Modelo para reconhecimento de faces num Raspberry usando KNN

Rafaela Melo[1], **Luis Cuevas Rodriguez**[1], **Jucimar Maia Junior**[1]

[1]Laboratório LUDUS – Universidade do Estado do Amazonas (UEA)
Manaus – AM – Brazil

{rmf.lic16,lrodriguez,jjunior}@uea.edu.br

Abstract. *This paper presents a study about nearest neighbor algorithm applied to facial recognition in a Raspberry PI. The experiment was developed using the Python programming language and a library specifically for face classification applications. Variables such as number of neighbors and limit distance were investigated; besides aspects of image distortion and magnification, in order to evaluate the algorithm performance.*

Resumo. *Este artigo apresenta um estudo sobre o algoritmo dos vizinhos mais próximos aplicado ao reconhecimento facial em um Raspberry PI. O experimento foi desenvolvido usando a linguagem de programação Python e uma biblioteca específica para aplicações com classificação de faces. Foram investigadas variáveis como número de vizinhos e distância limite; além de aspectos de distorção e ampliação de imagem, com o intuito de avaliar o desempenho do algoritmo.*

1. Introdução

O reconhecimento facial (RF) vem tomando grande atenção nos últimos anos por ser uma das aplicações de análise de imagens mais bem sucedidas, outro fator que influencia nesse cenário são as diversas aplicações comerciais e policiais, além da disponibilidade de tecnologias viáveis após tantos anos de pesquisa [Zhao et al. 2003]. Apesar dos sistemas de reconhecimento terem atingido certa maturidade, ainda existem problemas nessa área relacionados a aplicações reais. Rostos podem apresentar muitas variações, formato da boca, distância entre os olhos, altura da testa. Um ser humano é capaz de reconhecer uma pessoa apesar dessas variações e do contexto que a face se encontra, diferente de uma máquina que precisa de diversos processos para detectar e reconhecer um rosto [Silva and Cintra 2015].

Um sistema de RF é definido basicamente por três fases distintas, detecção, extração de recursos e reconhecimento (Figura 1), e possui dois modos: identificação e verificação. Um sistema de identificação tem por objetivo classificar uma face a partir de um banco de dados, informando quem é a pessoa; já um sistema de verificação precisa informar se a pessoa é conhecida ou desconhecida [Zhao et al. 2003]. O grande problema de tais aplicações é a capacidade de classificar uma pessoa com alta precisão de certeza, isto é importante principalmente por problemas de segurança e também pode ser usado para ter acesso a registros médicos e criminais; resolver esse problema significa fornecer melhores serviços e áreas seguras para as pessoas [Kambi Beli and Guo 2017].

O reconhecimento facial ainda é considerado um problema complexo da visão computacional porque as faces apresentam formas semelhantes e as mesmas características (nariz, boca, olhos), podem se apresentar em diferentes posições e expressões; além disso, as imagens que contém as faces apresentam variações de iluminação, dimensão e até mesmo qualidade, que podem influenciar na classificação dos rostos [Diniz et al. 2016, Kambi Beli and Guo 2017].

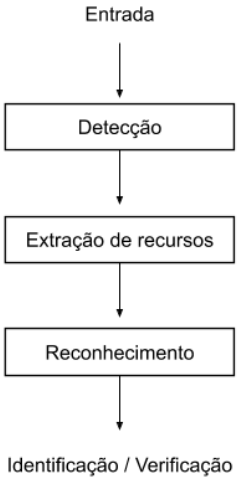

Figura 1. Sistema de Reconhecimento Facial

Existem diversos algoritmos para tratar o problema do reconhecimento de faces. Buscando uma alternativa para um sistema de baixo custo computacional, este trabalho realizou um estudo sobre o algoritmo dos vizinhos mais próximos – KNN (*K-Nearest Neighbor*) aplicado ao reconhecimento facial em um Raspberry PI. O artigo está organizado como segue: a Seção 2 descreve o algoritmo dos vizinhos mais próximos e sua aplicação ao RF; a Seção 3 apresenta o processo de implementação do experimento realizado; as Seções 4 e 5 tratam respectivamente da análise de resultados e das considerações finais.

2. Algoritmo dos Vizinhos Mais Próximos (KNN)

O algoritmo dos vizinhos mais próximos é um dos mais simples dentro do aprendizado de máquina (*Machine Learning*). O método consiste em classificar objetos de acordo com os exemplos de treinamento mais próximos no espaço de recursos [Cover et al. 1967, Kim et al. 2012].

O treinamento é realizado a partir de um conjunto de vetores de recursos das imagens e rótulos para identificar cada classe [Kim et al. 2012]. Na classificação, o objeto de entrada é atribuído ao rótulo de seus vizinhos mais próximos, de acordo com a quantidade de votos definida por *k*. A figura 2 demonstra o comportamento do algoritmo KNN, por exemplo, quando k=3, são pesados para classificação os três vizinhos mais próximos. Então, na imagem, quando k=3 o objeto de entrada é atribuído a classe azul.

A busca pela vizinhança é feita através de uma medida de distância, que determinará quais são os vizinhos mais próximos, para verificar a distância as funções utilizadas

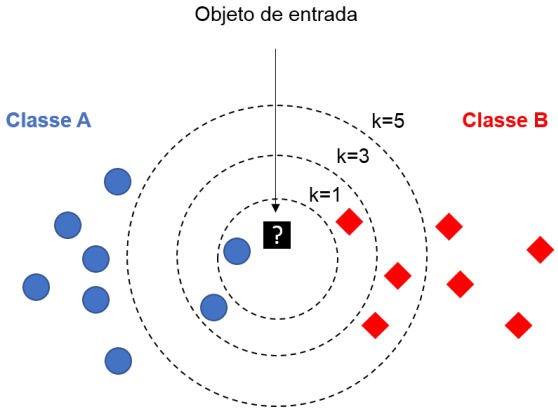

Figura 2. Algoritmo KNN

podem ser: distância Euclidiana, distância de Manhattan e distância Euclidiana normalizada [Jiangsheng 2002].

Uma das vantagens do algoritmo KNN é que ele funciona bem para classes multimodal (classes que possuem características diferentes para diferentes subconjuntos), ou seja, embora uma classe seja composta por vários subconjuntos, isso não vai influenciar na precisão do algoritmo, visto que a classificação é feita baseada nos vizinhos mais próximos [Kim et al. 2012]. A desvantagem do algoritmo KNN é a complexidade computacional porque é preciso calcular a distância entre todos os elementos do conjunto para descobrir quais são os k vizinhos mais próximos [Wang and Ju 2008].

2.1. KNN aplicado ao reconhecimento facial

Por mais que se trate de uma métrica antiga de reconhecimento de padrões, o algoritmo KNN pode ser aplicado em questões de visão computacional como reconhecimento de expressões e reconhecimento facial [Xu et al. 2013].

O algoritmo KNN tem sido comumente utilizado para classificar pessoas por possuir um tempo de execução menor e uma boa precisão se comparado a outros classificadores [Kambi Beli and Guo 2017]. Diversos estudos aplicaram o KNN para reconhecimento facial ao longo dos últimos anos. Em [Khastavaneh et al. 2017] foi implementado um classificador baseado em KNN para tentar resolver o problema da classificação de pessoas desconhecidas, que é essencial em um sistema de reconhecimento facial. Conforme o algoritmo KNN é definido, um objeto de entrada é classificado como a classe dominante nos vizinhos mais próximos, porém isso causa muitos falsos positivos. Os autores então, aplicaram uma regra em que a média das distâncias da classe dominante é calculada e a partir desse resultado, o objeto de entrada é classificado como conhecido ou desconhecido.

A combinação entre o KNN e o LBP (*Local Binary Patterns*) foi utilizada por [Kambi Beli and Guo 2017] em um sistema de identificação para resolver problemas de iluminação, pose e expressão. O LBP, ou padrão binário local é um método para extrair recursos de faces, caracterizado como um conjunto ordenado de comparações binárias de intensidade entre os pixels, mais especificamente entre o pixel central e os pixels adjacentes [Ojala et al. 1996]. O KNN foi escolhido pelos autores por ser um dos algoritmos mais simples de aprendizado de máquina e de classificação. No estudo realizado utilizou-

se uma abordagem de validação cruzada para estimar o melhor valor de K e para medida de similidade foi usada a distância euclidiana.

Conhecido como um dos primeiros métodos para reconhecimento facial, Eigenfaces [Turk and Pentland 1991] analisa características relevantes para diferenciar duas faces, sem levar em conta as formas geométricas do rosto (nariz, boca, olhos). Para melhorar as taxas de acurácia em um sistema de reconhecimento facial que une técnicas de Eigenfaces e o algoritmo KNN, [Diniz et al. 2016] investigaram imagens com 3 dimensões, número de características de faces, valores de k e diferentes medidas de distância (euclidiana, euclidiana normalizada e Manhattan), avaliando tais aspectos, os autores conseguiram identificar os principais parâmetros para melhorar o desempenho do sistema.

Na literatura existem muitos estudos de reconhecimento facial para computadores de alta velocidade, porém, sistemas de identificação costumam ser implementados em dispositivos pequenos e compactos, por isso, em [Setiawan and Muttaqin 2015] o algoritmo KNN é investigado para um processador de baixo custo. A proposta foi implementada em um Raspberry equipado com um processador ARM11 700MHz. Os autores obtiveram 91,5% como melhor taxa de acurácia, usando k=1.

2.2. Classificador KNN

Diferente de outros algoritmos de aprendizado de máquina, o treinamento de um classificador KNN apenas armazena codificações das imagens e seus respectivos rótulos. A partir de um conjunto de imagens organizado em pastas entituladas com o nome das classes (pessoas), o algoritmo faz a codificação das imagens, ou seja, transforma uma imagem em um vetor de números que representam uma face, esses números são os pixels existentes na imagem, de acordo com o canal de cor RGB [Goswami 2018]. A cada vetor (face) é atribuído um rótulo (nome da pessoa a quem a face pertence). A figura 3 ilustra basicamente um classificador KNN:

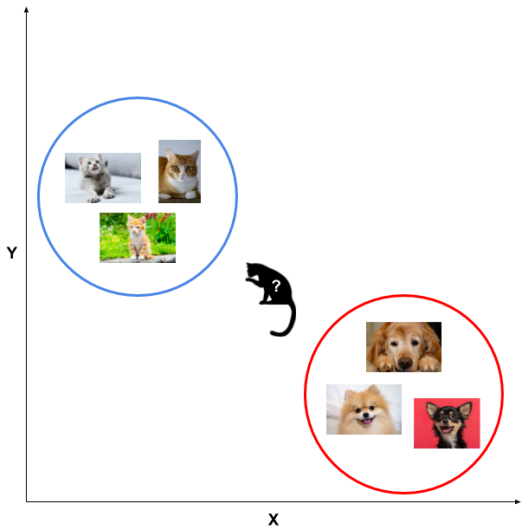

Figura 3. Representação do classificador KNN

Para identificar os vizinhos mais próximos é aplicada uma medida de distância, que pode ser a euclidiana, Manhattan ou euclidiana normalizada [Jiangsheng 2002,

Rosebrock 2016]. Então, a imagem de entrada será comparada com as imagens dos k vizinhos mais próximos, e a ela será atribuída o rótulo da classe dominante, ou seja, a classe que tiver maior número de vizinhos que correspondam a entrada.

3. Implementação

Nesta seção serão especificados os processos seguidos para a realização do experimento, assim como as ferramentas utilizadas.

3.1. Arquitetura do modelo

Para a realização do experimento utilizou-se a biblioteca *face_recognition* da linguagem Python (`https://pypi.org/project/face_recognition/`). Esta biblioteca contém funções que permitem localizar e manipular faces em imagens facilmente. Por este motivo, o modelo foi criado a partir dessas funções e de um algoritmo KNN, que atua como classificador de rostos. O modelo usado no experimento apresenta arquitetura conforme a figura 4.

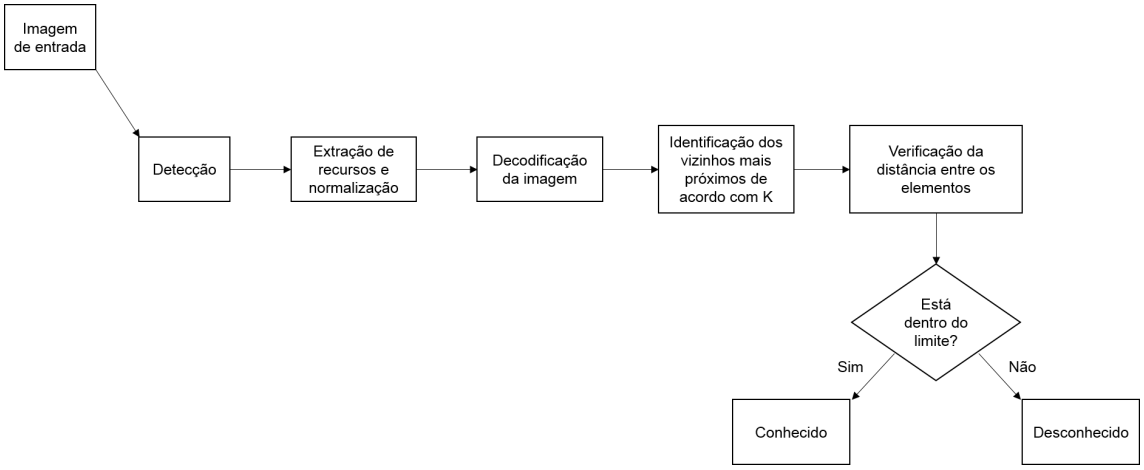

Figura 4. Arquitetura do modelo

O primeiro passo é detectar a face na imagem, o algoritmo de detecção usado pela biblioteca é o HOG [Dalal and Triggs 2005]. Após a detecção, é realizada uma extração de recursos que identifica 68 pontos que existem em todas as faces, em seguida é feita uma normalização da face, para que os olhos e a boca apareçam sempre no mesmo local. Posteriormente, uma função de codificação de imagem é aplicada e retorna um vetor de 128 dimensões que representam a face.

O próximo passo é feito pelo classificador KNN. São identificados os vizinhos mais próximos, de acordo com o k definido. Com o resultado do cálculo anterior, a face de entrada é comparada com as faces dos vizinhos mais próximos, essa comparação é feita através da distância entre os elementos existentes na faces, caso essa distância esteja dentro de uma distância limite, a face é classificada como "conhecida" (atribuindo o nome da classe correspondente), caso a distância seja maior do que o limite, então o rosto é classificado como "desconhecido".

A biblioteca *face_recognition* usa a distância euclidiana para comparar a similaridade entre duas faces. A partir dos pontos de características faciais obtidos, a distância

euclidiana é calculada para cada dois pontos dentro da face, dessa forma é possível fazer comparações entre duas imagens [Ahdid et al. 2017]. A distância euclidiana é dada por:

$$d = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}, \tag{1}$$

onde $p = (p_1, p_2, p_3, ..., p_n)$ e $q = (p_1, p_2, p_3, ..., p_n)$. Sendo p e q os pontos dentro de um espaço euclidiano n-dimensional.

3.2. Dataset

Figura 5. Exemplos de imagens do dataset

O dataset usado no experimento contém 10 classes e 714 imagens (Figura 5). Foi construído a partir de uma câmera PI – Raspberry PI (Figura 6), com sensor 5MP OV5647, 2592x1944 pixels para imagens estáticas e 1080p a 30 fps para vídeo. Foram feitos vídeos de 10 pessoas em poses distintas, rosto para: frente, direita, esquerda, cima, baixo; além disso, para pessoas que usam óculos, os vídeos foram gravados com e sem óculos. Após a gravação, foram extraídos os frames para criar um dataset de imagens.

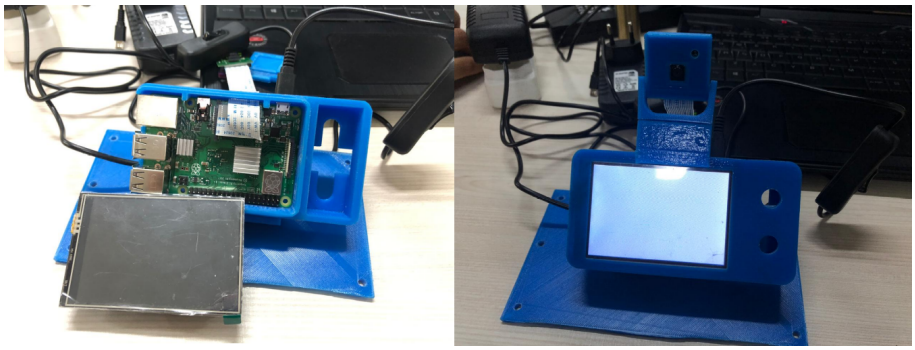

Figura 6. Raspberry PI

Além do dataset de treinamento, foi criado um conjunto de imagens para realizar testes, com um total de 189 fotos das 10 pessoas do conjunto de treinamento e também uma pasta com fotos de desconhecidos para avaliar a acurácia do modelo, contando com resultados conhecidos e desconhecidos.

3.3. Experimento

Conforme estudos realizados sobre o algoritmo KNN e a biblioteca *face_recognition*, o experimento foi executado levando em conta algumas variáveis independentes (Tabela 1) e sua influência na precisão da classificação. Como variável dependente, foi analisada somente a acurácia. O cálculo da acurácia foi feito usando o dataset de testes, de acordo com a equação abaixo:

$$acc = \frac{correctTests}{allTests}, \tag{2}$$

onde *correctTests* trata da quantidade de classificações feitas corretamente pelo algoritmo e *allTests* remete a quantidade de fotos disponível no dataset de teste.

Tabela 1. Variáveis independentes

ID	Variável	Descrição
1	K	Quantidade de vizinhos que serão pesados na classificação.
2	Distância limite	A distância limite para classificar uma pessoa como conhecida ou desconhecida.
3	num_jitters	Faz com que uma imagem seja distorcida *n* vezes para obter uma decodificação para cada versão da imagem e retornará a média. Permite melhorar a precisão do algoritmo.
4	number_of_times_to_upsample	A imagem é ampliada *n* vezes para encontrar faces menores, porém, isso torna o algoritmo mais lento.

O experimento foi realizado em duas etapas, na primeira etapa foram avaliadas duas variáveis e na segunda as quatro juntas, para verificar a acurácia do modelo nessas diferentes condições. Foram utilizadas 189 imagens de teste, com as pessoas do conjunto de treinamento e com pessoas desconhecidas.

4. Análise de resultados

Na primeira etapa dos testes (Tabela 2), onde foram avaliadas a distância limite e o número de vizinhos, é possível perceber que sendo 1 ou 3 a quantidade de vizinhos, a acurácia para distâncias iguais continua sendo a mesma. A melhor taxa de acurácia nessa fase foi com k=1 e distâncias 0.45 e 0.44. Considerando que o número de classes dentro do conjunto de treinamento é apenas 10, não foram realizados testes com um número *k* maior.

A partir de pesquisas realizadas sobre a biblioteca *face_recognition* e a melhora da precisão de classificação, foram encontradas duas variáveis dentro da biblioteca apontadas para isso. Então, realizou-se novos testes para verificar se essas variáveis influenciam no comportamento do algoritmo e na precisão.

A segunda etapa dos testes (Tabela 3) apresentou muitas variações com relação a taxa de acurácia e com os resultados da primeira fase, é possível perceber isso melhor no gráfico da figura 7.

Quanto maior for a distância limite para classificação, mais risco de identificar erroneamente uma pessoa. 0.6 é a distância indicada a ser utilizada, por isso os testes

Tabela 2. Primeira etapa dos testes

ID	K	Distância	Acurácia (%)
1	3	0.6	88,88
2	3	0.5	92,06
3	3	0.45	93,65
4	1	0.6	88,88
5	1	0.5	92,06
6	1	0.45	93,65
7	1	0.43	93,12
8	1	0.44	93,65

Tabela 3. Segunda etapa dos testes

ID	K	Distância	num_jitters	number_of_times_to_upsample	Acurácia (%)
1	3	0.6	100	2	88,88
2	3	0.5	100	2	90,47
3	3	0.45	100	2	94,17
4	1	0.6	100	2	89,41
5	1	0.5	100	2	91
6	1	0.45	100	2	94,7
7	1	0.43	100	2	92,59
8	1	0.44	100	2	95,76

se iniciaram por ela. Como forma de verificar se ela é suficiente realizaram-se testes com distâncias inferiores e os resultados foram satisfatórios, visto que todas as taxas de acurácia foram melhores quando a distância foi inferior a 0.6.

Na segunda fase, evidencia-se a influência das variáveis *num_jitters* e *number_of_times_to_upsample* quando a distância é 0.44 pois foi nessas condições que se obteve a melhor taxa de acurácia (95,76%), o que não aconteceu na primeira fase. Também foi possível perceber que quando a distância é muito pequena podem ocorrer erros do tipo classificar uma pessoa do conjunto de treinamento como desconhecida, o que diminuiu a taxa de acurácia.

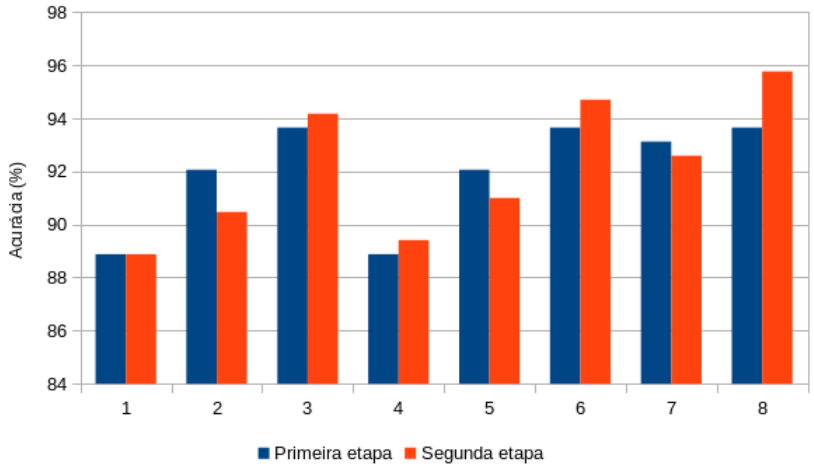

Figura 7. Gráfico de comparação

5. Considerações finais

O presente trabalho aprensentou um estudo sobre o algoritmo KNN como classificador de rostos, usando a linguagem Python e uma biblioteca disponível para tais aplicações. Embora os resultados com as variáveis da biblioteca tenham sido melhores do que os testes da primeira etapa, essas variáveis deixam o algoritmo mais lento, então não é viável para um sistema computacional de baixo custo.

Como trabalhos futuros, pretende-se aplicar técnicas de *Data Augmentation* (aumento de dados), com o intuito de avaliar se tais métodos são válidos para evitar o trabalho de gerar datasets manualmente, e se os resultados obtidos com o reconhecimento de faces é afetado positivamente pelo uso dessa técnica. Outro trabalho válido é a implementação de algum algoritmo de classificação junto ao KNN, de forma a analisar se a acurácia será melhor, igual ou pior que a obtida neste trabalho.

Ainda como futuros trabalhos, pretende-se deixar o sistema de reconhecimento facial mais robusto aplicando a técnica de *Anti-Spoofing*, para evitar falhas na segurança.

Agradecimentos

Os resultados apresentados nessa publicação foram obtidos por meio de atividades de Pesquisa e Desenvolvimento executadas no projeto KAM, com recursos oriundos da Lei nº 8.387/1991, devendo qualquer publicidade fazer referência a citada lei conforme Art. 48 do Decreto nº 6.008/2006, tendo a empresa Samsung Eletrônica da Amazônia LTDA financiado esse projeto nos termos da mencionada lei. A pesquisa foi realizada dentro das instalações do Laboratório de Tecnologia, Inovação e Economia Criativa – LUDUS.

Referências

Ahdid, R., Taifi, K., Said, S., and Manaut, B. (2017). Euclidean & geodesic distance between a facial feature points in two-dimensional face recognition system. *human-computer interaction*, 1:5.

Cover, T. M., Hart, P., et al. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection.

Diniz, F. A., da Silva, T. R., and Alencar, F. E. S. (2016). Um estudo empírico de um sistema de reconhecimento facial utilizando o classificador knn. *Revista Brasileira de Computação Aplicada*, 8(1):50–63.

Goswami, A. (2018). Intro to image classification with knn.

Jiangsheng, Y. (2002). Method of k-nearest neighbors. *Institute of Computational Linguistics, Peking University, China*, 100871.

Kambi Beli, I. and Guo, C. (2017). Enhancing face identification using local binary patterns and k-nearest neighbors. *Journal of Imaging*, 3(3):37.

Khastavaneh, H., Ebrahimpour-Komleh, H., and Hanaee-Ahwaz, A. (2017). Unknown aware k nearest neighbor classifier. In *2017 3rd International Conference on Pattern Recognition and Image Analysis (IPRIA)*, pages 108–112. IEEE.

Kim, J., Kim, B., and Savarese, S. (2012). Comparing image classification methods: K-nearest-neighbor and support-vector-machines. In *Proceedings of the 6th WSEAS international conference on Computer Engineering and Applications, and Proceedings of the 2012 American conference on Applied Mathematics*, volume 1001, pages 48109–2122.

Ojala, T., Pietikäinen, M., and Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59.

Rosebrock, A. (2016). k-nn classifier for image classification.

Setiawan, E. and Muttaqin, A. (2015). Implementation of k-nearest neighbors face recognition on low-power processor. *Telkomnika*, 13(3):949.

Silva, A. L. and Cintra, M. E. (2015). Reconhecimento de padrões faciais: Um estudo. In *Encontro Nacional de Inteligência Artificial e Computacional, 2015, Proceedings ENIAC*, pages 224–231.

Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86.

Wang, Q. and Ju, S. (2008). A mixed classifier based on combination of hmm and knn. In *2008 Fourth International Conference on Natural Computation*, volume 4, pages 38–42. IEEE.

Xu, Y., Zhu, Q., Chen, Y., Pan, J.-S., et al. (2013). An improvement to the nearest neighbor classifier and face recognition experiments. *Int. J. Innov. Comput. Inf. Control*, 9(2):543–554.

Zhao, W., Chellappa, R., Phillips, P. J., and Rosenfeld, A. (2003). Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4):399–458.