Trabalho Prático 2

David Camara (42414)
Diogo Patrício (42319)
Turma (51D)

Unidade Curricular Processamento de Imagem e Visão
Licenciatura em Engenharia Informática e Multimédia
Instituto Superior de Engenharia de Lisboa
www.isel.pt

Lisboa, 27 de dezembro de 2017

Índice

1-Introdução

1.1-Descrição

- Pretende-se realizar uma animação gráfica de um objeto simples, através da estimação e classificação dos movimentos de uma mão registados numa sequência de vídeo.
- O código desenvolvido deverá ser desenvolvido em Python/OpenCV e permitir a execução da aplicação em tempo real para interação pessoa-máquina.

1.2-Objetivos

Neste trabalho pretende-se desenvolver um algoritmo de estimação e classificação dos movimentos detetados, para que estes sejam usados numa aplicação gráfica interativa.

O algoritmo a ser criado é caracterizado pela seguinte sequência:

1. Leitura das Imagens.
2. Deteção da região da mão.
   2.1. Deteção do tom de pele com base no espaço de cor RG normalizado.
3. Correspondência de regiões entre imagens consecutivas.
4. Deteção dos pontos de interesse e correspondência nas imagens seguintes.
5. Classificação dos diferentes tipos de movimentos (direita, esquerda, cima, baixo, zoom in e zoom out).
6. Animação de um objeto gráfico com base na sequência de movimentos obtidos no ponto anterior.

2-Desenvolvimento

2.1-Deteção e seguimento da mão

2.1.1-Diferentes tentativas nesta fase do trabalho

Antes de tentarmos utilizar a cor para deteção da mão, realizámos a deteção de movimento com base em subtração de imagens, tendo sido refutada rapidamente, devido a esta forma de deteção não ser a melhor para trabalhos como este, como a palma da mão, ocupa uma área grande, quando movemos a mão por exemplo para a direita, este método deteta a mão nas extremidades da mão, mas no centro da mão cria um buraco devido ao movimento ser lento e a palma da mão, parte esquerda fica sobre a parte da mão direita.

Com a subtração de imagens consecutivas não conseguíamos encontrar forma de identificar certos movimentos com o zoom in e o zoom out. Além disso, o movimento da câmara ou de qualquer objeto poderia ser classificado como movimento, o que não era pretendido.

Seguidamente, foi usada a cor HSV, devido a termos encontrado informação relacionada com o intervalo da cor da mão nesse Sistema de Cores. Mas devido aos valores de limiar para deteção de pele serem dados em "hardcode", bastava mudar de sitio, que tivesse uma iluminação um pouco diferente para a deteção de pele começava a falhar.

Viramos para o RG normalizado, com as aprendizagens obtidas no sistema de cores HSV, sabendo que teríamos os mesmos problemas que no Sistema HSV de determinação dos limiares de cor de pele.

A solução foi criar um quadrado na imagem em que pedimos ao utilizador para colocar pele no quadrado, e desse quadrado converter de BGR para RG normalizado e fazer a media de cada componente, de forma a ficar com um valor para R e um valor para G, e dando uma margem inferior e superior de 0.05, obtivemos os melhores resultados para deteção de pele em diferentes ambientes de iluminação.

2.1.2-Deteção da região da mão

Para que o programa consiga detetar a mão do utilizador foi criado uma função para interagir com o utilizador, de forma a que este coloque numa região específica a sua mão para que seja calculado qual é a sua cor de pele (Figura 1).

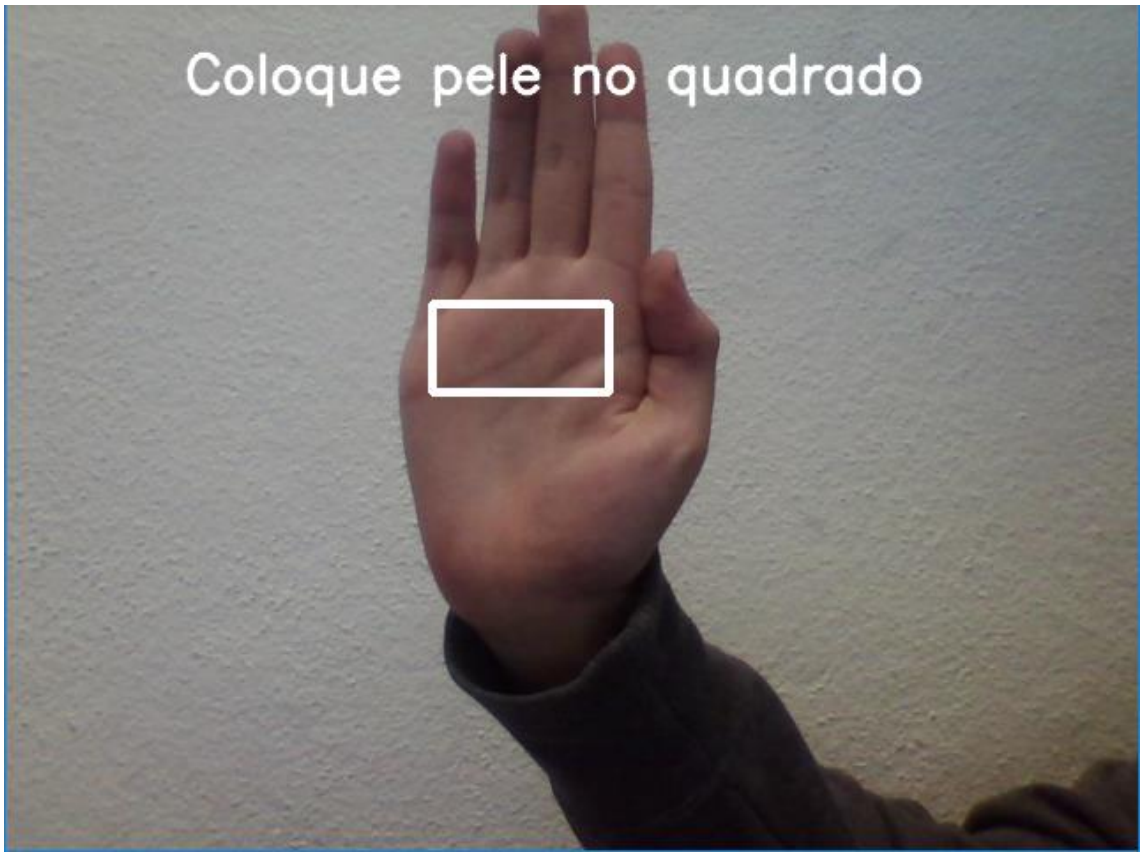

Figura 1:Imagem para interagir com o utilizador

Utilizando a imagem tirada pelo utilizador (Figura 1), o programa irá utilizar a Figura 2 para converter os seus pixéis em RG normalizado e poder calcular a média dos pixéis da área, desta forma, o programa poderá reconhecer a sua cor de pele de forma muito precisa.

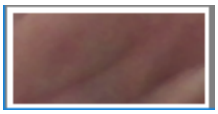

Figura 2:Parte da imagem guardada a ser analisada

2.1.3-Correspondência de regiões entre imagens consecutivas

Após o cálculo do valor médio da cor de pele em RG normalizado, as regiões detetadas seriam as que tivessem valores próximos à média calculada anteriormente.

Os componentes B e G do RG normalizado é calculado para a imagem atual e ficam ativos os pixéis com valores próximos da cor de pele medida. De seguida, as duas componentes calculadas (Figura 3, Figura 4) são juntas, ficando apenas como pixéis ativos os que aparecem em ambas as componentes (Figura 5).

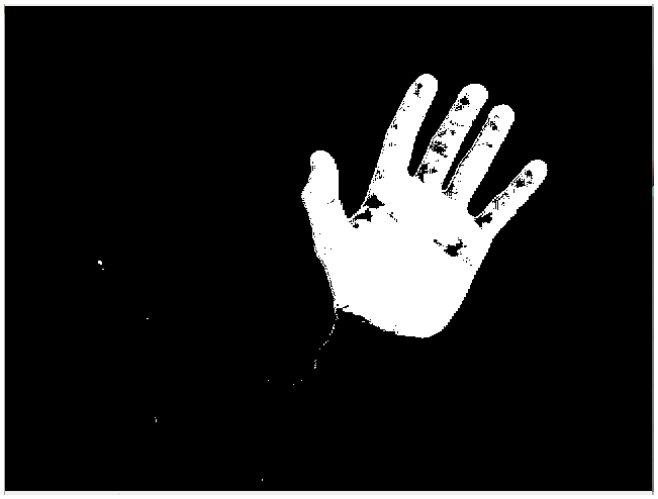

Figura 3: Pixéis dentro do intervalo de ton de pele da componente R

Figura 4:Pixéis dentro do intervalo de ton de pele da componente G

Além disso, é usado na Figura 5 os operadores morfológicos de dilatação e erosão para melhorar a imagem, permitindo eliminar ruído e de seguida fechar pequenos buracos dentro da região da mão.

Figura 5: Imagem usada como máscara com os pixéis dentro do intervalo de ton de pele

2.2-Algoritmo esparso de cálculo do campo de movimento

2.2.1-Deteção de pontos de interesse na primeira imagem

Para a deteção de pontos de interesse é usado o goodFeuturesTotrack do OpenCV, é passado a esta função uma imagem em tons de cinzento, no nosso caso, como queremos seguir a mão do utilizador, aplicamos a máscara criada anteriormente (2.1-Deteção e seguimento da mão), à imagem do vídeo, e o goodfeuturesToTrack, retorna os pontos que considere melhor para seguir, preferindo os pontos que estejam situados nos cantos.

Ao longo dos movimentos da mão, se for feito movimentos mais rápidos, perde-se os pontos retornados pela função goodfeuturesToTrack, causando dificuldade a seguir a mão. Deste modo quando deteta menos do que 5 pontos, é recalculado novos pontos para fazer o seguimento do movimento.

2.2.2-Determinação de correspondências na segunda imagem

A correspondência de pontos é feita através da função calcOpticalFlowPyrLK, pesquisando os pontos da imagem anterior na imagem atual, retornando os mesmos pontos nas novas posições em que foram encontrados (Figura 6). A função para indicar os pontos comuns às duas imagens devolve o status dos pontos, ficando a 0 os pontos perdidos e os encontrados ficam a 1. Deste modo, os que não foram encontrados são eliminamos, continuando apenas com os que foram encontrados desde que o número de pontos encontrados seja superior a 5 pontos. Se tal não acontecer os pontos serão recalculados.

Figura 6: Pontos detetados na imagem atual que foram detetados na imagem anterior

2.3-Classificação dos diferentes tipos de movimento

2.3.1-Movimentos Horizontais e Verticais

A classificação dos movimentos Horizontais e Verticais é feito através dos pontos encontrados pelo 2.2-Algoritmo esparso de cálculo do campo de movimento, como temos os pontos, na imagem anterior e temos os mesmos pontos na imagem atual, conseguimos através da distância saber para que direção estão a ir. Se no eixo de coordenadas x a distância obtida for negativa, quer dizer que o movimento está a ser feito para a esquerda e se positivo o movimento está a ser para a direita, o mesmo acontece para o eixo de coordenadas y, correspondendo os valores positivos ao movimento para baixo e os valores negativos ao movimento para cima.

2.3.2-Movimento de Zoom

No caso do Zoom, é calculada a diferença de área da mão, usando as caraterísticas da máscara criada para a 2.1-Deteção e seguimento da mão, entre imagens consecutivas. Se a diferença entre áreas for negativa significa que é feito um zoom in, caso seja positiva representa um zoom out. Para evitar pequenas diferenças de áreas entre imagens, devido à máscara e não devido a movimentos da mão, é feito uma média dos dez últimos cálculos da área, segundo o movimento calculado em cada imagem, para permitir que se tenha resultados mais fidedignos.

Nota: Na classificação dos diferentes movimentos cada direção é representada quer seja por um valor positivo ou negativo. Devido à deteção de pequenos movimentos, que não se deseja que sejam detetados, foi verificado a partir de que valores se poderia considerar movimento, por visualização e tentativa erro, sendo estes valores diferentes para os movimentos Horizontais, Verticais e de Zoom.

2.4-Animação de um objeto gráfico

Utilizamos as funções do OpenCV no Python para a criação do objeto gráfico. É usado o tamanho da imagem do vídeo para ser igual ao espaço a que o objeto se pode mover. Além disso, utiliza-se os pontos detetados pelo 2.2-Algoritmo esparso de cálculo do campo de movimento para saber o centro do objeto gráfico, pela média da posição dos pontos e usa-se o raio da área, calculada para o 2.3.2-Movimento de Zoom, para alterar o raio do objeto animado.

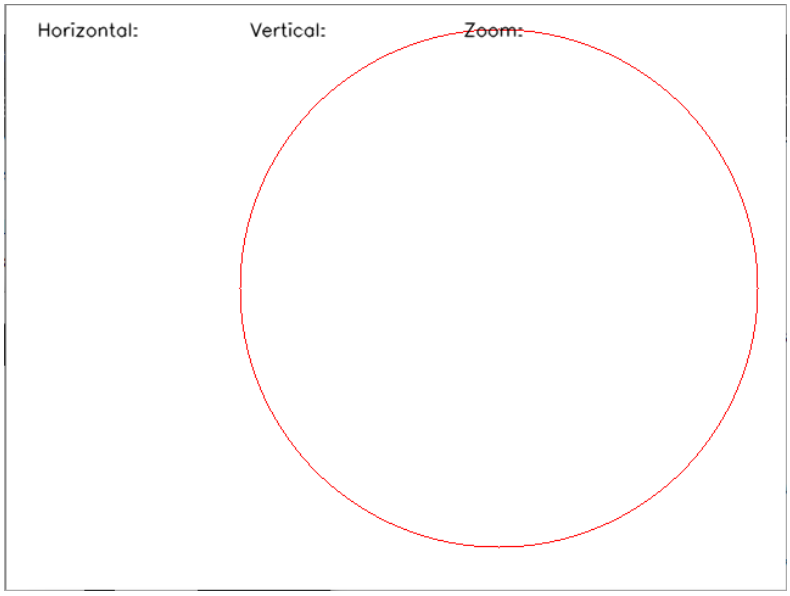

Figura 7:Objeto gráfico animado pelos movimentos detetados

Na Figura 7 é ainda colocado os tipos de movimentos detetados no momento. Desta forma, o utilizador pode sempre compreender os movimentos que o programa está a detetar.

3-Conclusão

A aplicação criada apresenta alguns problemas a detetar a presença de zoom, podendo sido provocado pela dificuldade de detetar as diferenças de valores de área entre as imagens consecutivas, dado que essa dificuldade possa ser vir da qualidade da câmara que transmite o vídeo ou das imperfeições do código desenvolvido.

Por outro lado, a deteção da cor de pele pode ser bastante alterada conforme o que for colocado na Figura 1 pelo utilizador, sendo, neste caso, existir uma interação do utilizador com a aplicação de forma a que este consiga melhorar a sua animação do objeto.

Também, o trabalho, ao poder funcionar com qualquer parte do corpo, desde que tenha a mesma cor, é prejudicada quando são visíveis na imagem mais do que uma parte do corpo ao mesmo tempo, podendo considerar que todas as partes correspondem a uma próxima da câmara ou que existiu um movimento instantâneo nesse momento.

Para estas situações onde aparece várias partes do corpo, a solução poderia ser passar por eliminar os pontos que com o passar de muitas imagens não se movam muito, ficando só com os pontos bons na mão devido ao movimento. Não foi implementado esta solução.

Para a deteção de movimentos horizontal e vertical, durante o processo de desenvolvimento da aplicação deparamos com dois métodos para poder classificar os movimentos, um foi o implemento neste mesmo trabalho através de calculo de pontos de interesse na mão. Que tem como vantagem, ser mais resistente ao ruido da máscara criada pela deteção de cor de peles. E desvantagem, necessidade de maior processamento, e complexidade.

Outra solução poderia ser, calcular o centroide da região da mão e através desse ponto, classificar os movimentos, mas constatamos ter pior resultados com este método, falhando muito, devido ao ruído da máscara dos tons de pele. Mas tendo a vantagem de ser um método mais simples para a solução do enunciado.

4-Bibliografia

Deteção de cor de pele usando HSV:

https://www.pyimagesearch.com/2014/08/18/skin-detection-step-step-example-using-python-opencv/

Deteção de pontos de interesse:

https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_shi_tomasi/py_shi_tomasi.html

Determinação de correspondências dos pontos nas imagens seguintes:

https://docs.opencv.org/3.3.1/d7/d8b/tutorial_py_lucas_kanade.html

Funções usadas (OpenCV):

- bilateralFilter
  https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html
- inRange e bitwise_and
  https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html
- goodFeaturesToTrack
  https://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html
- calcOpticalFlowPyrLK
  https://docs.opencv.org/2.4/modules/video/doc/motion_analysis_and_object_tracking.html