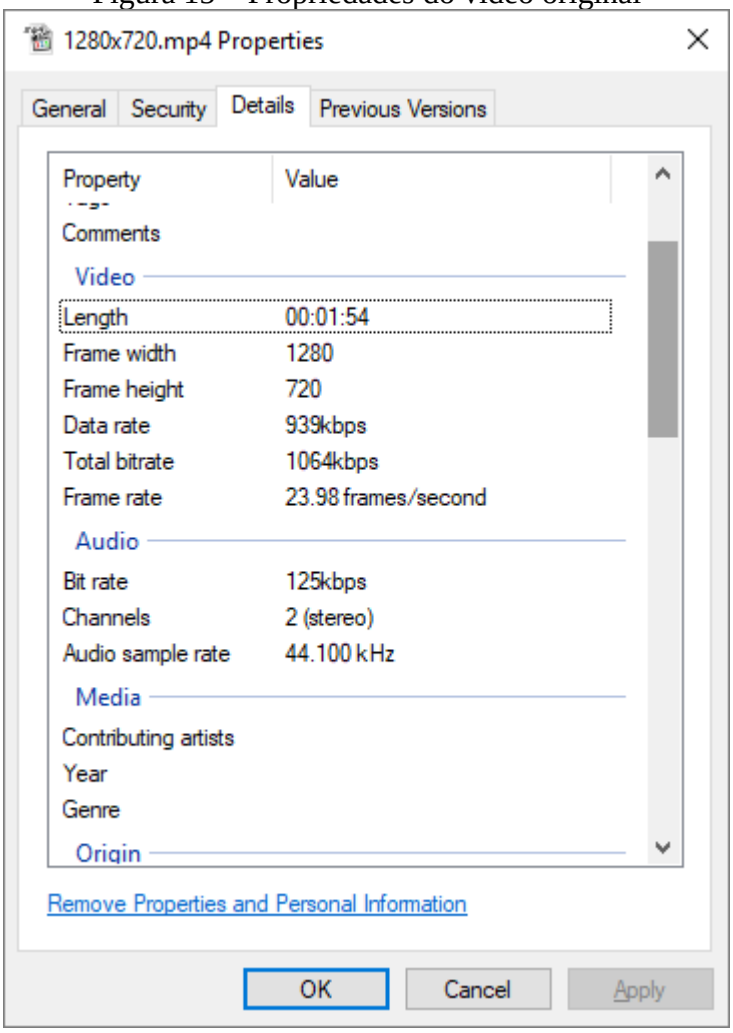


Fonte: elaborado pelo autor.

3.4.3 Análise de memória e performance

Esta seção apresenta a análise dos picos de uso de memória e da performance da ferramenta. A base de teste será a mesma para as duas análises. Foi utilizado como base um vídeo comercial da empresa Nubank. A partir desse vídeo foram geradas variações de tempo e de resolução utilizando a ferramenta Shotcut versão 18.01.02. Na Figura 15 são exibidas as propriedades do vídeo original, que possui um tamanho total de 14.6MB.

Figura 15 – Propriedades do vídeo original



Fonte: elaborado pelo autor.

Foram realizadas quatro variações de tempo do vídeo. Elas são apresentadas na Tabela 7. Os tempos foram cortados numa relação de ¼, ficando com os tamanhos original, ¾, ½ e ¼.

Tabela 7 – Variações de tempo

Vídeo	Tempo
Original	01:54
¾	01:25
½	00:57
¼	00:29

Fonte: elaborado pelo autor.

A resolução também foi variada em quatro partes, com a mesma relação original, ¾, ½ e ¼, como apresentado na Tabela 8.

Tabela 8 – Variações de resolução

Vídeo	Resolução x	Resolução y
Original	1280	720
¾	960	540
½	640	360
¼	320	180

Fonte: elaborado pelo autor.

3.4.3.1 Análise de memória

A primeira análise verifica a variação de consumo de memória em relação ao tempo de duração dos vídeos. Na Tabela 9 são apresentados os resultados dos testes.

Tabela 9 – Resultado memória x tempo

Vídeo	Memória (MB)
Original	951.9
¾	881.1
½	719.1
¼	662.4

Fonte: elaborado pelo autor.

A segunda análise avalia a variação da memória em relação a resolução dos vídeos. A Tabela 10 apresenta os resultados dos testes.

Tabela 10 – Resultado memória x resolução

Vídeo	Memória (MB)
Original	951.9
¾	718.3
½	314.1
¼	149.9

Fonte: elaborado pelo autor.

3.4.4 Análise de performance

Inicialmente é analisada a variação de performance em relação ao tempo de duração dos vídeos. A Tabela 11 apresenta os resultados dos testes.

Tabela 11 – Resultado performance x tempo

Vídeo	Tempo (seg)
Original	252
¾	173
½	115
¼	51

Fonte: elaborado pelo autor.

Por fim, é analisada a variação da performance em relação a resolução dos vídeos. A Tabela 12 apresenta os resultados dos testes.

Tabela 12 – Resultado performance x resolução

Vídeo	Tempo (seg)
Original	252
¾	161
½	89
¼	31

Fonte: elaborado pelo autor.

3.4.5 Comparação entre a ferramenta e os trabalhos correlatos

No Quadro 14 é apresentado um comparativo entre os trabalhos correlatos. As linhas representam as características e as colunas os trabalhos.

Quadro 14 – Comparativo entre a ferramenta e os trabalhos correlatos

Trabalhos / Características	Koch (2012)	Jiang et al. (2016)	Video Indexer (2017)	Trabalho desenvolvido (2018)
ferramenta reaproveitável (API)	Não	Não	Sim	Sim
processamento distribuído	Sim	Não	Sim	Não
efetua identificação	Sim	Não	Sim	Sim
realiza detecção facial em tempo real	Sim	Sim	Não	Sim

Fonte: elaborado pelo autor.

Como pode ser observado no Quadro 14, os trabalhos de Koch (2012) e Jiang et al. (2016) não têm possibilidade de reaproveitamento das funcionalidades em outra aplicação, no formato de ferramenta ou API. A aplicação da Microsoft, Video Indexer e este trabalho, possibilitam o reaproveitamento das funcionalidades em outros aplicativos, bastando utilizar a API ou biblioteca disponibilizada. Percebe-se também que o trabalho de Jiang et al. (2016) e este trabalho não efetuam processamento distribuído. A aplicação Video Indexer funciona como serviço, realizando o processamento sob demanda nos servidores *cloud* da Microsoft de forma parecida com o trabalho de Koch (2012), que realiza o processamento no servidor onde está hospedado o site ou aplicação.

Referente à identificação e à possibilidade de busca ou comparação entre faces, apenas o protótipo de Jiang et al. (2016) não possui essa funcionalidade. Na aplicação de Koch (2012), enquanto a câmera está aberta e até que o usuário pressione o botão para realizar a autenticação, é exibido um retângulo ao redor da face reconhecida, constatando a realização da detecção facial em tempo real, assim como no trabalho de Jiang et al. (2016) e neste trabalho. O aplicativo da Microsoft, Video Indexer, não permite a visualização da detecção facial em tempo real, apenas informa em quais pontos do vídeo determinada face foi reconhecida.

4 CONCLUSÕES

Este trabalho propôs o desenvolvimento de uma ferramenta para realizar indexação de vídeos com base nas faces humanas detectadas e reconhecidas. Foi utilizada a linguagem de programação C# com a biblioteca de processamento de imagens EmguCV. O objetivo geral do trabalho foi atingido e os resultados dos testes comprovam isto. Os objetivos específicos de disponibilizar uma biblioteca reaproveitável e de disponibilizar uma coleção com as faces reconhecidas em cada vídeo foram atingidos, comprovados com os testes realizados pelos desenvolvedores.

Apenas um dos objetivos específicos inicialmente elencado não foi atingido, sendo o objetivo de possibilitar marcar/identificar um rosto durante a reprodução de um vídeo. A intenção era permitir que o usuário apontasse onde há um rosto, nas situações em que o algoritmo utilizado não conseguisse detectar a face. Contudo, a biblioteca EmguCV utilizada não permite efetuar um retreinamento do classificador de faces. O classificador é baseado em um arquivo XML computado com um tempo considerável, com base em uma base de dados de faces. Como o retreinamento do classificador implicaria na interpretação e edição deste arquivo XML durante a execução da aplicação, considerou-se uma atividade fora do contexto do trabalho proposto. A possibilidade de retreinamento do classificador durante a classificação pode ser um assunto interessante para trabalhos futuros.

As ferramentas utilizadas se mostraram parcialmente adequadas para os fins utilizados. Percebeu-se em vários momentos, que por conta da biblioteca EmguCV ser uma versão simplificada e de mais alto nível se comparada à original OpenCV em C e C++, houve dificuldade ao acessar métodos e atributos de baixo nível internos da biblioteca OpenCV. Inicialmente foi utilizado o framework WPF da linguagem C# para desenvolver a interface gráfica. Contudo, a biblioteca EmguCV não é diretamente compatível com WPF, apenas com o framework WinForms. O uso da biblioteca Json.NET foi muito útil para a persistência, de fácil uso e com uma boa performance.

Por fim, conclui-se que a ferramenta desenvolvida atingiu o objetivo principal, mas possui certas limitações na acurácia da detecção e do reconhecimento das faces. Algumas faces não são detectadas, seja por impacto da iluminação, angulação ou foco da imagem. Em certos momentos, faces detectadas são consideradas de pessoas diferentes, quando na verdade são da mesma pessoa. Essas limitações apontam a necessidade de um estudo aprofundado para encontrar ou desenvolver melhores algoritmos para as fases da detecção e da identificação das faces.

4.1 EXTENSÕES

São sugeridos como extensões para trabalhos futuros:

a) utilizar ou desenvolver algoritmo com melhor acurácia na detecção das faces;
b) avaliar melhores métodos de classificação das pessoas;
c) utilizar algoritmo para a detecção com recursos da GPU, para obter melhores resultados na performance;
d) desenvolver uma interface mais intuitiva para o usuário final;
e) avaliar a possibilidade do retreinamento do classificador de faces, para permitir que o usuário final possa auxiliar a ferramenta nos momentos em que a ferramenta não conseguir detectar uma face.

REFERÊNCIAS

BRAGA, Luis Felipe Zenicola. **Sistemas de Reconhecimento Facial**. 2013. 84f. Trabalho de Conclusão de Curso (Engenharia Elétrica com ênfase em Eletrônica) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Paulo.

DINIZ, Fábio Abrantes; NETO, Francisco Milton Mendes; LIMA JÚNIOR, Francisco das Chagas; FONTES, Laysa Mabel de Oliveira. RedFace: um sistema de reconhecimento facial baseado em técnicas de análise de componentes principais e autofaces: comparação com diferentes classificadores. **Revista Brasileira de Computação Aplicada**, Passo Fundo, v. 5, n. 1, p. 42-54, abr. 2013.

DINIZ, Fabio Abrantes; SILVA, Thiago Reis da; ALENCAR, Francisco Eduardo Silva. Um estudo empírico de um sistema de reconhecimento facial utilizando o classificador KNN. **Revista Brasileira de Computação Aplicada**, Passo Fundo, v. 8, n. 1, p. 50-63, abr. 2016.

FARNESE, Rafael. **Recuperação de quadros de arquivos de vídeo H.264/AVC corrompidos**. 2012. 130 f. Dissertação (Mestrado em Engenharia Elétrica) – Departamento de Engenharia Elétrica, Universidade de Brasília, Brasilia.

GADA, Milan. **Introducing Video Indexer, a cloud service to unlock insights from your videos**. Redmond, 2017.

JIANG, Xiaodong; YU, Hui; LU, Yang; LIU, Honghai. A fusion method for robust face tracking. **Multimedia Tools and Applications**, Massachusetts, v. 75, n. 19, p. 11801-11813, out. 2016.

KOCH, Márcio. **Visão computacional para reconhecimento de faces aplicado na identificação e autenticação de usuários na web**. 2012. 139 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

KORNICH, Julia; FOWLER, Cory. **Video Indexer (preview)**. Redmond, 2017. Disponível em: <https://docs.microsoft.com/en-us/azure/cognitive-services/video-indexer/video-indexer-overview>; Acesso em: 9 set 2017.

NISSELSON, Evan; HUNTER-SYED, Abigail; SHAH, Sadhana. **45 Billion Cameras by 2022 Fuel Business Opportunities**: Five Year Visual Technology Market Analysis. New York, 2017.

ORTS, Jorge. **Face Recognition Techniques**. Madison, 2016. Disponível em: < https://homepages.cae.wisc.edu/~ece533/project/f06/orts_rpt.pdf>. Acesso em: 3 set 2017.

PASSARINHO, Cornelia Janayna Pereira; SALLES, Evandro Ottoni Teatini; SARCINELLI FILHO, Mario. Face Tracking in Unconstrained Color Videos with the Recovery of the Location of Lost Faces. **IEEE Latin America Transactions**, São Paulo, v. 13, n. 1, p. 307-314, jan. 2015.

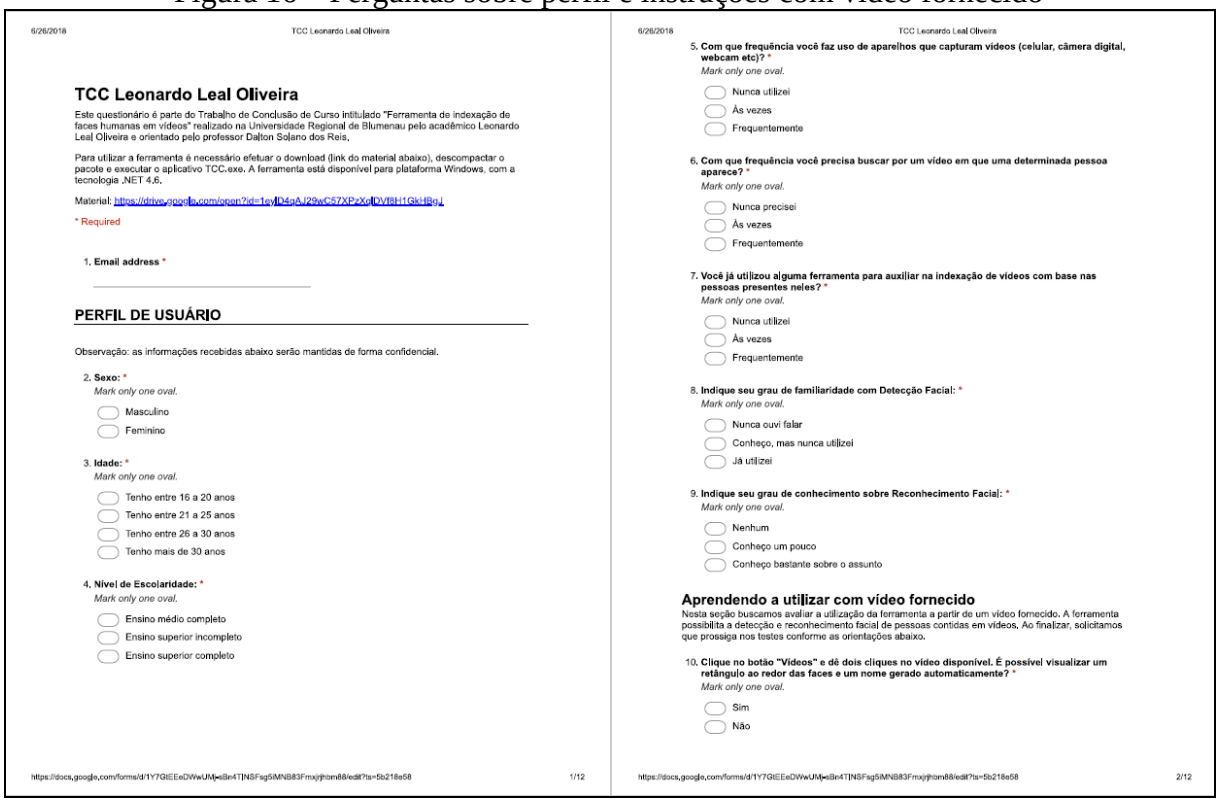
ROSA, André Luís Beling da. **Sistema de Tracking de Objetos a Partir de Várias Câmeras**. 2010. 48 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SILVA, Leandro Max Lima. **Implementação Física de Arquiteturas de Hardware para a Decodificação de Vídeo Digital Segundo o Padrão H.264/AVC**. 2010. 136 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

APÊNDICE A – Questionário de teste da ferramenta

Neste apêndice é apresentado o questionário realizado para o experimento de teste da ferramenta. A Figura 16 mostra as perguntas sobre o perfil do usuário e o início da seção de instruções com vídeo fornecido. As Figura 17 e 25 dão continuidade as instruções com vídeo fornecido e a Figura 18 dá início modo de teste livre. Na Figura 19 apresenta o questionário de usabilidade e dá início as questões direcionadas para usuários do perfil desenvolvedor. Por fim, as Figura 20 e 28 mostram as últimas questões para desenvolvedores.

Figura 16 – Perguntas sobre perfil e instruções com vídeo fornecido



6/26/2018 TCC Leonardo Leal Oliveira

TCC Leonardo Leal Oliveira

Este questionário é parte do Trabalho de Conclusão de Curso intitulado "Ferramenta de indexação de faces humanas em vídeos" realizado na Universidade Regional de Blumenau pelo acadêmico Leonardo Leal Oliveira e orientado pelo professor Dalton Solano dos Reis.

Para utilizar a ferramenta é necessário efetuar o download (link do material abaixo), descompactar o pacote e executar o aplicativo TCC.exe. A ferramenta está disponível para plataforma Windows, com a tecnologia .NET 4.6.

Material: https://drive.google.com/open?id=1eyJD4gAJ29wC57XPzXpIDVf8H1GkHBoJ

* Required

1. **Email address** *

PERFIL DE USUÁRIO

Observação: as informações recebidas abaixo serão mantidas de forma confidencial.

2. **Sexo:** *
Mark only one oval.
- Masculino
- Feminino

3. **Idade:** *
Mark only one oval.
- Tenho entre 16 a 20 anos
- Tenho entre 21 a 25 anos
- Tenho entre 26 a 30 anos
- Tenho mais de 30 anos

4. **Nível de Escolaridade:** *
Mark only one oval.
- Ensino médio completo
- Ensino superior incompleto
- Ensino superior completo

https://docs.google.com/forms/d/1Y7GtEEeDWwUMjw8n4TlNSFsg5iMNB83Fmxjrjhbm88/edit?ts=5b218e58 1/12

6/26/2018 TCC Leonardo Leal Oliveira

5. **Com que frequência você faz uso de aparelhos que capturam vídeos (celular, câmera digital, webcam etc)?** *
Mark only one oval.
- Nunca utilizei
- Às vezes
- Frequentemente

6. **Com que frequência você precisa buscar por um vídeo em que uma determinada pessoa aparece?** *
Mark only one oval.
- Nunca precisei
- Às vezes
- Frequentemente

7. **Você já utilizou alguma ferramenta para auxiliar na indexação de vídeos com base nas pessoas presentes neles?** *
Mark only one oval.
- Nunca utilizei
- Às vezes
- Frequentemente

8. **Indique seu grau de familiaridade com Detecção Facial:** *
Mark only one oval.
- Nunca ouvi falar
- Conheço, mas nunca utilizei
- Já utilizei

9. **Indique seu grau de conhecimento sobre Reconhecimento Facial:** *
Mark only one oval.
- Nenhum
- Conheço um pouco
- Conheço bastante sobre o assunto

Aprendendo a utilizar com vídeo fornecido

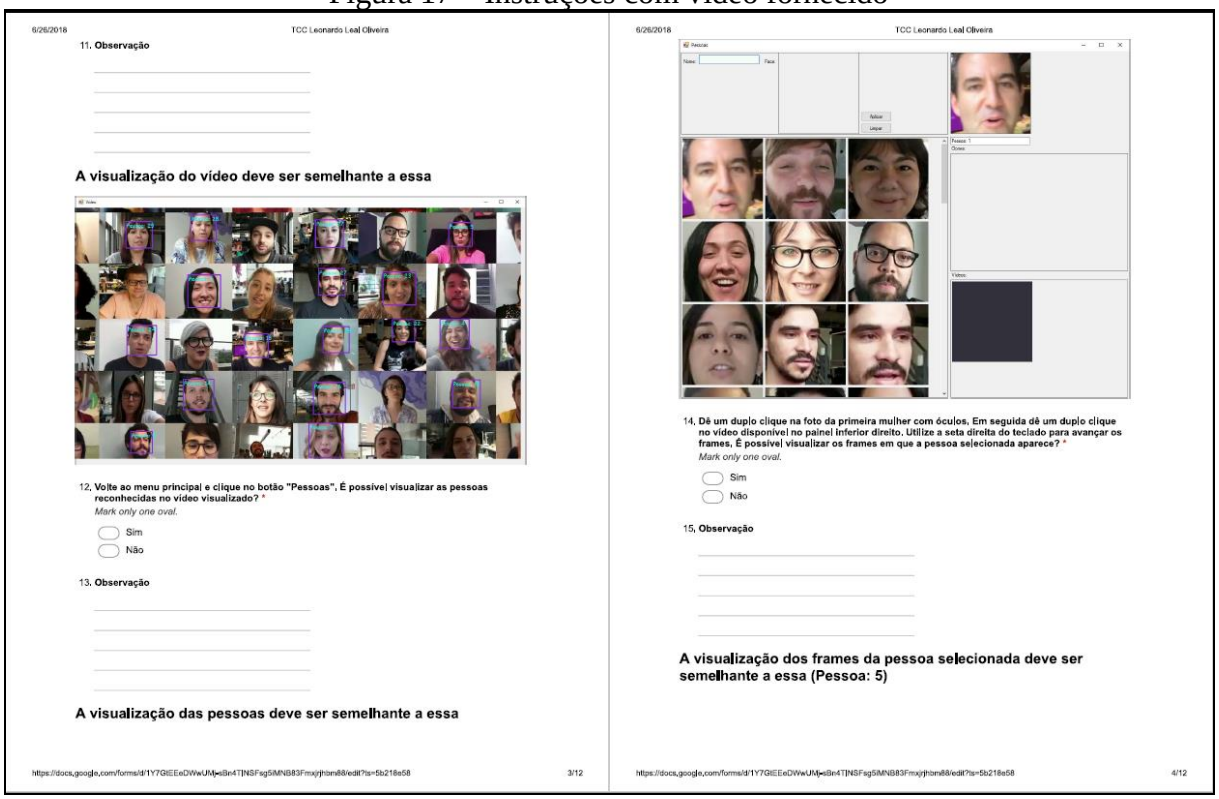
Nesta seção buscamos avaliar a utilização da ferramenta a partir de um vídeo fornecido. A ferramenta possibilita a detecção e reconhecimento facial de pessoas contidas em vídeos. Ao finalizar, solicitamos que prossiga nos testes conforme as orientações abaixo.

10. **Clique no botão "Vídeos" e dê dois cliques no vídeo disponível. É possível visualizar um retângulo ao redor das faces e um nome gerado automaticamente?** *
Mark only one oval.
- Sim
- Não

https://docs.google.com/forms/d/1Y7GtEEeDWwUMjw8n4TlNSFsg5iMNB83Fmxjrjhbm88/edit?ts=5b218e58 2/12

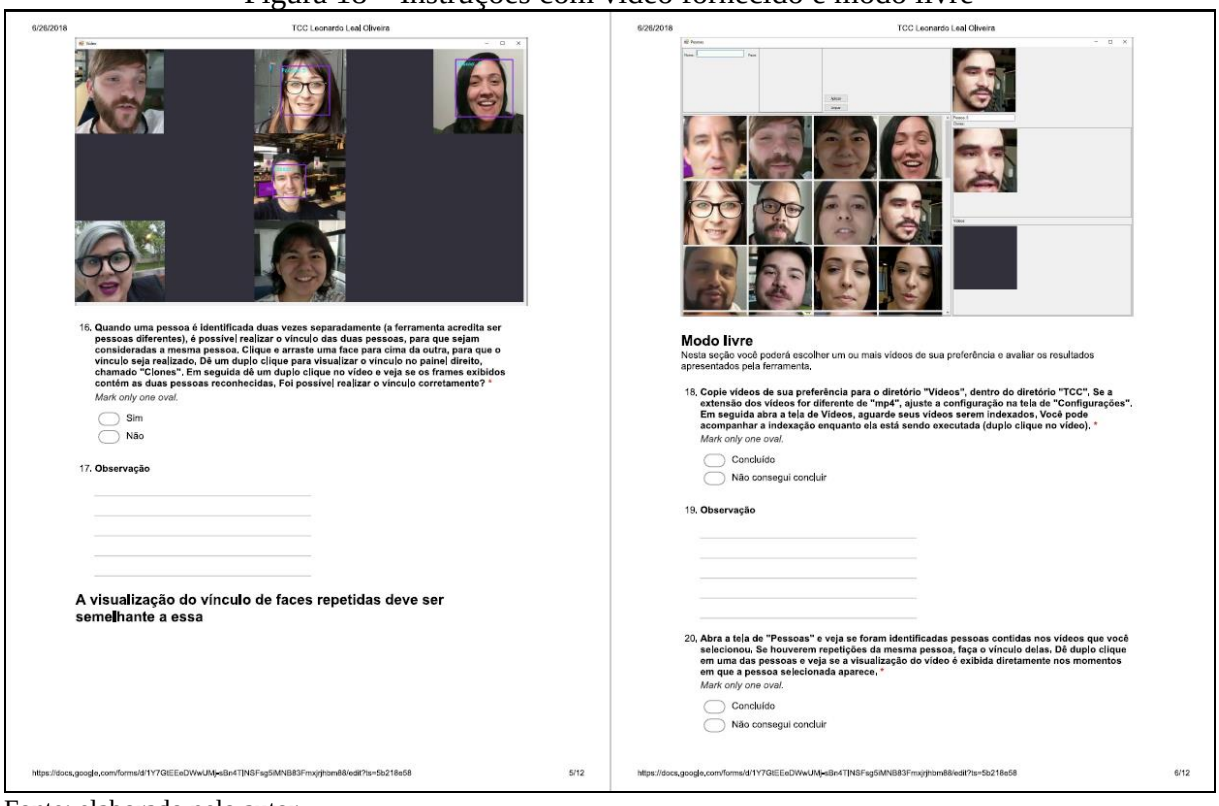
Fonte: elaborado pelo autor.

Figura 17 – Instruções com vídeo fornecido



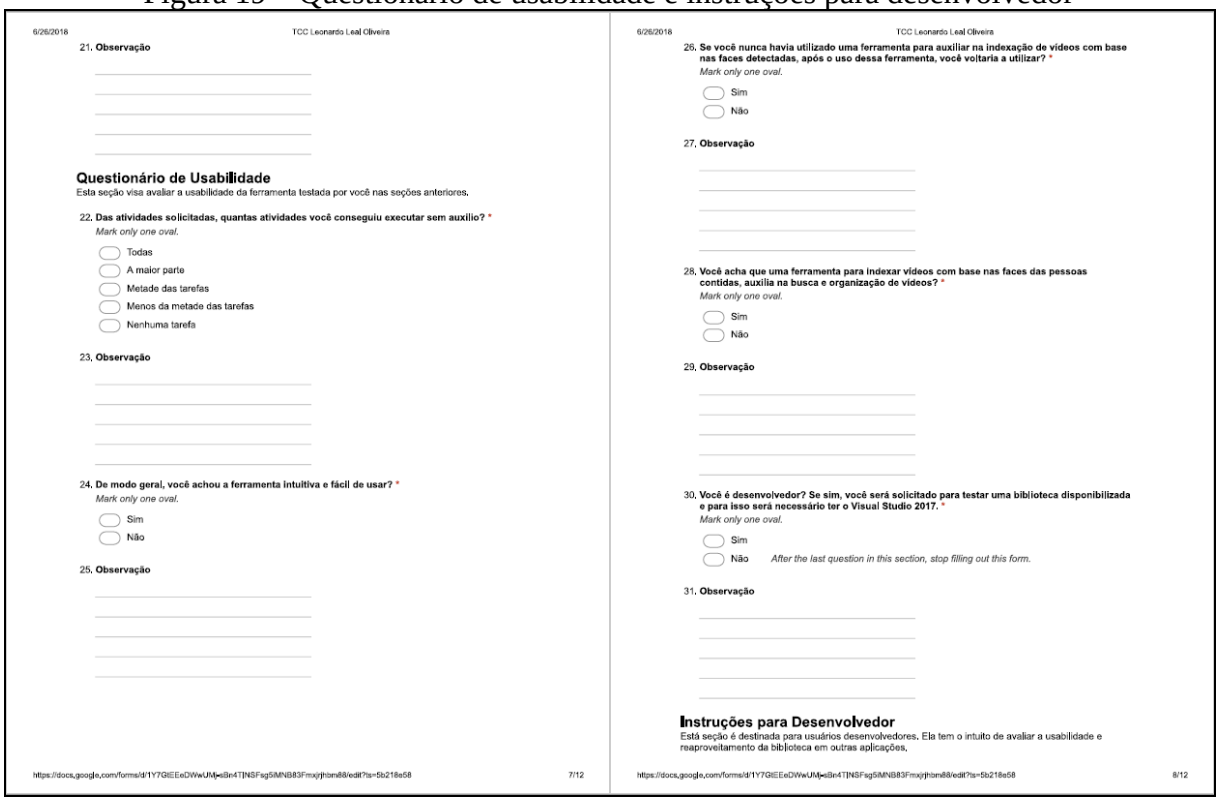
Fonte: elaborado pelo autor.

Figura 18 – Instruções com vídeo fornecido e modo livre



Fonte: elaborado pelo autor.

Figura 19 – Questionário de usabilidade e instruções para desenvolvedor



6/26/2018 TCC Leonardo Leal Oliveira

21. **Observação**

Questionário de Usabilidade

Esta seção visa avaliar a usabilidade da ferramenta testada por você nas seções anteriores.

22. **Das atividades solicitadas, quantas atividades você conseguiu executar sem auxílio?** *
Mark only one oval.
- Todas
- A maior parte
- Metade das tarefas
- Menos da metade das tarefas
- Nenhuma tarefa

23. **Observação**

24. **De modo geral, você achou a ferramenta intuitiva e fácil de usar?** *
Mark only one oval.
- Sim
- Não

25. **Observação**

https://docs.google.com/forms/d/1Y7GtEEeDWwUMjwBn4TlNSFsg5iMNB83Fmxjrjhbm88/edit?ts=5b218e58 7/12

6/26/2018 TCC Leonardo Leal Oliveira

26. **Se você nunca havia utilizado uma ferramenta para auxiliar na indexação de vídeos com base nas faces detectadas, após o uso dessa ferramenta, você voltaria a utilizar?** *
Mark only one oval.
- Sim
- Não

27. **Observação**

28. **Você acha que uma ferramenta para indexar vídeos com base nas faces das pessoas contidas, auxilia na busca e organização de vídeos?** *
Mark only one oval.
- Sim
- Não

29. **Observação**

30. **Você é desenvolvedor? Se sim, você será solicitado para testar uma biblioteca disponibilizada e para isso será necessário ter o Visual Studio 2017.** *
Mark only one oval.
- Sim
- Não *After the last question in this section, stop filling out this form.*

31. **Observação**

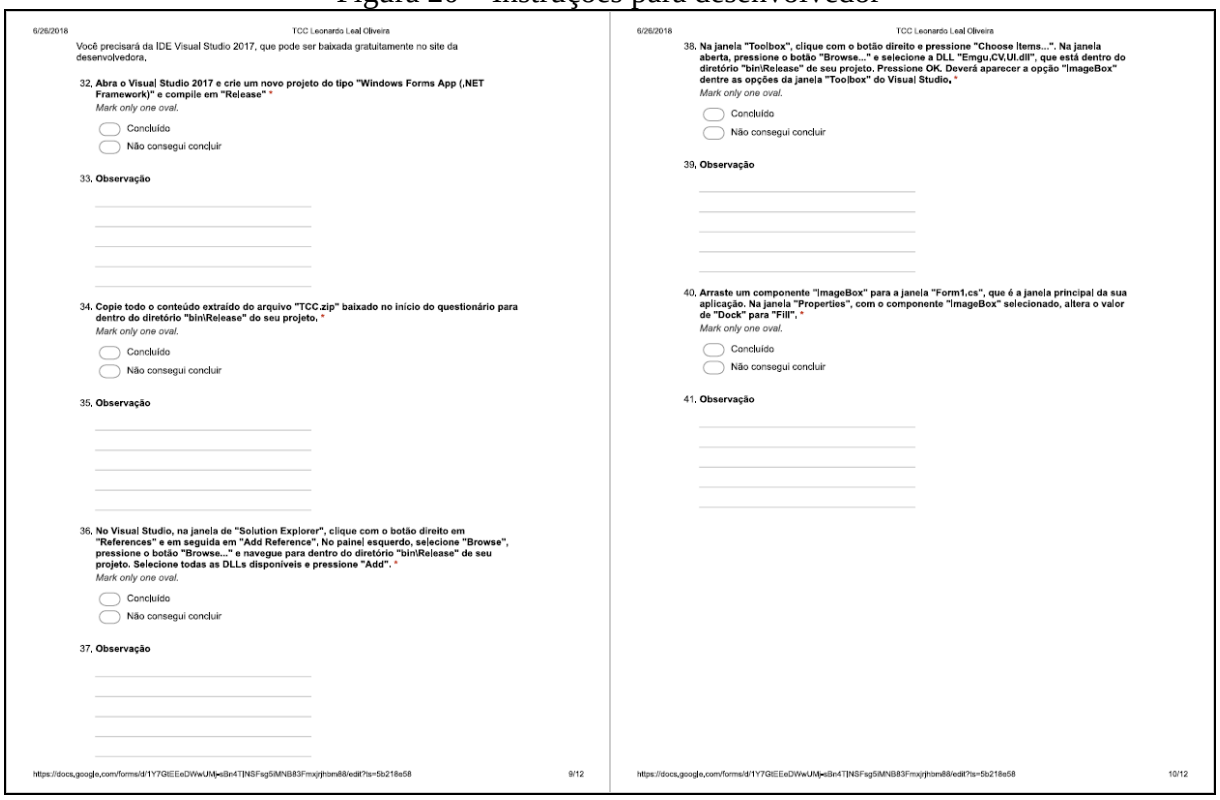
Instruções para Desenvolvedor

Está seção é destinada para usuários desenvolvedores. Ela tem o intuito de avaliar a usabilidade e reaproveitamento da biblioteca em outras aplicações.

https://docs.google.com/forms/d/1Y7GtEEeDWwUMjwBn4TlNSFsg5iMNB83Fmxjrjhbm88/edit?ts=5b218e58 8/12

Fonte: elaborado pelo autor.

Figura 20 – Instruções para desenvolvedor



6/26/2018 TCC Leonardo Leal Oliveira

Você precisará da IDE Visual Studio 2017, que pode ser baixada gratuitamente no site da desenvolvedora.

32. **Abra o Visual Studio 2017 e crie um novo projeto do tipo "Windows Forms App (.NET Framework)" e compile em "Release"** *
Mark only one oval.
- Concluído
- Não consegui concluir

33. **Observação**

34. **Copie todo o conteúdo extraído do arquivo "TCC.zip" baixado no início do questionário para dentro do diretório "bin\Release" do seu projeto.** *
Mark only one oval.
- Concluído
- Não consegui concluir

35. **Observação**

36. **No Visual Studio, na janela de "Solution Explorer", clique com o botão direito em "References" e em seguida em "Add Reference". No painel esquerdo, selecione "Browse", pressione o botão "Browse..." e navegue para dentro do diretório "bin\Release" de seu projeto. Selecione todas as DLLs disponíveis e pressione "Add".** *
Mark only one oval.
- Concluído
- Não consegui concluir

37. **Observação**

https://docs.google.com/forms/d/1Y7GtEEeDWwUMjwBn4TlNSFsg5iMNB83Fmxjrjhbm88/edit?ts=5b218e58 9/12

6/26/2018 TCC Leonardo Leal Oliveira

38. **Na janela "Toolbox", clique com o botão direito e pressione "Choose Items...". Na janela aberta, pressione o botão "Browse..." e selecione a DLL "Emgu.CV.UI.dll", que está dentro do diretório "bin\Release" de seu projeto. Pressione OK. Deverá aparecer a opção "ImageBox" dentre as opções da janela "Toolbox" do Visual Studio,** *
Mark only one oval.
- Concluído
- Não consegui concluir

39. **Observação**

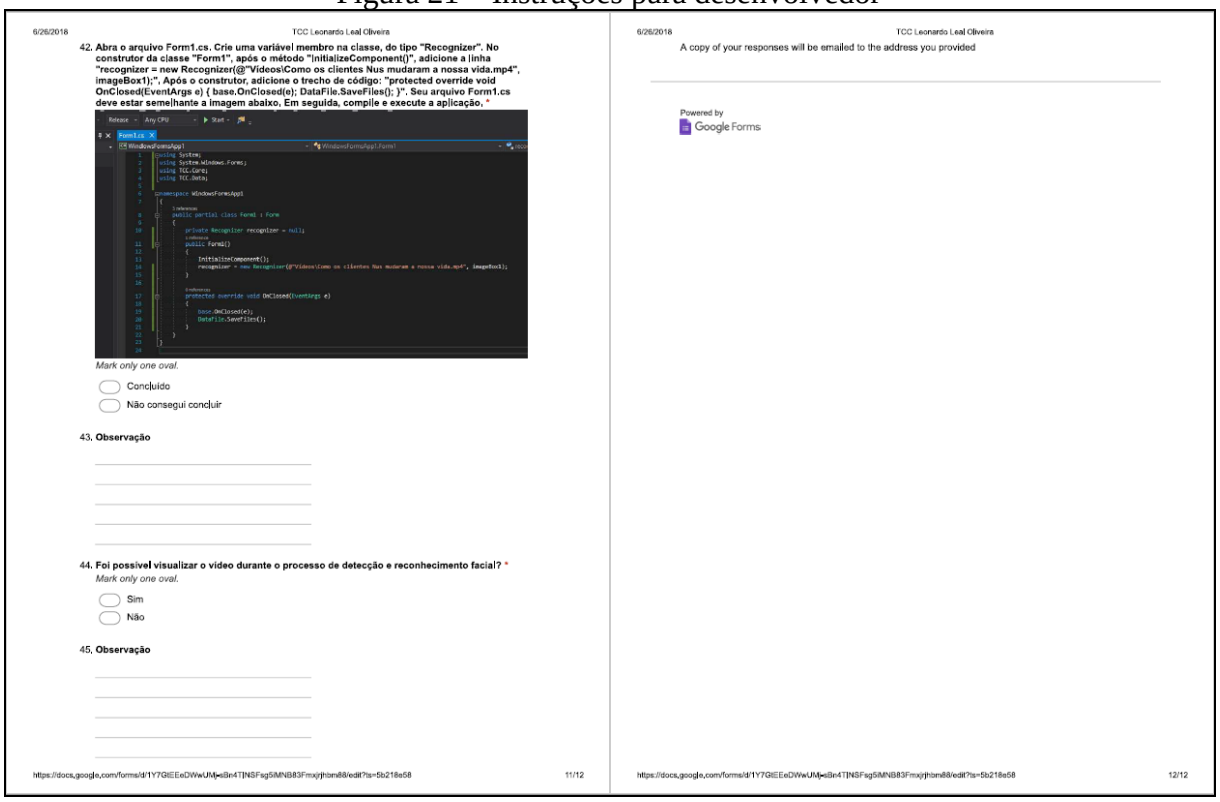
40. **Arraste um componente "ImageBox" para a janela "Form1.cs", que é a janela principal da sua aplicação. Na janela "Properties", com o componente "ImageBox" selecionado, altera o valor de "Dock" para "Fill",** *
Mark only one oval.
- Concluído
- Não consegui concluir

41. **Observação**

https://docs.google.com/forms/d/1Y7GtEEeDWwUMjwBn4TlNSFsg5iMNB83Fmxjrjhbm88/edit?ts=5b218e58 10/12

Fonte: elaborado pelo autor.

Figura 21 – Instruções para desenvolvedor



6/26/2018 TCC Leonardo Leal Oliveira

42. **Abra o arquivo Form1.cs. Crie uma variável membro na classe, do tipo "Recognizer". No construtor da classe "Form1", após o método "initializeComponent()", adicione a linha "recognizer = new Recognizer(@"Videos\Como os clientes Nus mudaram a nossa vida.mp4", imageBox1);", Após o construtor, adicione o trecho de código: "protected override void OnClosed(EventArgs e) { base.OnClosed(e); DataFile.SaveFiles(); }". Seu arquivo Form1.cs deve estar semelhante a imagem abaixo. Em seguida, compile e execute a aplicação.** *

Mark only one oval.

- ◯ Concluído
- ◯ Não consegui concluir

43. **Observação**

44. **Foi possível visualizar o vídeo durante o processo de detecção e reconhecimento facial?** *

Mark only one oval.

- ◯ Sim
- ◯ Não

45. **Observação**

https://docs.google.com/forms/d/1Y7GtEEeDWwUMjwBn4TlNSFsg5iMNB83Fmxjrjhbm88/edit?ts=5b218e58 11/12

6/26/2018 TCC Leonardo Leal Oliveira

A copy of your responses will be emailed to the address you provided

Powered by Google Forms

https://docs.google.com/forms/d/1Y7GtEEeDWwUMjwBn4TlNSFsg5iMNB83Fmxjrjhbm88/edit?ts=5b218e58 12/12

Fonte: elaborado pelo autor.