DESENVOLVIMENTO DE UM SISTEMA DE VISÃO COMPUTACIONAL PARA RECONHECIMENTO DE MÁSCARA FACIAL

Giovane de Oliveira Pianco[1], Débora Pelicano Diniz[1]

[1]Faculdade de Tecnologia de FATEC Ribeirão Preto (FATEC)

Ribeirão Preto, SP – Brasil

giovane.pianco@fatec.sp.gov.br,debora.diniz2@fatec.sp.gov.br

Resumo. *Devido a pandemia mundial provocada pelo COVID-19, um vírus altamente contagioso, o uso de máscara passa a ser de extrema importância e obrigatória na maioria dos lugares, e por esse motivo, o trabalho aqui apresentado teve como objetivo o desenvolvimento de um sistema para identificar se uma pessoa está usando máscara facial, por meio de uma câmera e utilizando a abordagem de Análise de Componentes Principais (PCA) e o algoritmo KNN (K-Nearest Neighbors). O software será útil não somente durante a pandemia, mas também no futuro, em lugares onde pode haver risco de contaminação, como por exemplo em hospitais.*

Abstract. *Due to a worldwide pandemic caused by COVID-19, a highly contagious virus, the use of a mask becomes extremely important and mandatory in most places, and for this reason, the project presented here aimed to develop a system to identify whether a person is wearing a face mask, using a camera and using a Principal Component Analysis (PCA) and KNN algorithm (K-Nearest Neighbors) approach. The software will be useful not only during a pandemic, but also in the future, in places where there may be a risk of contamination, such as in hospitals.*

1. Introdução

Tendo em vista que o mundo hoje está totalmente conectado e caminha para a transformação digital (SILVA, 2018), deve-se considerar a importância de automatizar e digitalizar todos os processos possíveis.

A visão computacional se tornou uma grande ferramenta para a identificação de objetos em imagens estáticas ou vídeos. As técnicas de visão computacional aliadas a grandes recursos computacionais conseguem resolver problemas complexos que antigamente só seriam possíveis de serem resolvidos por pessoas, por exemplo: a detecção de incêndios florestais (ALVES, 2018) ou diagnósticos de doenças em vinhas (AFONSO, 2019).

Vivendo em um momento de tensão pandêmica mundial, consegue-se enxergar a necessidade de um software que consiga fazer a leitura e, com precisão, indicar se um indivíduo está ou não usando uma máscara facial. Assim, o objetivo deste trabalho é desenvolver um sistema capaz de reconhecer uma pessoa que esteja usando máscara facial.

2. Visão Computacional

O ser humano utiliza os olhos para ter contato com o mundo a sua volta. Por meio da visão pode contar, observar cores, perceber expressões faciais, notar padrões, entre outras coisas.

Inclusive o processo de tomada de decisões pode ser facilitado ao se considerar os dados obtidos pela visão.

Assim como a visão é capaz de identificar objetos e facilitar processos no cotidiano do ser humano, pode também auxiliar a humanidade por meio de computadores com a chamada visão computacional. Segundo Backes e Sá Junior (2016, p.1) "Visão computacional é a área de estudo que tenta reproduzir para computadores a capacidade de visão que nós humanos temos, a chamada visão natural."

Ainda segundo os autores, um sistema de Visão computacional é divido em algumas fases, sendo elas (BACKES & SÁ JUNIOR, 2016):

- **Aquisição**: Fase responsável pela captação das imagens por meio de um sensor ótico.
- **Processamento de Imagem**: Responsável por fazer alguns ajustes na imagem captada na fase anterior como remover ruídos, salientar borda, suavizar a imagem, entre outros.
- **Segmentação**: Responsável por particionar a imagem em regiões de interesse, como por exemplo, em uma imagem de pessoa para analisar se está usando máscara facial ou não, a região de interesse seria somente o rosto.
- **Extração de características / Análise de imagens**: Obtém um conjunto de características do objeto de interesse.
- **Reconhecimento de Padrões**: Esta fase é responsável por classificar ou agrupar as imagens com base em suas características.

2.1. Aplicações da visão computacional

De Acordo com Barelli (2018, p.8) existem diversos problemas que podem ser resolvidos com alguns algoritmos de visão computacional, principalmente quando se alia a visão computacional às técnicas de aprendizado de máquina, que permitem que o computador aprenda e aperfeiçoe o seu próprio desempenho em alguma determinada tarefa.

Pode-se implementar essa tecnologia para auxiliar diversas áreas, desde a área médica até a indústria, como se percebe pelos exemplos apresentados a seguir.

Na indústria automotiva, veículos são projetados para trafegar autonomamente usando sistemas que são capazes de reconhecer outros veículos, placas e pedestres que podem estar transitando na via, por meio da visão computacional (BARELLI 2018).

Na medicina, técnicas de visão computacional estão sendo empregadas para detectar anomalias em exames por imagem como tomografias, ressonâncias magnéticas e ultrassons, diagnosticando precocemente doenças e garantindo a qualidade de vida de diversos pacientes (BARELLI 2018).

Pode-se observar que a visão computacional vem se desenvolvendo em diversos campos e à medida que esses sistemas se tornam mais precisos e seguros, podem se expandir para setores ainda mais complexos, provendo uma boa expectativa para o futuro da área de estudo.

2.2. As vantagens do uso de máscara facial

A partir do reconhecimento dos primeiros casos de síndrome respiratória aguda grave 2 (SARS-CoV-2), foram iniciados diversos debates sobre medidas de controle da propagação

do vírus via gotículas e ou aerossol (CAMARGO, 2020).

Um dos métodos eficientes para a contenção da propagação da pandemia foi orientar a população a utilizar máscaras faciais, visando a diminuição de casos de vítimas da COVID-19, pois o uso de máscaras faciais permite as seguintes vantagens (OMS, 2020):

- Reduzir a propagação de gotículas respiratórias contendo partículas virais infecciosas, incluindo de pessoas infectadas antes de desenvolverem sintomas;
- Aumentar o nível de aceitação do uso de máscara, seja para prevenir a infecção de outras pessoas ou por pessoas que cuidam de pacientes com COVID-19 em ambientes não clínicos;
- Fazer com que as pessoas sintam que podem desempenhar um papel contribuindo para parar a propagação do vírus;
- Prevenir a transmissão de outras doenças respiratórias, como tuberculose e gripe, reduzindo a carga dessas doenças durante a pandemia.

3. Ferramentas utilizadas

Nessa seção serão comentadas as ferramentas utilizadas para o desenvolvimento do sistema para reconhecimento do uso de máscara facial.

3.1. Python

Python é uma linguagem de programação multi-paradigma que foi criada por Guido van Rossum, no início da década de 1990, com o intuito de suceder uma linguagem chamada ABC (PYTHON.ORG, 2021).

Python é uma das linguagens que mais crescem atualmente devido a sua compatibilidade com muitas plataformas e a capacidade de auxiliar outras linguagens. É uma linguagem simples que conquistou a comunidade cientifica e a comunidade de análise de dados (CAELUM).

Segundo Borges (2014, p.14) a linguagem Python possui uma sintaxe simples e clara, o que facilita a legibilidade do código tornando o desenvolvimento mais produtivo.

3.2. OpenCV

OpenCV é uma biblioteca multiplataforma de código aberto, projetada para desenvolvimento de aplicativos na área de visão computacional e processamento de imagens (OPENCV, 2021).

A biblioteca OpenCV tem mais de 2.500 algoritmos otimizados, que incluem um conjunto abrangente de algoritmos de visão computacional e aprendizado de máquina clássicos e de última geração. Esses algoritmos podem ser usados para detectar e reconhecer rostos, identificar objetos, ações humanas, juntar diversas imagens para produzir uma cena, entre outros (OPENCV).

3.3. Numpy

O NumPy é uma poderosa biblioteca Python que é usada principalmente para realizar cálculos em Arrays Multidimensionais. O NumPy fornece um grande conjunto de funções e operações que ajudam os programadores a executar facilmente cálculos numéricos

(NUMPY.ORG, 2021).

A estrutura de dados mais importante que o NumPy fornece é um objeto poderoso, um tipo de Array, chamada ndarray. O objeto ndarray consiste em um segmento unidimensional contíguo da memória do computador, combinado com um esquema de indexação que mapeia cada item para um local no bloco de memória. Pode-se executar grandes cálculos em uma única linha de código com a excelente interface que o NumPy expõe. Entre vários benefícios o código vetorizado é mais resumido e fácil de ler, sendo mais rápido devido existir uma menor possibilidade de *bugs* pelo fato de usar menos linhas de código (NUMPY.ORG, 2021).

3.4. Scikit-Learn

Scikit-learn é um módulo Python que implementa uma grande quantidade de algoritmos de aprendizado de máquina. Este módulo, ou biblioteca, tem como ênfase, oferecer aos usuários facilidade e desempenho. É uma biblioteca gratuita e de simples uso (PEDREGOSA et al. 2011).

4. Desenvolvimento do sistema

Essa seção descreve as técnicas utilizadas para o desenvolvimento do sistema de reconhecimento de máscara facial, que é o objetivo deste trabalho. Este capítulo demonstra uma abordagem prática para o reconhecimento de faces e treinamento de modelos para identificação de pessoas que estejam utilizando máscara facial.

4.1. Reconhecimento de faces

O primeiro passo para reconhecer de fato se uma pessoa está ou não usando máscara facial, é identificar e classificar a face que está sendo apresentada na imagem. Existem diversos métodos para implementar o reconhecimento de faces, porém, neste trabalho foi utilizada a abordagem de Análise de Componentes Principais (*PCA - Principal Component Analysis*).

A PCA é um método que tem por finalidade básica analisar os dados utilizados visando a redução, eliminação de sobreposições e escolha das formas mais representativas dos dados a partir de combinações lineares de variáveis (SILVA, 2009), ou seja, a *PCA* busca minimizar as variáveis e encontrar os objetos que são mais notáveis (Componentes Principais) dentro de uma base de dados para fazer a classificação de um objeto.

Para extrair as características mais notáveis de uma imagem, foi utilizada a classe *sklearn.decomposition.PCA*, passando como parâmetro para o construtor da classe o total de características que deverá ser extraído da imagem que classificará a face reconhecida.

A Figura 1 exemplifica a instanciação de um modelo *PCA* e seu treinamento utilizando a biblioteca do Scikit-Learn.

O número 30 que pode ser observado na linha 58 da Figura 1, representa a quantidade de características mais notáveis que o algoritmo irá utilizar para fazer a classificação da imagem. Após testes no treinamento do modelo, chegou-se a um resultado de 85% de assertividade treinando o modelo com 30 características.

```
54  def pca_model(X_train):
55      '''
56      PCA para extração de features das images
57      '''
58      pca = PCA(n_components=30)
59      pca.fit(X_train)
60
61      return pca
62
```

Figura **1 – Modelo PCA**
Fonte: Autoria própria

4.2. Treinamento dos modelos

A partir da identificação das características mais notáveis por meio do *PCA* (veja a seção anterior) o modelo foi treinado com base na amostra de dados (banco de imagens) que previamente foi adquirido (KAGGLE, 2020). Após este procedimento, foi transformado os conjuntos de teste e de treino para um tipo de dado que fosse reconhecido como parâmetro para treinamento de um outro modelo que irá detectar se a face identificada está ou não com máscara facial. Para o treinamento deste novo modelo foi utilizado o algoritmo KNN (*K-Nearest Neighbors*).

O *K-Nearest Neighbors* (KNN) é um algoritmo que toma por base medidas de distancias entre pares de padrões para classificar um objeto. Geralmente, no caso de atributos numéricos, a distância Euclidiana entre dois padrões é usada (JIANG et al., 2007). Para o treinamento do KNN foi necessário passar por parâmetro o conjunto das características (mais notáveis da face) de teste e o conjunto de classes de treino.

A Figura 2 exemplifica a instanciação de um modelo KNN utilizando o módulo *GridSearchCV* da biblioteca Scikit-Learn. O modelo KNN está sendo criado também com os parâmetros de *grid* definidos no objeto "*grid_params*"(linha 70), além disso, na linha 76 é passado um terceiro parâmetro ("*refit*") que indica o retreinamento do modelo.

```
64  def knn(X_train, y_train):
65
66      warnings.filterwarnings("ignore")
67      '''
68      Modelo K-Nearest Neighbors
69      '''
70      grid_params = {
71      "n_neighbors": [1, 3, 5, 11, 19, 23, 29],
72      "weights": ["uniform", "distance"],
73      "metric": ["euclidean", "manhattam", "cosine", "l1", "l2"]
74      }
75
76      knn_model = GridSearchCV(KNeighborsClassifier(), grid_params, refit=True)
77
78      knn_model.fit(X_train, y_train)
79
80      return knn_model
81
```

Figura 2 – Modelo KNN
Fonte: Autoria própria

O algoritmo de treinamento do KNN irá escolher entres os parâmetros passados em ***grid_params*** (linha 70) qual o conjunto de informações irá obter um melhor nível de acurácia. O parâmetro ***n_neighbors*** (linha 71) representa quantos 'vizinhos' o algoritmo poderá usar para classificar um certo dado como parte daquele conjunto, o parâmetro ***weights*** (linha 72) retrata o peso que cada vizinho poderá ter na classificação, e o último parâmetro, ***metric*** (linha 73), representa o tipo de distância que poderá ser utilizada.

4.3. Captura e processamento da imagem da Câmera

Para capturar a imagem de uma câmera foi utilizado um laço de repetição (linha 32) no qual o módulo *cv2.VideoCapture* foi utilizado para receber as informações da webcam instalada no computador, como pode-se notar na Figura 3.

```
31  #Abrindo a webcam...
32  while True:
33      status, frame = cam.read() #Lendo a imagem e extraindo frame
34
35      if not status:
36          break
37
38      if cv2.waitKey(1) & 0xff == ord('q'):
39          break
```

Figura 3 – Captura de Figura
Fonte: Autoria própria

O processamento da imagem é exemplificado na Figura 4 e os procedimentos são:

- Linha 42 - Transformar a imagem em escala de cinza com o método *cv2.cvtColor,* passando como parâmetro o *frame* que foi capturado;
- Linha 45 - Identificar as faces no *frame* com o método *cv2.CascadeClassifier. DetectMultiScale;*
- Para cada face identificada, o *software* realiza o seguinte processamento:
    - Linha 49 - Recorta a imagem capturada;
    - Linha 51 - Verifica se a imagem obtida é maior ou igual a 150x150 *pixels;*
    - Linha 52 - Redimensiona a imagem obtida para 160X160 *pixels* utilizando o método *cv2.resize;*
    - Linha 53 - Extrai os componentes principais da imagem com o método *pca.tranform;*
    - Linha 55 - Classifica a imagem com o método *knn.predict.*

Em seguida o software faz o desenho dos retângulos, escreve na tela se a pessoa está ou não utilizando a máscara facial como pode-se observar na Imagem 5 e Imagem 6.

```
41      #Transformando a imagem em escala de cinza
42      gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
43
44      #Detectando faces na imagem
45      faces = classifier.detectMultiScale(gray)
46
47      #Iterando nas faces encontradas
48      for x,y,w,h in faces:
49          gray_face = gray[y:y+h, x:x+w] #Recortand região da face
50
51          if gray_face.shape[0] >= 150 and gray_face.shape[1] >= 150:
52              gray_face = cv2.resize(gray_face, (160,160)) #Redimensionando
53              vector = pca.transform([gray_face.flatten()]) #Extraindo features da imagem
54
55              pred = knn.predict(vector)[0] #Classificando a imagem
```

Figura 4 – Processamento Imagem 01
Fonte: Autoria própria

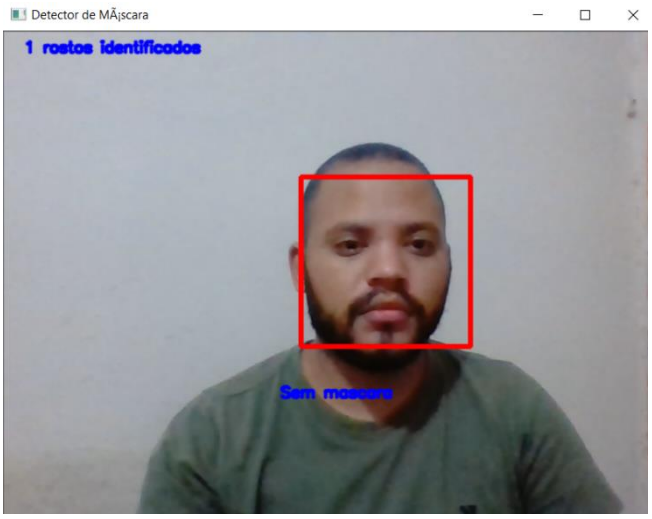

Figura 5 – Identificação de pessoa sem máscara facial.
Fonte: Autoria própria

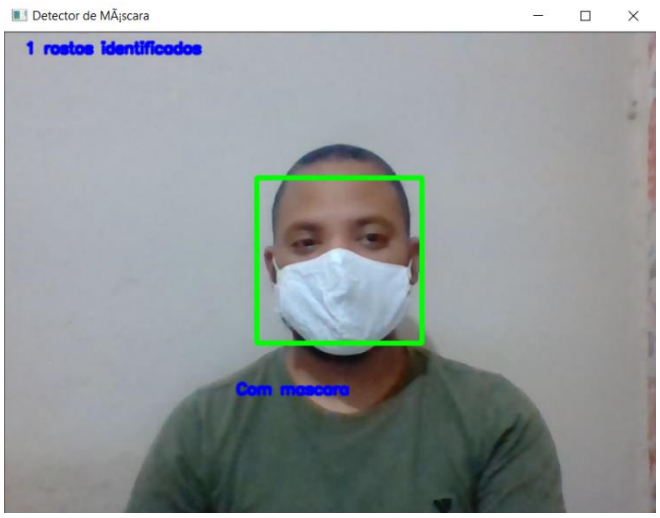

Figura 6 – Identificação de pessoal com máscara facial
Fonte: Autoria própria

5. Conclusão

Considerando a situação provocada pela pandemia que está instalada mundialmente, em que as pessoas estão sendo obrigadas a usar máscaras faciais para adentrar na maioria dos ambientes, visto ser esse um dos meios para se evitar maior contaminação da população, e considerando também a transformação digital e a maior facilidade para automatizar praticamente todos os processos, o objetivo do trabalho aqui apresentado foi o desenvolvimento de um sistema que identifica se uma pessoa está usando máscara facial.

O objetivo proposto foi alcançado, visto que o sistema está funcionando. Foram utilizados algoritmos de reconhecimento facial e seu treinamento foi realizado com imagens previamente adquiridas do repositório *Kaggle* (KAGGLE, 2020). Foi alcançado neste trabalho um resultado satisfatório visto que foi atingida uma acurácia de aproximadamente 85% no treinamento dos modelos e a partir de testes realizados em 5 pessoas, o algoritmo foi capaz de identificar o uso de máscara facial em todos os participantes.

O sistema consegue capturar imagens em tempo real por meio de uma câmera e identificar se um indivíduo está ou não fazendo uso da máscara facial, apesar de apresentar dificuldades na detecção de faces em ambientes com pouca luminosidade e na identificação quando um indivíduo está fazendo o uso de acessórios como óculos ou chapéu. Esse segundo problema ocorreu por não conseguir fazer um treinamento adequado dos modelos, pois não havia um conjunto satisfatório de imagens com pessoas utilizando estes acessórios no banco de imagens usados para os treinamentos do algoritmo do sistema.

O sistema desenvolvido pode ser utilizado em trabalhos futuros para a identificação de pessoas em lugares que seja obrigatório o uso da máscara facial, como por exemplo: lojas, comércio, aplicativos de transporte, shoppings, entre outros.

Referências

AFONSO, S. Utilização de algoritmos de visão computacional para diagnóstico de doenças da vinha. 2019. Disponível em: < https://repositorio-aberto.up.pt/bitstream/10216/125792/2/380526.pdf>. Acesso em: 15 mai. 2021.

ALVES, J. M. R. Detecção de incêndios florestais com recurso a Deep Learning e Visão Computacional. 2018. Disponível em: < https://bdigital.ufp.pt/handle/10284/7058>. Acesso em: 15 mai. 2021.

BACKES, A. R.; SÁ JUNIOR, J. J. M. Introdução à Visão Computacional usando MATLAB. Rio de Janeiro: Alta Books Editora, 2016.

BARELLI, F. Introdução à Visão Computacional: Uma abordagem prática com Python e OpenCV. São Paulo: Casa do Código, 2018.

CAELUM. Python e Orientação a Objetos. Disponível em <https://www.caelum.com.br/apostila-python-orientacao-a-objetos>. Acesso em: 11 abr. 2021.

BORGES, L. E. Python para desenvolvedores. São Paulo: NOVATEC, 2014.

CAMARGO, M. C. et al. Eficácia da máscara facial (TNT) na população para a prevenção de infecções por coronavírus: revisão sistemática. Ciência & Saúde

Coletiva [online]. v.25, n.9, p.3365-3376, 2020. Disponível em: <https://doi.org/10.1590/1413-81232020259.13622020>. Acesso em: 28 mar. 2021.

JIANG, L. et al. Survey of improving k-nearest-neighbor for classification. In: FSKD, 2007. p. 679–683.

KAGGLE. Detecção de máscara facial, 2020. Disponível em:<https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset>. Acesso em: 15 mai. 2021.

NUMPY.ORG. O que é NumPy?. 2021. Disponível em: <https://numpy.org/doc/stable/user/whatisnumpy.html>. Acesso em 11 abr. 2021.

OMS, ORGANIZAÇÃO MUNDIAL DA SAÚDE (WHO). Mask use in the context of COVID-19, 2020. Disponível em: <https://www.who.int/publications/i/item/advice-on-the-use-of-masks-in-the-community-during-home-care-and-in-healthcare-settings-in-the-context-of-the-novel-coronavirus-(2019-ncov)-outbreak>. Acesso em: 28 mar. 2021.

OPENCV.ORG. Introduction. 2021. Disponível em: <https://docs.opencv.org/4.5.2/d1/dfb/intro.html>. Aceso em 11 abr. 2021.

PEDREGOSA et al., JMLR 12, pp. 2825-2830, 2011.

PYTHON.ORG. Python. 2021. Disponível em: < https://docs.python.org/3/license.html>. Acesso em 11 abr. 2021.

SILVA, G. N. Estudo da técnica PCA (Análise de Componentes Principais) e Autofaces aplicadas ao reconhecimento de faces humanas. Marilia, 2009. TCC – Curso de Bacharelado em Ciência da Computação, FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA" CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA – UNIVEM.

SILVA, N. Transformação Digital, a 4ª revolução industrial. FGV Energia. p. 4, ago. 2018. Disponível em: <https://fgvenergia.fgv.br/sites/fgvenergia.fgv.br/files/coluna_opiniao_-_transformacao_digital.pdf >