ADDETC – Área Departamental de Engenharia Eletrónica e Telecomunicações e de Computadores

LEIM -Licenciatura Engenharia informática e multimédia

Processamento de Imagem e Visão

Trabalho laboratorial 2

Turma:

LEIM-51D

Trabalho realizado por:

Duarte Domingues Nº45140

Pedro Henriques Nº45415

Docente:

Pedro Mendes Jorge

Índice

Introdução

Este trabalho foi realizado no decorrer da disciplina de Processamento de Imagem e visão, com o objetivo proposto de:

“Desenvolver algoritmo capaz de detetar, e classificar, zonas da imagem onde ocorreram movimentos de objetos.”

Neste trabalho iremos desenvolver um algoritmo capaz de ser integrado num sistema de vídeo vigilância autónomo. Para tal o algoritmo deve ser capaz de identificar o movimento de objetos (regiões ativas) de interesse, tais como pessoas ou carros e classificar esses objetos numa de 3 classes possíveis (Pessoa, carro, outra). Após identificar com sucesso o objeto cada objeto deve ter o seu próprio identificador e a sua trajetória deve ser sobreposta ao vídeo original de modo a permitir a posterior análise.

Neste trabalho iremos utilizar a linguagem de programação Python, em conjunto com as bibliotecas Numpy pela sua capacidade de manipulação de matrizes listas e números, e a biblioteca OpenCV para manipular imagens e vídeos.

Este relatório ira explicar o processo de desenvolvimento e algumas das escolhas realizadas durante o desenvolvimento deste projeto.

Desenvolvimento

Extração do fundo

Neste trabalho o plano de fundo será composto por todas as regiões que não estão em movimento, independentemente da distância á camara e da sua relação com os planos da imagem. Desta forma, e construindo em cima daquilo que foi descrito no trabalho pratico anterior, a extração do fundo irá consistir na aplicação de uma máscara onde as regiões a manter irão corresponder àquelas regiões onde existe movimento.

Existem diversas formas de calcular a máscara de fundo, por exemplo uma simples diferença entre imagens com uma subtração em absoluto. Desta forma um *pixel* com variação nula, ou muito perto de nula, entre duas ou mais *frames*, são tidas como parte do fundo e não são consideradas para a região ativa. Numa versão inicial do projeto este foi o método utilizado para extrair o fundo, no entanto não proporcionava robustez, ou seja, não lidava bem com movimentos pequenos (pequenos ramos de arvores a balançar ao vento e artefactos da codificação de vídeo). Por este motivo foi necessário utilizar um método mais robusto e que não leve em consideração apenas unicamente a frame anterior.

Assim sendo, optámos por utilizar um dos algoritmos presente na biblioteca OpenCV, *BackgroundSubtractorMOG2* este algoritmo possui 3 argumentos fundamentais, *history* quantas frames anteriores a serem consideradas, para o cálculo do fundo, *varThreshold* valor a partir do qual a diferença é contabilizada como pixel ativo e finalmente *detectShadows* que não foi utilizado no nosso trabalho. No entanto este algoritmo não é á prova de falhas, pois se um objeto ficar parado no mesmo ponto por mais de N frames o objeto é incluído como parte do fundo. No entanto este algoritmo fornece resultados bastante bons para detetar as regiões ativas.

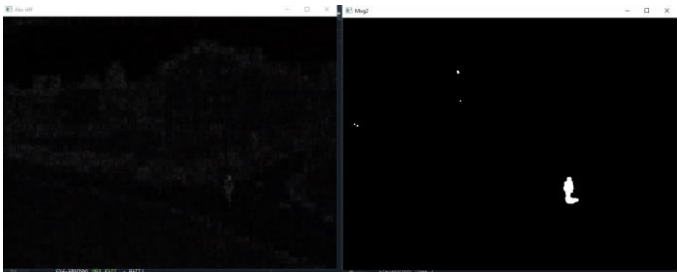


Figura 1 - Comparação métodos de subtração de fundo. Á esquerda diferenca absoluta, á direita Mog2

No entanto, e independente dos métodos testado existe sempre algum ruido aquando da deteção do fundo, nomeadamente pontos isolados que não correspondem a nenhum objeto e que possivelmente nem sequer ligariam a outros pontos. Para resolver estes problemas procedemos á suavização de cada frame depois de ser feita a extração do fundo através de mecanismos *Blur.* Através destes mecanismos é feita uma mediana para cada pixel com todos os vizinhos mais próximos pelo que pequenas inconsistências são rapidamente eliminadas.

Filtro de mediana – suavização de imagens (blur)

O filtro de mediana é um método de suavização de imagens em que o valor de um pixel P(i,j) é tido como o valor da mediana do *kernel* envolvente. Desta forma removemos regiões ativas de pequenas dimensões presentes no vídeo. O método do OpenCV medianBlur implementa esta média para toda a imagem fornecendo a imagem filtrada.

A equação seguinte demonstra como se comporta o filtro de mediana com um *kernel* de tamanho 3.

$$P(i,j) = mediana \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} = 1$$

No caso da máscara utilizada para extração do fundo o filtro de mediana é muito útil pois remove o ruido "sal e pimenta" em que pequenos pontos seriam incluídos ou excluídos da máscara, levando a regiões ativas inadequadas. No processo de desenvolvimento do trabalho este filtro foi aplicado com um *kernel* de 3 pois foi o que obteve melhores resultados em relação á perda de qualidade de imagem.

Existem diversos métodos de suavização tais como suavização de medias, suavização gaussiana e suavização bilateral, todos estes filtros possuem o mesmo objetivo, remover ruido e melhorar a imagem para facilitar o processamento.

Tendo em conta os nossos objetivos neste ponto do trabalho o filtro de mediana foi o mais adequado para filtrar a máscara, e o filtro gaussiano o mais adequado para pré processar cada frame.

Figura 2 - Imagem original (esquerda) e imagem suavizada com kernel gaussiano (direita)

Operações morfológicas

A seguir da aplicação de *thresholding*, foram realizadas transformações morfológicas. As transformações morfológicas baseiam-se num conjunto de operações nas quais se modificam as estruturas espaciais de um objeto numa imagem, com uso de um elemento estruturante que vai percorrer a imagem aplicando operações bit a bit.

As operações morfológicas mais básicas são a erosão e a dilatação. A erosão remove pixéis das fronteiras de objetos nas imagens, enquanto que a dilatação adiciona pixéis às fronteiras dos objetos nas imagens.

As operações morfológicas realizadas foram as seguintes: fecho com *kernel* (2,2) com cinco iterações e de seguida uma dilatação com um *kernel* (3,3).

O fecho baseia-se numa dilatação seguida de uma erosão, foi efetuado de forma a remover ruido na imagem e para fechar pequenos buracos dentro dos objetos das imagens.

$$B \bullet S = (B \oplus S) \ominus S$$

A dilatação foi utilizada de forma a aumentar os objetos das imagens.

Figura 3 Máscara de background após processo de fecho e dilatação

Deteção de contornos

Apos a remoção do ruido, aplicação de filtros e utilização de operadores morfológicos para melhoramento das diversas imagens, foi realizada a extração de componentes conexos.

Na etapa de extração de componentes conexos, a imagem é dividida em regiões ou objetos, segundo um critério. Estes algoritmos baseiam-se na descontinuidades e similaridades da imagem binária. A descontinuidade consiste no particionamento da imagem em zonas caracterizadas por mudanças bruscas na imagem binária, tendo interesse em descobrir linhas, bordas e pontos separados da imagem.

As bordas na imagem de interesse caracterizam os contornos dos objetos nela presentes, sendo útil para identificar os diferentes objetos na cena.

Para se extrair os contornos recorreu-se ao método do cv2.findContours, esta função tem um parâmetro chamado RETR_TYPE, que é responsável por definir de que maneira vai ser retornado o output. Ao usar *CV_RETR_EXTERNAL* são só tomados em conta os contornos externos, não é tido em conta o interior do objeto, desta forma eliminamos reflexos presentes nas portas dos carros que iriam ter um vetor de movimento contrário ao do veículo.

Classificação de objetos

O próximo passo consiste em atribuir a cada objeto a sua respetiva classe, classificando o objeto e atribuindo um identificador único respetivo.

A primeira etapa deste processo consistiu na validação dos contornos obtidos, sendo que objetos detetados só são tomados em consideração para o processo de classificação se cumprirem requisitos específicos.

A partir do método *cv2.contourArea* é possível obter a área para os diferentes contornos, se a área do contorno for inferior a 480 assume-se que o objeto delimitado por este contorno não tem dimensão suficiente para ser classificado como uma das nossas três classes de objetos, logo é descartado. De forma a

No processo de classificação foram tidos em conta os seguintes parâmetros: rácio entre altura e largura e a área.

Como os humanos têm um rácio largura altura muito menor que um carro e uma bicicleta é possível classificar um humano. Um objeto é classificado como humano se o seu rácio for superior a 0.34, inferior a 0.80 e a área for inferior a 5000, um valor demasiado elevado para ser obtido por um humano.

Os carros têm um rácio largura altura significativamente maior ao dos outros objetos por classificar. Um objeto é classificado como carro, se o seu rácio for superior a 1.10 e a sua área for superior a 1500.

Um objeto é classificado como bicicleta, se o seu rácio for superior a 0.80 e inferior a 1.10, tendo área inferior a 850.

Utilização do histograma de cores para classificar objetos

Um histograma de cores indica a distribuição de cores dentro de uma imagem ou parte desta, no nosso caso apenas a *bounding box* correspondente a um objeto. Por isso apenas parte da imagem foi considerada para a criação dos histogramas de cor, assim para calcularmos um histograma fazíamos um *slice* da imagem original utilizando a biblioteca Numpy e chamamos o método calcHist do openCV.

A forma como iriamos classificar o objeto seria comparando, o histograma do objeto com o histograma de um objeto que sabíamos ser representativo da classe - Isso ou classificaríamos por outros métodos e guardávamos o respetivo histograma para posterior comparação com o original, através da distância entre os histogramas.

A distância de histogramas é também computada pelo openCV, pelo método compareHist, sendo que histogramas idênticos teriam valores iguais ou próximos de 1. Seria esta a métrica utilizada para classificar objetos entre eles.

Uma das possíveis formas de classificar os objetos seria através da sua variação de cores. A ideia seria distinguir entre objetos, nomeadamente pessoas e carros através da variação das cores, um carro teria um tom metálico e apenas 1 ou duas cores dominantes vivas, e uma pessoa teria uma forte componente correspondente ao seu tom de pele. Tal não se demonstrou verdade por alguns motivos nomeadamente:

- O histograma de cores de uma pessoa apresenta sobretudo cores relativas á sua roupa e não ao seu tom de pele. E inclusive a cor do fundo (estrada) teria um peso maior que a própria pessoa

- O histograma de um carro no espaço RGB não reflete o tom metálico sendo necessário converter para outro espaço, nomeadamente YCbCr que diferencia entre cor, e luminância.

Este método não demonstrou resultados satisfatórios pelo que acabou por não ser utilizado para classificar os objetos, mas apenas como desempate entre a arvore de decisão e os vizinhos mais próximos. No entanto mantivemos parte dos métodos responsáveis para identificar alguns problemas relativos á cor.

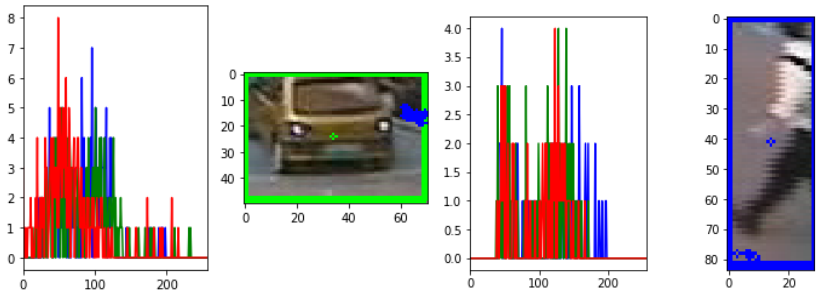


Figura 4 - Histograma de dois objetos (Carro á esquerda, pessoa á direita)

Processo de atribuição de identificador

Foi desenvolvido um algoritmo que permite classificar objetos em movimento, atribuindo-os um identificador único consoante o cálculo da distância euclidiana entre objetos de regiões ativas e objetos de *frames* anteriores.

$$E(x,y) = \sqrt{\sum_{i=0}^{n}(x_i - y_i)^2}$$

Figura 5 Fórmula da Distância Euclidiana

De forma a conseguir guardar objetos de regiões ativas classificadas criou-se uma Classe denominada ClassifiedObject(), esta classe permite guardar informação dum objeto classificado, respetivamente: coordenadas do objeto, tipo do objeto (humano, bicicleta ou carro) e identificador. Os objetos ClassifiedObject() têm um tempo de vida limitado, de modo ao processo de atribuição de identificador estar apenas a referir-se a regiões recentes.

Foi também criada a classe ClassifiedObjects(), esta classe permite armazenar e criar diversas instancias de objetos ClassifiedObject().

De modo a ser possível detetar o movimento de um objeto ao longo do tempo, é necessário comparar a região ativa corrente, com objetos de *frames* anteriores. Utilizou-se a distância euclidiana entre o objeto da região ativa, e objetos das regiões ativas anteriores, consoante o tipo ser igual, sendo obtido o valor mínimo. No caso do valor mínimo da distância euclidiana for inferior a 30 considera-se que o objeto da região corrente é igual ao da região anterior para qual se obteve esta distância. De seguida define-se que o objeto classificado é o mesmo, do que desta região, atualizando as suas coordenadas e atualizando o seu tempo de vida. Caso passar mais que cinco segundos sem o tempo de vida um objeto classificado ser atualizado, o objeto classificado torna-se inativo. Se não houver nenhum objeto ativo em *frames* anteriores é criado um objeto.

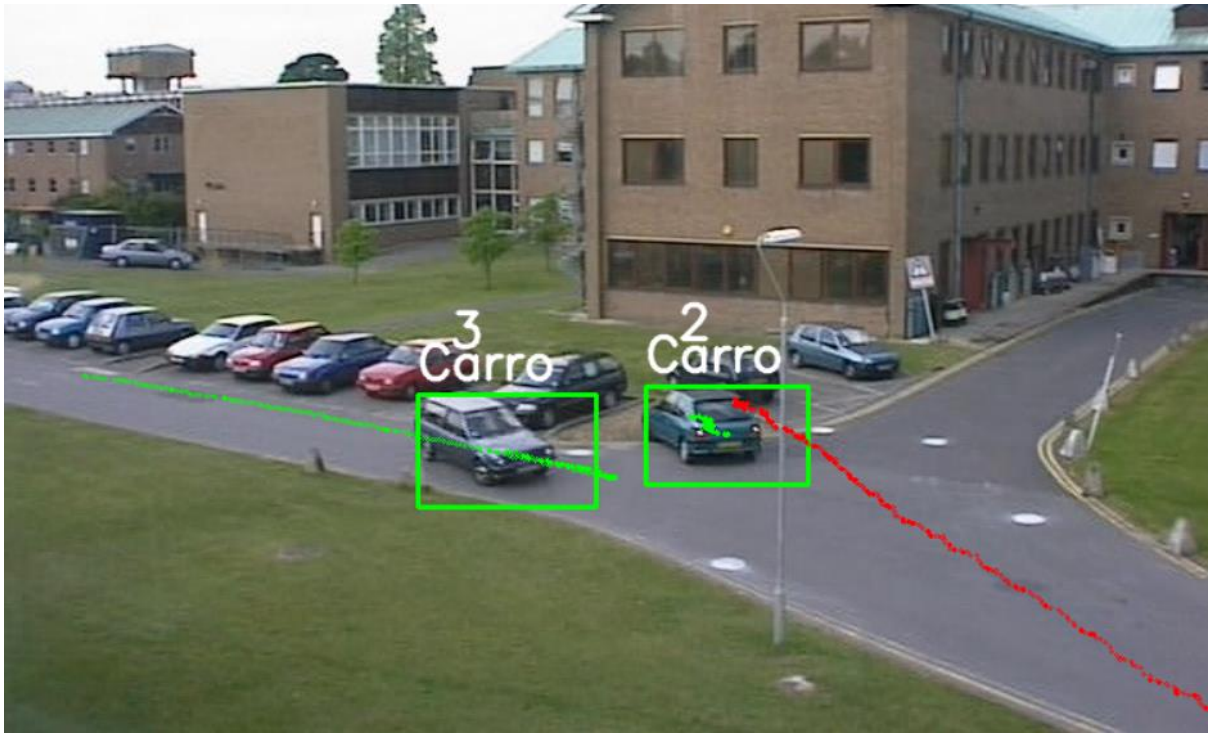
Visualização dos objetos classificados

Para efeitos de visualização, foram criadas *boundingboxes* e etiquetas de texto, para cada região ativa, coloridas de acordo com os resultados da classificação e com o identificador do objeto seguido.

Foi criada uma classe denominada, drawStuff(), com métodos que permitem criar diversas *bounding boxes* e etiquetas de texto numa *frame* especifica. Estes objetos de visualização são criados consoantes as coordenadas dos diversos objetos classificados e o seu tipo. A cor dos objetos de visualização para os carros é verde, para os humanos é vermelho e para as bicicletas é azul.

De modo a capturar o movimento ocorrido nas diversas *frames* foi criado um método que permite visualizar os vetores de movimento ao longo do vídeo. Cada ponto do vetor de movimento é representado com o uso do *cv2.circle*, com a cor respetiva ao tipo de objeto classificado responsável por esse movimento.

As coordenadas dos vetores de movimento e o tipo de objeto que produziu este movimento são guardados numa lista, na forma de um tuplo: (tipoObjetoClassificado, coordenadas).



Conclusões

A partir do trabalho realizado foi possível criar indícios de um sistema vigilância, capaz de detetar e seguir regiões ativas numa sequência de imagens.

A elaboração deste trabalho prático serviu para colocar em prática o conhecimento adquirido na disciplina de Processamento de Imagem e Visão numa situação real, tendo permitido obter conhecimento sobre diversas técnicas e metodologias de processamento de imagem.

Foi feito um estudo e implementação prática de subtração de fundo, deteção de movimento a partir de pixéis ativos, operadores morfológicos, extração de propriedades de regiões e classificação de regiões ativas.

A maior dificuldade ocorreu no problema de eliminar sombras, tentou-se utilizar diferentes parâmetros da função cv2.createBackgroundSubtractorMOG2, e diferentes parâmetros do *threshold* no processo de binarização de modo a resolver este problema.

Bibliografia

Documentação disponibilizada pelo docente na unidade curricular Processamento de Imagem e Visão, Moodle 20/21

https://stackoverflow.com/questions/33266239/differences-between-mog-mog2-and-gmg

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html