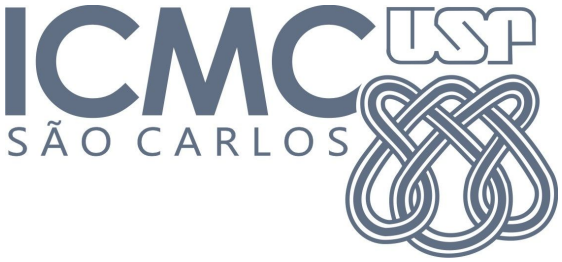


UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO

Processamento de Imagens - SCC0251-2018

Determinação de função em time de futebol por semelhança de face

Destinatário: Prof. Dr. Moacir Antonelli Ponti

Nome: Lucas Yudi Sugi **Número USP: 9293251**
Nome: Ricardo França Fernandes do Vale **Número USP: 9293477**

1. Introdução

Atualmente existem muitos testes no Facebook que permitem verificar com quem você é mais parecido ao realizar a comparação de semelhanças das faces. Dada essa popularidade, este projeto visa determinar a função (atacante, goleiro, lateral, meia, ponta e zagueiro) que o usuário teria como jogador de futebol, através de similaridades encontradas nos rostos dos jogadores profissionais.

2. Objetivo do Projeto

A finalidade deste projeto é estudar e melhorar os conceitos aprendidos em sala de aula sobre as técnicas de processamento de imagens. Para alcançar isso propomos um software que irá classificar o usuário segundo uma posição do futebol, conforme detalhado na seção *1. Introdução* deste documento.

Assim, através de uma foto do rosto fornecida como entrada pelo usuário, iremos analisar fatores como textura, traços e semelhanças gerais em uma base de dados de jogadores das diversas funções. Dessa forma, através de um algoritmo de classificação, será determinada a posição da pessoa a quem pertence a foto da entrada.

3. Divisão de arquivos

A confecção deste projeto está dividida nos seguintes arquivos/pastas:

- main.py: Leitura dos dados.
- filter.py: Filtros utilizados para pré-processamento.
- resize.py: Redimensionamento da imagem.
- segmentation.py: Segmentação da imagem.
- descriptors.py: Extração de características de textura e gradiente da imagem.
- build_dataset.py: Aplicação dos métodos da main em todas as imagens escolhidas previamente para gerar dados para o *dataset*.
- classification.py: Implementação dos algoritmos de classificação para predição da classe da imagem.
- Jogadores: Fotos de todos os jogadores utilizados para comparação.

4. Hospedagem do projeto

O projeto encontra-se hospedado no GIthub, no repositório abaixo:
https://github.com/ricardoffv/Processamento-de-Imagens_Projeto-Final

Importante salientar que as imagens da base de dados também se encontram no repositório e que as mesmas foram extraídas do site:

https://www.futwiz.com/en/fifa18/worldcup/players

5. Execução do código

Para realizar a execução do projeto baixe todos os arquivos do repositório github (link está na seção *4. Hospedagem do projeto*), caso esteja zipado utilize o seu descompactador de preferência para extrair a pasta.

Após isso, utilize o terminal da sua distribuição LInux para acessar o diretório do projeto e faça o seguinte comando:

python3 main.py [nome da foto]

Supondo que o nome da foto seja picture.jpg então o comando ficaria:

python3 main.py picture.jpg

Após a execução, a saída gerada é baseada na posição escolhida de acordo com a predição do algoritmo de classificação:

‘Parabéns! Você é um [posição encontrada, atacante, por exemplo]!

6. Solução

Para alcançar o objetivo de classificar o usuário via semelhança realizou-se uma divisão do projeto em 5 etapas que serão descritas a seguir:

6.1 Pré-processamento

A imagem fornecida pelo usuário provavelmente estará em más condições de luz, já que, o mesmo não está em ambiente controlado para tirar fotos. Além disso, há o fator de ser necessário preparar a imagem para realizar o processamento “bruto” (segmentação e extração de características).

Portanto, para resolver esses problemas foi utilizado:

- Suavização (filtro da média): Permite diminuir o ruído, além de ser interessante para a segmentação por deixar os pixels com valores mais iguais (segmentação será baseada na intensidade) criando regiões mais separáveis.
- Equalização do histograma: Permite melhorar a nitidez da imagem além de converter a imagem para escala de cinza (será útil para a segmentação).
- Filtro Laplaciano da Gaussiana (LOG): Extrai as bordas da imagem, permitindo identificar o rosto do usuário.

6.2 Redimensionamento

Para realizar a comparação da imagem do usuário com as imagens do banco de dados é necessário redimensioná-la para obter tamanhos iguais. Assim, esta etapa foi solucionada da seguinte maneira:

- A imagem é recortada em altura e largura para que suas dimensões sejam proporcionais a 160 pixels (altura e largura das imagens da base de dados);
- Com um filtro, realiza-se uma média dos pixels de modo que há um mapeamento de vários pixels da imagem original para a redimensionada.

Um exemplo de como esta etapa funciona: Supondo uma imagem de tamanho 360 x 360 pixels, então cortamos ela para obter como saída as dimensões 320 x 320 pixels (proporcional a 160). Após isso, é efetuado um mapeamento dos pixels com filtro de 2 x 2 selecionando quatro pixels da imagem e efetuando sua média para calcular um pixel. Isso é aplicado até obtermos um mapeamento de todos os pixels. Note que o tamanho do filtro não é um valor aleatório, ou seja, esse 2 x 2 é calculado. Isso porque, para realizar o mapeamento correto de todos os pixels calcula-se um fator multiplicativo que indica quantas vezes a imagem do usuário é maior que as imagens da base de dados, encontra-se uma proporção portanto. Tal valor é calculado da seguinte maneira: floor(360/160) = 2, para este exemplo em específico.

6.3 Segmentação

A segmentação da imagem na finalidade de identificar o rosto do usuário foi realizada utilizando uma limiarização de intensidades, isso porque a etapa de pré-processamento permite deixar o rosto do usuário com tons brancos e o fundo com tons escuros, tornando possível tal limiarização.

A confeção desta etapa será melhor explicada a seguir:

- A partir da imagem de 160x160 tenta-se extrair uma sub-imagem menor que conterá a face do usuário, permitindo trabalhar de maneira mais determinística, já que, não teremos a interferência de fatores externos.
- Com esta sub-imagem é selecionado um pequeno conjunto de pixels do rosto que serão utilizados para obter uma média de intensidade.
- Com esta média aplica-se a limiarização, obtendo ao final uma máscara que possui as mesmas dimensões da imagem original (160 x 160 pixels) contendo zeros e uns que serão utilizados para selecionar a região de interesse.

6.4 Extração de características

A extração de características baseou-se em obter informações em dois âmbitos: textura e histograma de gradiente. A ideia seria obter semelhanças entre pessoas por meio dos traços faciais, independente de cor, tanto que, para uma obtenção menos complexa de tais valores, a imagem é convertida para uma versão em que possui apenas tons de cinza.

Os descritores de textura foram feitos através da matriz de co-ocorrência, cujas células são computadas com a soma e, por fim, normalização, da quantidade de vezes que, através de um deslocamento de (1,1), a intensidade de cor a encontra-se um pixel abaixo e 1 pixel à direita da cor b, para a e b cores presentes na imagem, que constituem as linhas e colunas da matriz de co-ocorrência.

Na matriz de co-ocorrência, são aplicadas as medidas de Haralick: energia (valor entre 0 e 1 que representa a soma dos quadrados dos pixels da matriz de co-ocorrência), entropia (espalhamento medido de acordo com os valores da matriz), contraste, correlação e homogeneidade.

Por sua vez, o histograma de gradiente é medido através da aplicação do filtro de Sobel na imagem original, que gera duas matrizes e através delas é calculada uma medida de magnitude. O histograma é dividido em 18 pedaços em que há um intervalo de 20 graus para os ângulos entre 0º e 360º, afinal é necessário calcular a arco-tangente individual da razão entre os valores das matrizes Gx e Gy após a aplicação do filtro de Sobel. O histograma é atualizado à medida em que ocorre um ângulo incrementa nesse pedaço do histograma com o valor da magnitude daquele pixel.

São extraídos vinte e três valores de características para cada imagem do *dataset*, que precisa ser criado antes de qualquer outro teste no projeto, justamente, para criar a base de comparação para a última etapa da solução.

6.5 Classificação

A classificação é utilizada para predizer qual rótulo a nova entrada do usuário possuirá. Primeiramente, é necessário criar a base de dados, ou seja, os dados para cálculo utilizados pelos algoritmos de aprendizado de máquina.

As imagens são lidas, arquivo por arquivo, passando pelo processo de pré-processamento, segmentação e extração de características, sendo que este último gera os dados do *dataset*.

Através destes dados é possível realizar o cálculo de distâncias para verificar a proximidade das características de uma imagem de interesse com as imagens previamente calculadas. Sendo assim, são implementados dois algoritmos de classificação: o DWNN (*Distance Weighted Nearest Neighbors*), que realiza a

classificação com rótulos contínuos, baseando-se na distância entre dados de entrada e dados conhecidos, sendo que a distância sofre um processo de balanceamento devido aos pesos calculados pelo algoritmo; e o KNN (*K-Nearest Neighbors*), em que são escolhidos os k vizinhos dos descritores da imagem de entrada mais próximos, que pertencem à base de dados e, dentre esses k vizinhos, a classe predominante ditaria a classe da imagem de entrada.

7. Restrições

A partir da etapa anterior foi possível perceber que certas condições devem ser satisfeitas no intuito de gerar uma melhor classificação para o usuário. É extremamente recomendado que:

- É necessário evitar pontos de iluminação muito fortes ou muito baixos em direção ao rosto.
- Obtenha-se fotografia o mais próxima possível de medidas quadradas, ou seja, largura e comprimento de tamanho similar, pois, fotos muito retangulares, em específico, fotos com uma altura muito maior que a largura não obtém bom desempenho de redimensionamento e segmentação. Além disso, deve ter resolução maior ou igual a 160x160 pixels, que é o tamanho das imagens do *dataset*, para evitar problemas de cálculo de extração de características, mais precisamente para o histograma de gradientes.
- O rosto do usuário esteja centralizado, pois isso garante que o redimensionamento assim como a segmentação consigam extrair a região necessária.
- O fundo seja “limpo” e com uma cor clara, ou seja, possua apenas uma cor (branco de preferência) sem haver muitos detalhes.

Caso essas condições não sejam atendidas não há garantias da veracidade do resultado final.

8. Testes

As etapas deste projeto assim como suas respectivas soluções já foram descritas na seção *6. Soluções*. Contudo, é importante salientar que foram realizados diversos testes para determinar quais seriam os melhores algoritmos e técnicas para o projeto.

Assim, nas próximas subseções explicaremos os testes realizados. Por fim, para facilitar a estruturação deste documento, abaixo encontra-se a imagem original, extraída do banco de dados, que foi utilizado para testes:

Figura 1: Imagem original.

8.1 Pré-processamento

Para essa parte foram implementados as seguintes técnicas:

- Suavização (Filtro da média): De uma maneira geral notou-se que a suavização auxilia na etapa de segmentação por fazer com que os pixels tendam a mesma média. Deve-se notar que as vezes ocorre uma piora na qualidade visual da imagem, mas isso não é importante visto que desejamos obter melhores resultados para a segmentação. Abaixo temos o resultado desse filtro:

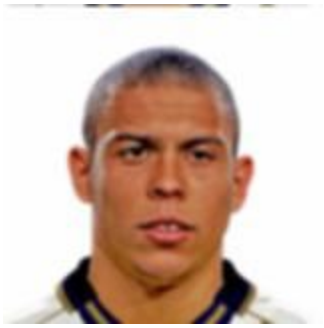


Figura 2: Suavização com um filtro da média de tamanho 3.

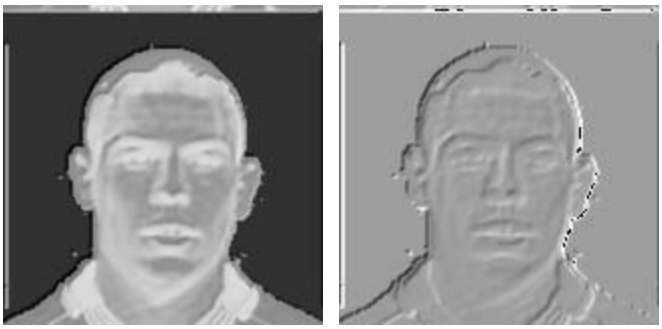
- Ajuste gamma/Equalização de histograma/High Boost: Essa três técnicas foram implementadas para tentar melhorar a qualidade visual da imagem ou ajudar na segmentação. Aquela que se saiu melhor foi a equalização do histograma por ela permitir melhorar a nitidez da imagem assim como “convertê-la” para escalas de cinza (isso porque esse processamento não foi realizado no espaço HSV). As duas outras não causam melhorias tão significativas. Abaixo encontra-se o resultado de cada uma delas aplicada após a suavização:

A B C

Figura 3: A) Equalização de histograma. B) Ajuste gamma. C) Filtragem High Boost.

Pode-se perguntar porque não utilizou-se o ajuste gamma visto que ele parece auxiliar a distinguir as intensidades da imagem. Deve-se notar que o fundo apenas ficou preto por uma propriedade matemática da função potência, sendo que em fundos diferentes o resultado não seria o mesmo. Assim, vale dizer que o sua principal função é apenas melhorar o brilho da imagem. Dessa forma, a melhor escolha ainda é a equalização do histograma por realizar uma conversão de cores e ajustar a nitidez da imagem.

- Operador Sobel/Log: Com o intuito de realizar uma extração de bordas, essas duas técnicas foram implementadas. Abaixo encontra-se o resultado de cada uma delas:



A B

Figura 4: A) Log. B)Operador Sobel.

Note que após aplicar a suavização e a equalização de histograma, obtemos melhores resultados utilizando o log. Isso porque, o rosto do jogador fica em tons de branco enquanto que o fundo fica em tons escuros. Deve-se salientar que isso quase

sempre ocorre quando são atendidas as restrições estabelecidas na seção *6. Restrições*, mesmo para jogadores que são negros e o fundo seja um pouco mais escuro.

8.2 Redimensionamento

Não foram realizados muitos testes nesta etapa visto que apenas uma técnica de redimensionamento foi implementada. De qualquer forma, abaixo encontra-se o resultado do redimensionamento de uma imagem 652 x 408 pixels para 160 x 160 pixels.

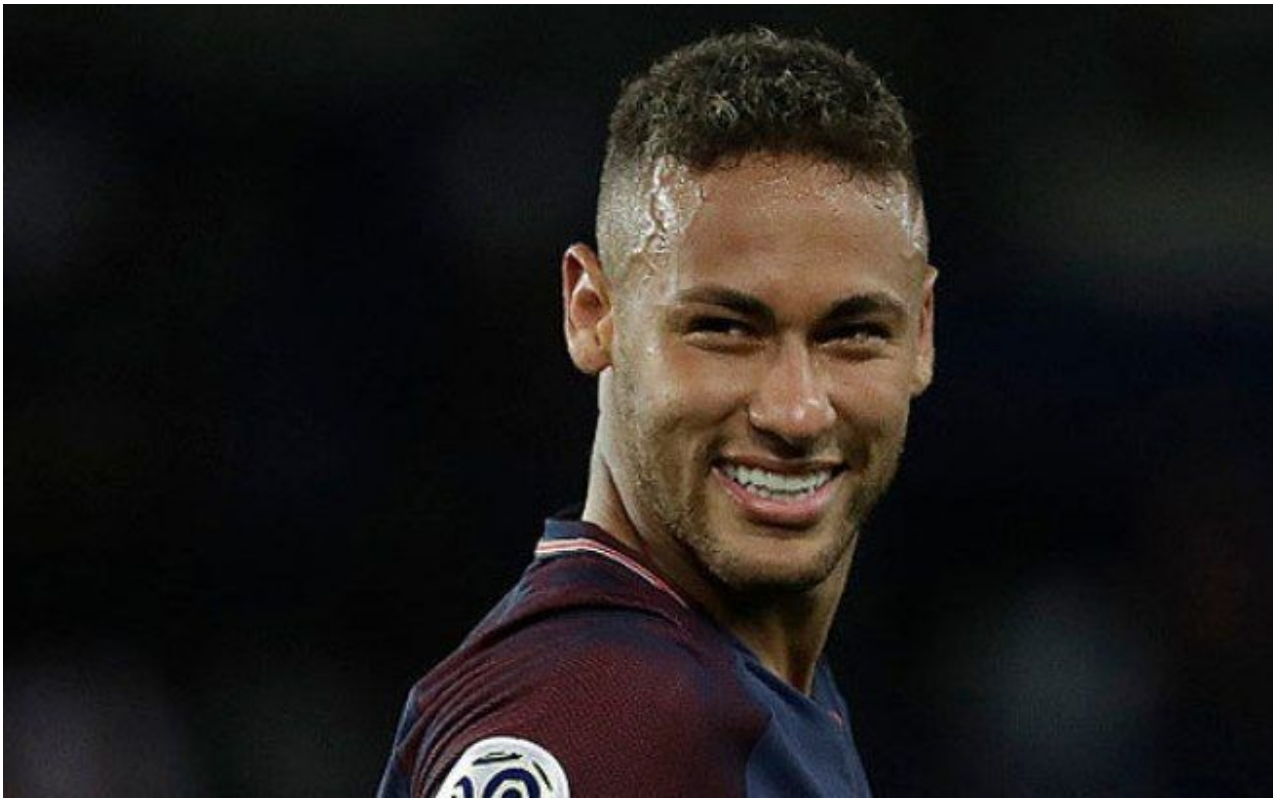


Figura 5: Foto original.

Figura 6: Foto redimensionada.

8.3 Segmentação

Para esta etapa utilizou-se uma segmentação baseada em intensidades visto que a figura 4A possui o rosto bem separado do fundo com relação a cor. Desse modo, os testes que foram realizados basearam-se em encontrar o threshold (limiar para inclusão de pixel) adequado assim como a melhor região para se extrair pixels.

Assim, os melhores valores dado vários experimentos é utilizar um limiar de 200 com uma região de 25 x 25 pixels. Abaixo temos o resultado desta etapa:

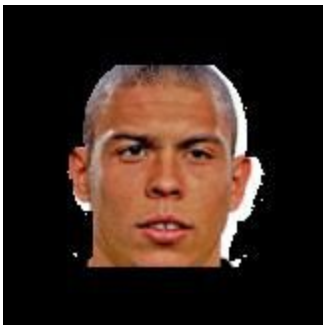
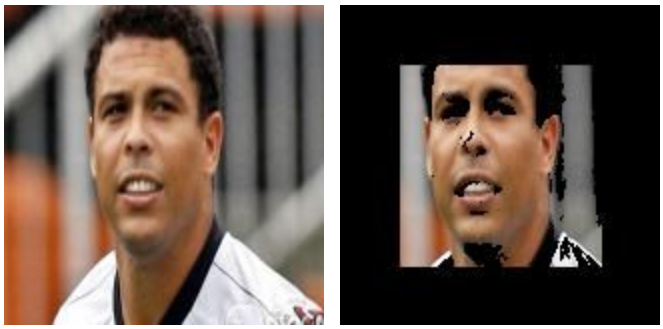


Figura 7: Imagem segmentada.

Note que os pixels em preto são aqueles que não foram selecionados pela segmentação, os demais são os eleitos.

Para uma melhor compreensão de como é necessário atender as restrições propostas para este trabalho, abaixo temos uma imagem ruim que demonstra qual seria o resultado após todos esses processos:



A B

Figura 8: A)Imagem ruim original. B) Imagem segmentada.

Note que como o fundo da figura 8A possui bastante ruído além de ter bastantes detalhes (grade), faz com que o resultado da segmentação seja péssimo pois o fundo

acaba sendo selecionado como sendo considerado o rosto. Isso irá afetaria a etapa de extração de características assim como a classificação final.

8.4 Extração de características

Todos os descritores de textura e de histograma de gradiente foram utilizados, por serem os dados de interesse das imagens para comparação, conforme dito na seção *6.4 Extração de Características*.

Foi possível perceber que nenhum dos valores descritivos burlou as especificações matemáticas que regem a corretude das técnicas, tanto na aplicação para preenchimento do *dataset*, quanto na extração de características da imagem de entrada, ou seja, energia e homogeneidade ficaram no intervalo [0,1], entropia ficou entre os valores de 0 e 16 (que é o logaritmo na base 2 da intensidade máxima em escalas de cinza multiplicado por 2).

Sabendo que os descritores seriam os dados decisivos para a classificação, analisando de maneira geral os dados, os descritores de textura mostraram valores muito parecidos entre si (principalmente a entropia), sendo assim, os descritores de gradiente demonstraram-se como o maior poder de decisão para a classificação das imagens.

8.5 Classificação

Tendo em vista que utilizamos tanto classificadores contínuos como discretos, a escolha de implementação culminou na representação numérica do rótulo da posição do jogador, em que classe 0 representaria atacante; classe 1, goleiro; classe 2, lateral; classe 3 indicaria um meio-campista, classe 4, um ponta (atacante lateral) e classe 5, um zagueiro.

Primeiramente, a classificação da imagem de entrada foi realizada com o algoritmo DWNN, que, mesmo tendo como vantagem observar características em comum de certos dados com o mesmo rótulo em vez de apenas calcular as distâncias, a técnica gera rótulos contínuos e, até para os dados de treino, ou seja, do próprio *dataset*, gerava apenas rótulos muito próximos de 2.5, variando para mais e para menos em um intervalo quase insignificante.

Isso deve-se tanto à escolha de pesos para o algoritmo, que não tiveram tratamento posterior, quanto à escolha do parâmetro sigma, que, quanto maior for, mais próximo da média o rótulo inferido irá se encontrar.

Dada a ineficiência deste último algoritmo, foi utilizado o preditor discreto KNN, em que k=1, ou seja, foi escolhido como rótulo, aquele que possuir alguma instância que

possa ter a menor das distâncias, entre os dados de treino, em relação às características da imagem de entrada.

O 1NN foi escolhido para realizar a classificação discreta de jogadores. Foram escolhidos alguns jogadores, tanto existentes na base de dados, quanto inéditos nela.

Tal técnica permite que não haja erros para classificar dados de treino, porém, os dados de teste culminam em uma pífia taxa de acertos. Dentre os jogadores testados, têm-se Pelé, Nedved, Ronaldinho Gaúcho, Lucas Paquetá, ambos, meio-campistas, eles foram caracterizados como jogadores defensivos (zagueiros e laterais); isso expressa a dificuldade de haver um jogador classificado como meia, além da ineficiência de classificação. Em contraposição, o atacante lateral Vinícius Jr. foi caracterizado como ponta, gerando ao menos uma predição correta.

Podem-se citar alguns fatores que permitem que haja ineficiência na escolha de classe, como má escolha do espaço algébrico da base de dados, desatenção em relação à diversidade de conteúdo de cada rótulo ou, até mesmo, ausência de pesos escolhidos corretamente para a definição do classificador ser eficaz.

9. Conclusão

O projeto em questão pode ter uma temática não funcional em relação à pesquisa científica, porém, técnicas do mesmo remetem à área de reconhecimento facial, utilizado para questões de segurança, à área de aprendizado de máquina, que tem algoritmos mais robustos e precisos para classificação que poderiam aumentar a acurácia, entre outras ramificações. Além disso, o processo de extração de características é bem extenso e complexo, sendo que envolve métodos anteriores ao cálculo dos descritores que, por si só, já são uma área de pesquisa, como a segmentação.

Existem diversas melhorias, no âmbito de Processamento de Imagens, que poderiam ser feitas, como por exemplo, a validação dos descritores de Haralick, outros métodos de análise de textura e de gradiente, além de possíveis melhorias no redimensionamento. Porém, crê-se que a maioria dos erros de classificação deveu-se ao fato do não determinismo das fotos utilizadas como entrada, em relação às fotos bem tratadas do *dataset*.