Projeto 2-2: Contagem de pessoas

Bruna Mayumi Kimura
Visão computacional

1. Introdução

Este projeto consiste em fazer um programa capaz de contar a quantidade de pessoas em uma sequência de frames. Para tanto, foi necessário definir previamente o que seria considerado como fundo, e a partir deste treinamento, descobrir os objetos estranhos, ou seja, as pessoas. O treinamento de fundo foi feito utilizando a técnica de codebook, baseado no artigo "*Real-time foreground-background segmentation using codebook model*"[1]. Já a contagem de pessoas, foi utilizada a função do próprio OpenCV de detecção de contorno e área de contorno.

2. Definição do fundo utilizando o codebook

Este primeiro passo foi inteiramente baseado no algoritmo de codebook do artigo. Para o programa entender o que é o fundo foi feito uma espécie de treinamento utilizando codebook. Para cada pixel de cada imagem foram feitos os codewords correspondentes, estes foram armazenados em uma lista, chamada de C (conjunto de codebook).

A primeira iteração ocorre para cada uma das 'N' imagem, em seguida o programa itera sobre cada um dos seus pixels. O conjunto de codebooks, possui o mesmo tamanho da matriz da imagem, assim, há um codebook para cada um dos seus pixels. Já cada codebook recebe um conjunto de codewords. Cada codeword recebe um vetor (v) de cores R, G, B e uma tupla de seis itens (aux). São eles: luminosidade mínima e máximo (I_{min} e I_{max}), frequência que o codeword ocorre (f), maior tempo em que o codeword NÃO apareceu (ƛ), o primeiro acesso em que o codeword ocorreu (p) e por fim, o último acesso em que o codeword ocorreu (q).

Sabendo disso, ocorre a checagem dos codewords, para tanto, foram feitas duas funções. A primeira é a colordist que é responsável por ver a distorção das cores, ela recebe o valor RGB atual e o RGB do codeword daquele pixel e calcula a distorção, segundo a equação 1. A segunda é a brightness, esta recebe o valor de RGB atual e o luminosidade mínima e máxima armazenada no codeword. Esta função é responsável por checar se a luminosidade atual, está entre a luminosidade baixa e alta pré-determinada, caso seja verdade ela retorna True, caso contrário False como mostra a equação 2. Dessa forma, caso colordist seja menor que o um coeficiente Ɛ

pré-definido e o brightness for verdade o valor do codeword é redefinido segundo uma fórmula representada pela equação 3. Caso contrário, é adicionado os valores atuais no codeword, os vetores RGB e a tupla aux.

Após finalizar todas as iterações das 'N' imagens, o próximo passo é atualizar o lambda, segundo a equação 4.

A última iteração necessária é o filtro temporal. Este checa se o valor do lambda em cada codebook é menor ou igual que a metade da quantidade total de imagens, ou seja N/2. Um novo codeword é atualizado apenas com os valores onde a condição de cima é verdade.

3. Detecção das pessoas

Para detectar as pessoas no fundo da imagem foi feito o algoritmo de subtração de fundo baseado no mesmo artigo citado na introdução. A primeira etapa foi criar uma imagem do mesmo tamanho que as imagens de fundo totalmente preta. Cada pixel das imagens foram enviadas para uma função para ser feita a mesma checagem de colordist e brightness da etapa anterior. Caso a checagem desse verdade o valor do codeword foi atualizado da mesma forma (equação 3) e a função retorna True, caso contrário apenas retorna False.

Assim, caso a função retornasse falso, significa que o fundo não é igual, ou seja, aquele pixel representa um objeto estranho e portanto é pintado de branco. Por outro lado, caso a função retorne verdade, quer dizer que o fundo é igual e portanto aquele pixel deve continuar preto. Dessa forma, no final das iterações deve sobrar uma imagem onde o fundo é preto e os objetos estranhos (pessoas) estão em branco.

4. Contagem de pessoas

Para contagem de pessoas foi utilizada a função de contorno (findContours)[2] mais a função de área de contorno (contourArea)[3], dos tutoriais do OpenCV. Primeiramente, utilizou-se a função de findContours, de todos os contornos encontrados, foi selecionado apenas as que possuíam áreas maiores que 100 e menores que 400. Para tanto, foi utilizada a função de contourArea.

Outra tentativa para a contagem de pessoas foi a função do próprio OpenCV, "SimpleBlobDetector"[4]. Para conseguir tirar o ruído da imagem foi utilizada a técnica de erosão e dilatação. Primeiramente a imagem foi erodida com um kernel de 5 por 5. Em seguida foi feita a dilatação com o mesmo kernel. Com a imagem mais "limpa" foi utilizado a função do blob detector. Porém, esta função não se mostrou boa, pois perde-se muito ao fazer a erosão e dilatação e as pessoas próximas acabam se tornando uma sombra só.

5. Erros e melhorias futuras

Não consegui implementar corretamente a detecção para várias iluminações. O codeword só funciona se os fundos forem de tempos parecidos. Dessa forma, as imagens utilizadas para o testes são semelhantes em iluminação e cor com aquelas utilizadas para detectar multidões. A pasta bg_fixo, possui essas imagens semelhante em tempo da simulação, já o bg_random possui imagens de tempos distintos. Apenas o treinamento da primeira pasta funcionou.

O ε_2 precisou ter um valor extremamente alto (5000). A contagem não está perfeita, pois algumas pessoas são consideradas uma só. O filtro temporal também não está funcionando, já que quando aplicado ele exclui todos os codewords.

A biblioteca do pickel quando utilizada bastante imagens no teste (~20 imagens) não era possível salvar, pois ocorria um problema de memória.

Como melhoria, poderíamos arrumar os itens acima não finalizados.

6. Funções

$$X_t^2 = R^2 + G^2 + B^2$$
$$V_i^2 = Ri^2 + Gi^2 + Bi^2$$
$$< V_i , X_t >^2 = (Ri \cdot R + Gi \cdot G + Bi \cdot B)^2$$

$$\rho^2 = \frac{<V_i , X_t>^2}{V_i^2}$$
$$colordist = \sqrt{X_t^2 - \rho^2}$$

equação 1

$$I_{low} = \alpha \cdot I_{max}$$
$$I_{hi} = min\{\beta \cdot I_{max}, \frac{I_{min}}{\alpha}\}$$
$$brightness = \{true : \ if\ I_{low} \leq |Xt| \leq I_{hi}\}$$
$$brightness = \{false : \ otherwise\}$$

equação 2

$$V_m = \left(\frac{f_m \cdot R_m + R}{f_m + 1}, \frac{f_m \cdot G_m + G}{f_m + 1}, \frac{f_m \cdot B_m + B}{f_m + 1} \right)$$
$$aux_m = (min\{I, I_{min}\}, max\{I, I_{max}\}, f_m + 1, max\{\lambda_m, t - q_m\}, p_m, t)$$

equação 3

$$\lambda_m = max\{\lambda_i, (N - q_i + p_i - 1)\}$$

equação 4

7. Bibliografia

[1] KIM, K.; CHALIDABHONGSE, T. H.; HARWOOD, D.; DAVIS, L. **Real-time foreground-background segmentation using codebook model**. ELSEVIER. 2005 vol: 11 pp: 172-185.

[2] Finding contours in your image. OpenCV. Disponível em: <https://docs.opencv.org/2.4/doc/tutorials/imgproc/shapedescriptors/find_contours/find_contours.html> . Acesso em: 8 de outubro de 2018.

[3] Contour Features. OpenCV. Disponível em <https://docs.opencv.org/3.4/dd/d49/tutorial_py_contour_features.html>. Acesso em: 8 de outubro de 2018.

[4] Blob Detection Using OpenCV (Python, C++). Learn OpenCV. Disponível em: <https://www.learnopencv.com/blob-detection-using-opencv-python-c/>. Acesso em: 8 de outubro de 2018.