

Visão por Computador

10 - Object detection

Diogo Corte, Diogo Silva
 DETI, Universidade de Aveiro
 Aveiro, Portugal
 {diogo.corte, dbtds}@ua.pt

Resumo – Este relatório foi realizado para a unidade curricular de Visão por Computadores com o objetivo de demonstrar a percepção da aula (VC Exercises 10). Sendo que o objectivo é explorar algoritmos que permitam fazer a detecção de objectos, como Template Matching ou Haar Cascades.

NOTA SOBRE AS IMAGENS

Apesar de as imagens terem aparência pequena, dispõem de boa resolução sendo que se for aplicado zoom sobre a imagem é possível ver com detalhe que se pretende.

I. DESCRIÇÃO DOS FICHEIROS

O conteúdo da pasta deste relatório contém os seguintes ficheiros:

1. match_template.cpp, template matching
2. face_crop_from.image.cpp, face detection de uma imagem
3. face_crop_from.video.cpp, face detection de vídeo
4. face_recognizer.cpp, face recognition
5. script.py permite criar um ficheiro CSV que representa a base de dados das imagens para o recognition

II. TRAFFIC SIGNS

O código fonte desenvolvido para os sinais de trânsito teve por base a seguinte fonte: http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html

A. Template usado

O template escolhido (imagem 1) para os testes efetuados foi um sinal de stop que foi desenhado e não obtido de uma imagem real.



Figura 1: Sinal stop usado como template para o matching

Este sinal foi escolhido porque é fácil de obter imagens reais com este símbolo sendo mais fácil de efetuar os testes pretendidos.

B. Template matching sobre a imagem de onde o template foi retirado

Na imagem seguinte 2 é possível verificar o resultado do template matching para a imagem de onde o sinal foi retirado, ou seja, é suposto conseguir encontrar sem problemas.



Figura 2: Imagem original de onde o sinal de trânsito foi recortado

Pode-se verificar que existe um ponto mais negro na imagem do lado direito contém um ponto mais negro que corresponde aos varrimentos onde o template coincidiu mais, foi naquelas coordenadas. Sendo que a representação do sinal mais provável está marcada na imagem original (do lado esquerdo) onde está exatamente o sinal de stop.

C. Detecção de imagens reais

Esta secção é relativa a template matching para imagens reais sendo que é utilizado o mesmo template.

C.1 Imagem com dimensões muito grandes relativamente ao template

Na seguinte image 3 contêm um exemplo de uma situação real.

É possível reparar que na parte de baixo do sinal foi onde o template matching indicou com a area mais provável para aquele template, no entanto, não está correta. Mas ao comparar o tamanho do template com o tamanho do sinal na imagem real é possível verificar que o tamanho do template é muito menor, sendo que o template matching



Figura 3: Sinal da imagem para a deteção demasiado grande comparado com o template

não tem em conta resizes de imagens.

Para este problema existe uma solução muito simples, efetuar vários resizes sobre uma imagem e ver qual a solução mais provável. Sendo que na imagem 4 foi feita a redução da imagem para que o sinal da imagem real ficasse das mesmas dimensões que o template.



Figura 4: Imagem de tamanho ajustado de acordo com o template

E pode-se verificar que é feita a detecção da localização do sinal sem quaisquer problemas.

D. Imagens com a zona de interesse parcialmente coberta

Na imagem 10 é possível verificar que o sinal de trânsito está parcialmente correto. Sendo que a deteção feita pelo template matching apesar de estar nas mesmas dimensões que o sinal não o consegue encontrar na mesma.



Figura 5: Sinal obstruído por uma árvore

E. Conclusão

Template matching funciona bem para cenários em que se pode aplicar directamente o template, como por exemplo, numa fábrica onde se produz uma determinada peça e é preciso identificá-la, mesmo que essa peça varie de tamanho ou esteja rodada basta aplicar rotações ou ajustes sobre o tamanho da imagem.

No entanto se houver problemas em que a imagem está parcialmente coberta ou destruída já se torna mais complicado.

III. FACE DETECTION

O código fonte desenvolvido para os deteção de cara teve por base a seguinte fonte: http://docs.opencv.org/2.4/doc/tutorials/objdetect/cascade_classifier/cascade_classifier.html.

Antes de prosseguir com o resultado do Haar Cascades para efetuar a deteção, convém perceber minimamente o seu funcionamento.

A. Explicação do HaarCascadas

Para o Haar Cascades validar se um dado objeto existe na imagem é feito através de várias fases, mas conhecidas por stages, estas stages são validadas de forma crescente, sendo que primeiro começam com pequenas validações mais genéricas e as ultimas já são mais específicas e detalhadas. Sendo que se uma das stages falha, não é considerado como objeto detetado. Estas validações são pequenas features que são verificadas na área de interesse onde o Haar Cascades se encontra a avaliar [1].

Para melhorar o comportamento do mesmo ainda é feito variar o tamanho da área que é validada pelo HaarCascadas com detectMultiScale.

Também é possível visualizar no video seguinte o comportamento do algoritmo: <https://www.youtube.com/watch?v=nVbaNcRldmw>.

B. Uso do HaarCascadas para deteção

No nosso caso, não se treinou qualquer dataset, sendo que foi utilizado um dataset já treinado para efetuar a deteção de caras, haarcascade_frontalface_alt.xml. No entanto para efetuar o treino é usado um conjunto de imagens positivas e negativas, sendo que as positivas são as que contêm o objeto a ser detetado e as negativas são as que não contêm.

É possível verificar o resultado da deteção das caras na imagem 6.



Figura 6: Deteção de cara

Os resultados são bastante positivos. No entanto por vezes é possível visualizar alguns falsos positivos, principalmente quanto o scaling do detectMultiScale ainda é pequeno, sendo que parece mais suscetível a erros.

IV. FACE RECOGNITION

O código fonte desenvolvido para os reconhecimento de cara teve por base a seguinte fonte: http://docs.opencv.org/2.4/modules/contrib/doc/facerec/tutorial/facerec_video_recognition.html

A. Modelo FisherFaceRecognizer

Para reconhecimento facial o OpenCV tem o FaceRecognizer, um modelo que pode ser treinado com a relação imagem, label.

Os algoritmos disponibilizados pelo OpenCV são Eigenfaces, Fisherfaces e Local Binary Patterns Histograms [2].



Figura 7: Dataset com label 'Diogo Corte'

O dataset de imagens de treino é normalizado para um tamanho fixo (200x200 pixels), requisito para o algoritmos Fisherfaces utilizado.

B. Resultados



Figura 8: Reconhecimento de caras

Os resultados previstos mantêm-se corretos se não existir alteração de cena ou iluminação, mesmo com um dataset muito pequeno de 20 frames.



Figura 9: Reconhecimento de cara errado

Com alterações de cena e de iluminação, obtiveram-se previsões erradas.

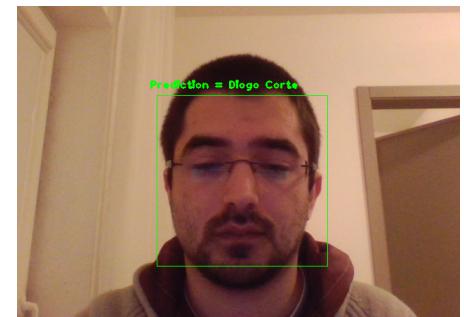


Figura 10: Reconhecimento de cara depois de adicionadas capturas ao dataset

Com mais uma iteração de treino na nova cena as previsões foram corretas.

C. Conclusão

O uso do algoritmo Fisherfaces com Haar Cascade para detecção de caras é uma boa abordagem para reconhecimento de caras classificadas, no entanto o modelo devolve sempre uma das classificações conhecidas, logo não faz de todo distinção entre conhecidos e desconhecidos.

REFERÊNCIAS

- [1] *Visual Recognition and Search* Harshdeep Singh. Acedido a 29 de Novembro.
- [2] *Face Recognition with OpenCV*. Acedido a 29 de Novembro.