Reconhecimento de cartas

Pedro Alves 88861 Ruben Menino 89185

***Resumo* - O presente artigo apresenta uma forma de deteção de cartas de jogar em tempo real através de uma câmera, projeto proposto pela unidade curricular de Computação Visual onde foi utilizado a linguagem python com o uso da biblioteca de openCV.**

***Abstract* - This article presents a way of detecting playing cards in real time through a camera, a project proposed by Visual Computing course where the python language was used and the OpenCV library.**

I. INTRODUÇÃO

O seguinte trabalho foi desenvolvido para a unidade curricular de Computação Visual (4º ano, 1º semestre) do Mestrado Integrado de Computadores e Telemática.

A visão computacional procura modelar e replicar a visão humana usando software e hardware. O processamento de imagens funciona com uma imagem como parâmetro de entrada e um conjunto de valores numéricos como saída (objetos complexos, vetores, matrizes...)

O objetivo da aplicação que vai ser aqui descrita é detectar e classificar cartas de jogar. Para isso é utilizada uma câmera e um baralho de cartas. Existem algumas limitações, como o baralho utilizado e as condições de iluminação a que as cartas estão expostas. Vai ser explicado ao longo do documento o que foi feito para minimizar essas limitações.

II. SETUP

O primeiro passo foi montar um "setup" num sítio com boa iluminação, onde fosse possível fixar uma câmara a uma distância que conseguisse focar as cartas que estavam a ser apresentadas e a apontar para um fundo que não fosse da mesma cor das cartas, numa primeira fase foi utilizado um fundo escuro, que provoca um maior contraste. O que facilita a deteção da carta.

.

Fig. 1 - Primeiro setup utilizado

III. PRÉ- PROCESSAMENTO DA IMAGEM

Inicialmente, antes de avançar para o reconhecimento da carta, foi feito um pré-processamento à imagem.

Como as cartas utilizadas possuem fundo branco e têm tamanho e forma fixa, para o contorno exterior da carta ser detectado fizemos com que os contrastes fossem mais perceptíveis. Para isso começámos por converter as imagens do vídeo para uma escala de cinzentos, e de seguida aplicar um filtro Gausseano para remoção de ruído gaussiano, que pode surgir durante o processo de aquisição de vídeo. Para separar os objetos que desejamos analisar da imagem inicial, utilizamos técnicas de binarização ou limiarização, método de segmentação que permite separar a imagem em regiões de interesse e não interesse através de um valor de corte, threshold. O método de segmentação utilizado é uma operação,

que dada uma imagem em escala cinza, transforma os pixels com intensidade superior ao valor estipulado como threshold em branco e os que têm intensidade inferior a preto (para o nosso caso, o valor mais utilizado é 140), criando assim uma imagem binária.

Em casos de iluminação variável, o valor do threshold tem de ser alterado, de maneira a se conseguir obter a binarização que permite identificar claramente a carta. A melhor forma que encontramos de colmatar as possíveis variações de iluminação, foi criar uma trackbar, onde é possível variar o valor de threshold em tempo real e através da amostragem do resultado gerado, o utilizador consegue estabelecer o valor ideal para o mesmo.

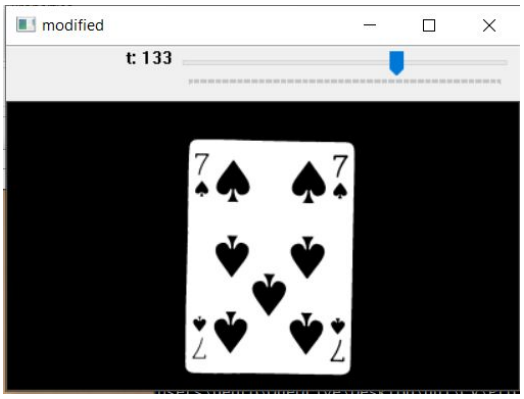

Fig. 2 -Threshold com valor adequado

Fig. 3 - Threshold inadequado, não permite o reconhecimento da carta

.

IV. ENCONTRAR A CARTA

Após executar o que foi mencionado anteriormente, foram alcançadas as condições necessárias para encontrar os contornos das cartas e conseguir manuseá-las para chegar ao pretendido.

Posto isso, para encontrar os contornos das cartas foi utilizada uma função da livraria OpenCV, findContours(), esta une de forma curvilínea todos os pontos contínuos, que possuem a mesma cor ou fronteira. Esses contornos foram desenhados na imagem utilizando a função drawContours(). Ao desenhar os contornos apercebemos-nos que existiam contornos que não nos interessavam para detectar a carta, como por exemplo o contorno dos desenhos da carta. Para fazer a detecção, apenas é necessário o contorno exterior. Tendo em consideração que a forma da carta é regular, e a câmera permanece fixa, limitamos a área a que o contorno é desenhado para o valor correspondente à área da carta. Desse modo, sempre que há um contorno com dimensão aproximada a essa, sabemos que há uma carta a ser captada pela câmera.

Utilizando a função cv2.approxPolyDP(), em cada contorno obtido, faz-se a detecção dos 4 vértices das extremidades da carta. Valores que vão ser necessários para o que vai ser descrito no tópico seguinte.

Fig. 4 - Contornos desenhados, com a contagem das cartas(escrito a azul)

V. PROCESSAMENTO E DATA SET

Uma vez que a carta pode estar disposta de várias maneiras (vertical, horizontal, diagonal,..), ordenaram-se os quatro pontos que representam cada carta . Para isso, foi encontrada a centróide dos pontos, e em seguida calculado o ângulo subentendido da centróide para cada um desses pontos, a partir daí, os pontos são ordenados pelo ângulo realizado, no sentido ``clockwise".

A ordem dos pontos será BOTTOM_RIGHT, BOTTOM_LEFT, TOP_LEFT, e por fim TOP_RIGHT..

Para que a disposição da carta não afetasse os resultados vimos a necessidade de criar uma

transformação em perspetiva de quatro pares que correspondem aos quatro pontos da carta. Para isso utilizamos a função cv2.warpPerspective(). Com isso, independentemente da disposição da carta, ela permanece sempre na vertical. O que nos permitiu fazer o recorte do conjunto do número e símbolo da carta para comparação.

$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = \texttt{map\_matrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$dst(i) = (x'_i, y'_i), src(i) = (x_i, y_i), i = 0, 1, 2, 3$$

Fig. 4 - Matriz 3x3 de uma transformação em perspetiva

Na seguinte imagem podemos ver a carta com uma perspetiva vista de cima.

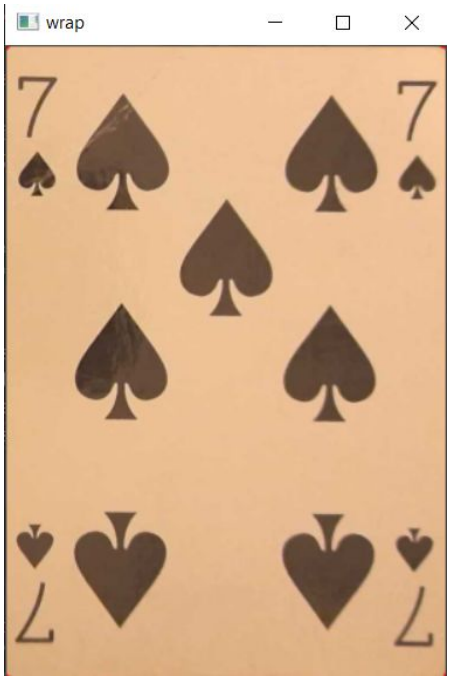

Fig. 5 - Perspetiva da carta

Após ter a carta numa vista de perspetiva é possível começar a análise e recorte do canto da carta.

Definimos uma área que consiga englobar tanto o número como o símbolo..

Após o recorte, e para conseguirmos ter somente o número e símbolo, sem áreas não usadas à volta, fizemos um rectângulo à volta dos contornos do número e do símbolo, e demos "resize" à imagem do conteúdo dentro desse retângulo (Figura 7). Essas áreas foram recortadas, e resultaram na Figura 8. Com isso, pode-se agora proceder à comparação com imagens guardadas de números e símbolos.

Fig. 6 - Recorte do canto da carta

Fig. 7 - Retangulo para proceder ao resize da imagem

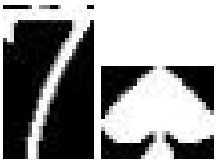

Fig. 8 - Recorte novamente para obter duas variáveis de comparação

De referir que estas imagens vão ter novamente um pré processamento. O processamento vai ser igual ao da carta no geral, porém vai ter uma ligeira diferença no threshold.

Como são imagens mais pequenas e de maior detalhe foi necessário utilizar THRESH_BINARY juntamente com THRESH_OTSU pois iria fazer uma melhor limitação dos pixels brancos da imagem.

Invertemos também as cores dos pixels, para posteriormente ser mais fácil visualizar as diferenças.

Para todas as cartas do baralho, guardamos o recorte já com a imagem processada utilizando a função cv2.imwrite(), para posterior comparação com as cartas que vão ser obtidas no vídeo. Numa pasta 'numbers' guardamos os números de 1 ao 10 e as figuras dama, valete e rei, sendo o 1 o ás do baralho, e numa pasta diferente 'symbols', os naipes, espadas, corações, ouros, paus.

VI. COMPARAÇÃO DA IMAGEM COM O DATASET

Tendo já as imagem guardadas, começa-se o processo de comparação.

A carta apresentada vai ser comparada com todas as imagens do dataset, pelo número e símbolo.

Para realizar essa comparação faz-se uma sobreposição da imagem que está a ser analisada com cada uma das imagens guardadas no dataset.
Como são analisadas duas variáveis diferentes, o número encontrado vai ser sobreposto a todas as imagens dos números armazenados, e o mesmo para os símbolos.

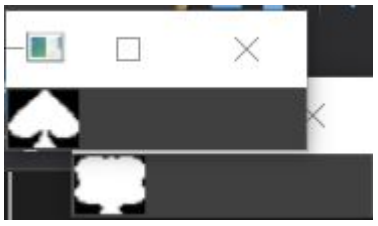

Fig. 8 -Sobreposição espadas/espada e espada/coração

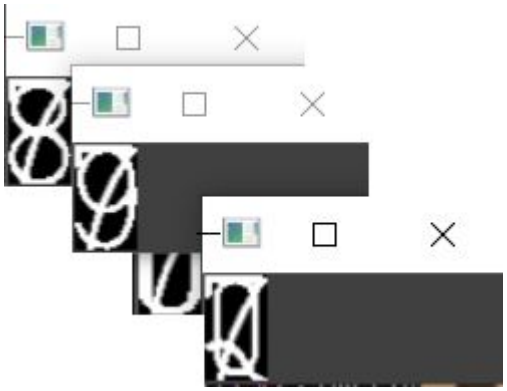

Fig. 10 -Sobreposição 7/8, 7/9, 7,Q

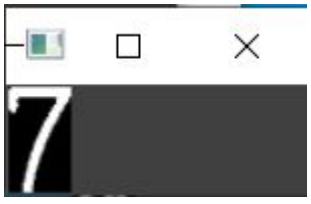

Fig. 11 -Sobreposição 7/7

Ao se efetuar a sobreposição das duas imagens, vai-se contar os números de pixels brancos da imagem sobreposta. De entre todas as sobreposições, a matriz resultante que tiver menor número de pixels brancos é a carta a ser reconhecida.

O número e símbolo da carta são escritos no centro de cada carta. A nossa aplicação deteta tantas cartas quantas couberem no espaço de visualização.

Fig. 12 -Cartas detectadas e classificadas

VII. TESTES E RESULTADOS

Inicialmente, optámos por tentar analisar usando somente o canto da carta recortada, porém não obtivemos os resultados esperados.

Como há bastantes variáveis de combinações de símbolos com números, ao sobrepor toda a informação (número e símbolo) e comparar com a imagem guardada no dataset, os resultados variam bastante.

Foi resolvido ao em vez de analisar a imagem toda, cortar o símbolo e o número e analisar separadamente ambos.

Foram também realizados testes, com diferentes câmaras, ex: câmara do telemóvel, e diferentes bases para as cartas e o resultado foi bastante positivo.

A versão final do projeto, num setup com uma câmara com uma qualidade média e com uma iluminação ambiente obteve uma taxa de acerto de 100%.

VIIII. COMO EXECUTAR O PROGRAMA

1. Testar com câmara

Para testar a aplicação com câmara vai ser necessário ter o baralho o mais parecido possível ao do dataset e executar a seguinte linha no terminal:

python cardRecognize.py video

2. Testar com imagens

Para executar a aplicação sem câmara, para provar o seu funcionamento, são dadas cinco imagens de testes, no diretório principal. Para testar e executar deve-se escrever a seguinte linha no terminal:

python cardRecognize.py testX.jpg

(onde X será entre 1 e 5)

IX. CONTRIBUIÇÃO DOS AUTORES

O trabalho foi realizado por ambos os membros do grupo, resultando numa participação de 50% para cada um

X. CONCLUSÃO

De um modo geral conseguimos chegar ao pretendido, que era o reconhecimento das cartas de um baralho. Com um ambiente favorável à detecção a taxa de acerto é de 100%. As maiores limitações do trabalho foram superadas, conseguindo a partir da trackbar do threshold adaptar o reconhecimento da carta para qualquer iluminação. Relativamente ao baralho, irá funcionar perfeitamente para o utilizado ao longo do projeto. Se o baralho for alterado irá ter algumas falhas pois a comparação estará a ser feita com o dataset gerado do baralho utilizado para o projeto. É possível constatar então , que os principais fatores de variação dos resultados são, a câmara de gravação, o baralho de cartas utilizado e a iluminação presente.

Todos os passos realizados ao longo do projeto, desde a revisão do relatório, e à realização deste projeto, foram uma mais valia. O principal objetivo deste trabalho foi atingido, e ainda foi possível pôr em prática os conhecimentos adquirido ao longo do semestre, tanto nas aulas prática, como nas aulas teóricas lecionadas pelos professores

REFERÊNCIAS

[1] Joaquim Madeira, Paulo Dias
https://elearning.ua.pt/course/view.php?id=1828
[2] Youtube Edje Eletronics
https://www.youtube.com/watch?v=m-QPjO-2IkA
2017
[3] Geaxgx1
https://www.youtube.com/watch?v=pnntrewH0xg&t=43s
[4] OpenCV
https://opencv.org/