ISEL
INSTITUTO SUPERIOR
DE ENGENHARIA DE LISBOA

ÁREA DEPARTAMENTAL DE ENGENHARIA DE ELETRÓNICA E TELECOMUNICAÇÕES E DE COMPUTADORES

LICENCIATURA EM ENGENHARIA INFORMÁTICA E MULTIMÉDIA

PROCESSAMENTO DE IMAGEM E VISÃO

1º Trabalho laboratorial - Contagem e classificação de moedas

André Gomes 38432

Inês Leandro 38434

Indice

Introdução

A evolução exponencial observada tanto nos equipamentos informáticos como nos produtos de captura de imagem, têm levado ao desenvolvimento de sistemas capacitados de observar o mundo real e tirar conclusões acerca do que se está a ver, tal como os nossos olhos recolhem imagens e enviam essas mesmas informações para o cérebro, onde são tiradas conclusões.

Aplicações deste género são desenvolvidas recorrendo a técnicas de inteligência de inteligência artificial aliadas a dados recolhidos de sensores de captura de imagem.

É possível encontrar online várias bibliotecas para desenvolvimento destes sistemas, com funções cada vez mais optimizadas tanto ao nível da gestão dos recursos da máquina, como no próprio processamento das imagens, automatizando a maioria da matemática necessária para estes projectos. *Ex: OpenCv, VXL, libCVD*

Este trabalho prático tem por objectivo o desenvolvimento de algoritmo de visão por computador, capaz de contar automaticamente a quantia em dinheiro (apenas moedas), colocado em cima de uma mesa, a partir de várias fotografias com um fundo homogéneo e todas obtidas a partir da mesma câmara, à mesma distância.

A biblioteca escolhida para este trabalho é o OpenCV (**Open** Source **C**omputer **V**ision), e Python como linguagem para desenvolvimento da aplicação.

A aplicação deve ser capaz de a partir do conjunto de treino utilizado, retirar das imagens todas as informações que precisa, descartando toda a informação desnecessária, para reconhecer e posteriormente classificar as moedas observadas. Pode-se então dividir o processo em três grandes passos: processamento da imagem, reconhecimento das moedas e classificação.

O algoritmo a criar deve ser capaz de, em qualquer imagem criada nas mesmas condições, ignorar objectos diferentes de moedas, pequenas sombras e o contacto que possa existir entre objectos.

1. Desenvolvimento

1.1. Processamento de imagem

O algoritmo começa por fazer a leitura das imagens (função *loadImage*), obtendo os três canais de cor das mesmas (vermelho, verde, azul).

Em seguida as imagens são submetidas a um processo de binarização (função *binarizarImage)*. Para tal foi necessário analisar os histogramas das imagens, e dado que as imagens foram todas adquiridas nas mesmas condições, as diferenças entre todas são ligeiras, e como tal bastou analisar apenas um histograma. Como se pode observar pela **Erro! A origem da referência não foi encontrada.**, dos três canais de cores o que melhor separa o fundo do objeto é o vermelho, e o melhor valor de limiar 105, numa escala de 0 a 255.

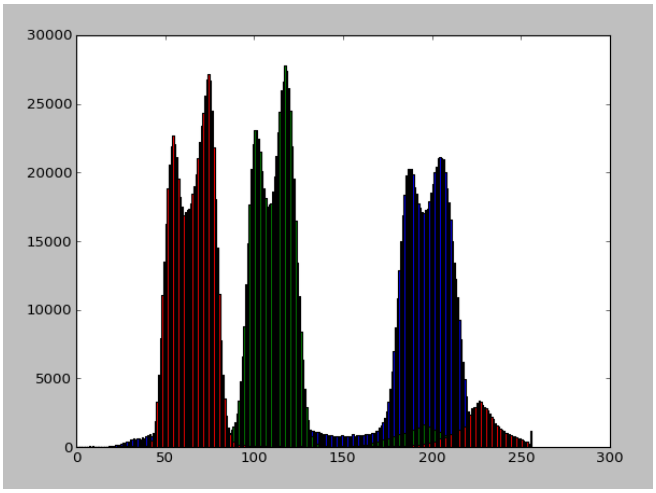


Figura 1 - Histograma da imagem P1000713s

Procedeu-se então à binarização (função *binarizarImage*) da imagem utilizando o canal vermelho (função *redImage*) e o limiar, sendo o resultado uma imagem a preto e branco, Imagem 1.

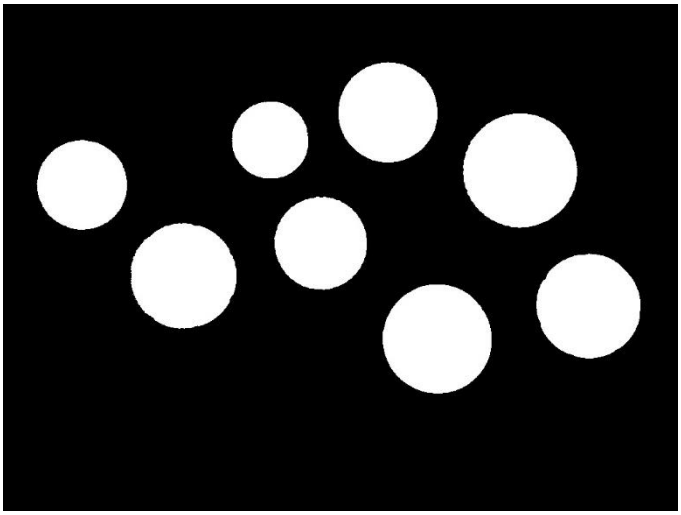


Imagem 1 - Imagem P1000697s binarizada

Contudo, observou-se que apesar da binarização, algumas moedas não se encontravam totalmente brancas, possuíam alguns pixéis pretos, derivados de algum tipo de ruído ou saturação na imagem, Imagem 2.

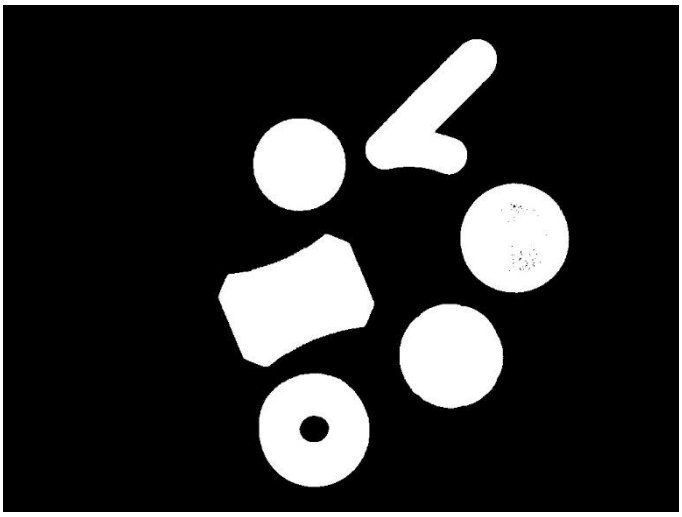


Imagem 2 - Imagem P1000710s com ruído

Para resolver esse problema realizou-se um *close*, dilatação de uma imagem seguida de uma erosão, que é utilizado para retirar essas pequenas falhas (função *melhoramentoImagem*), Imagem 3.

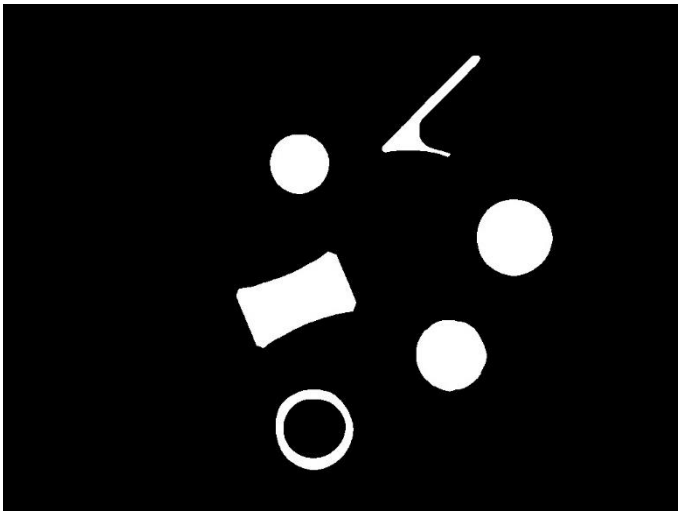


Imagem 3- Imagem P1000710s após dilatação e erosão

Ao retirar os buracos das moedas para deixá-las totalmente brancas, as moedas que se encontravam muito próximas umas das outras acabaram por ficam coladas, ou seja, em vez de ter um contorno para cada moeda, passou-se a ter um único contorno para todas as moedas que se encontravam bastante próximas, Imagem 4.

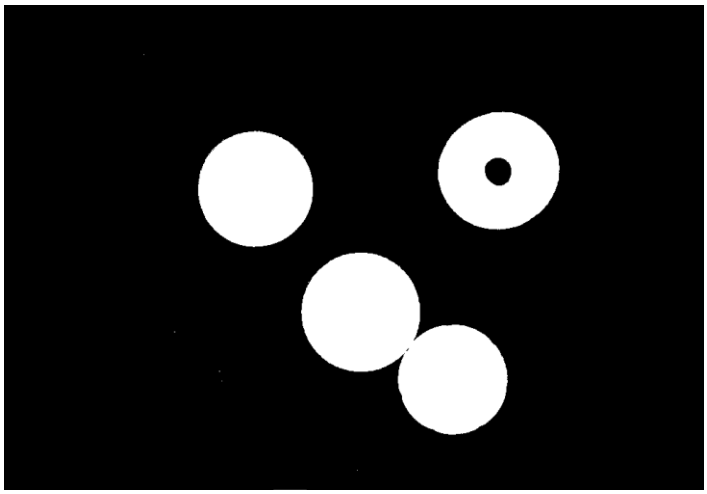


Imagem 4 - Imagem P1000713s após dilatação e erosão

Para resolver este problema, procedeu-se a uma erosão para separá-las, Imagem 5.

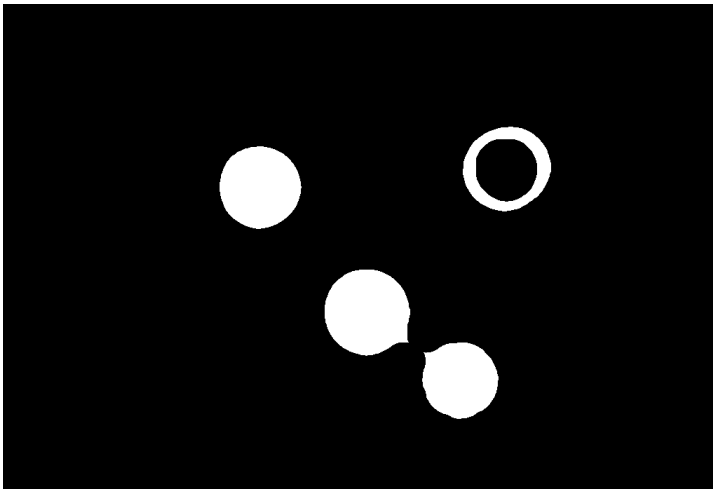
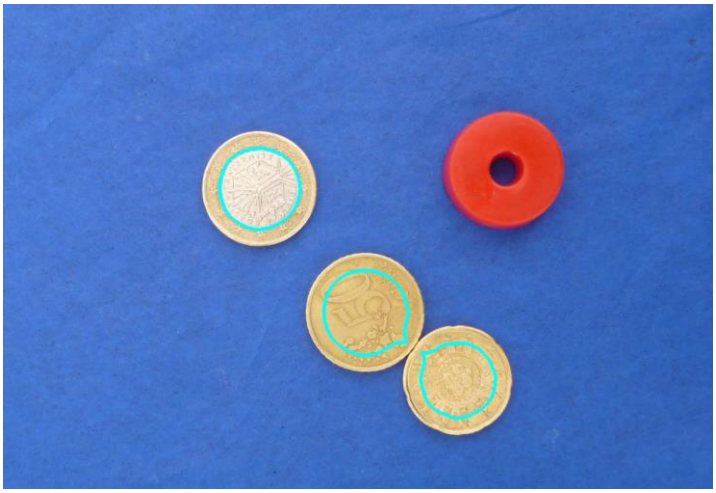


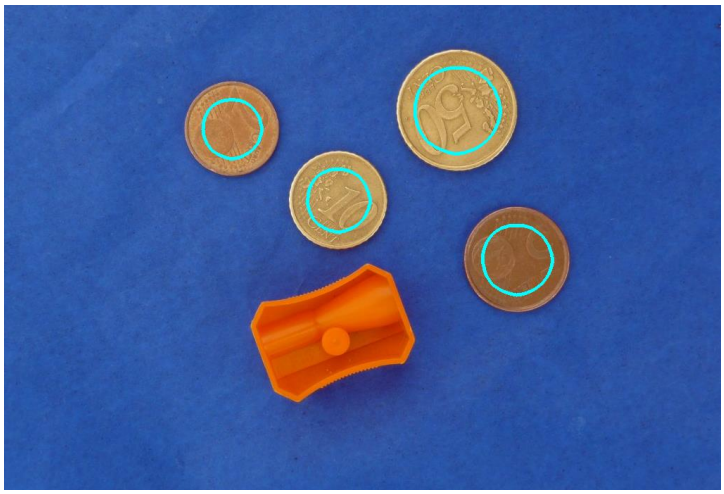
Imagem 5 - Imagem P1000713s após segunda erosão

2.2. Reconhecimento de moedas

A partir de uma imagem já tratada passada como argumento (função *extracCompConexos*), é feita a extracção dos contornos e das respectivas hierarquias entre si, recorrendo à função *findContours* do OpenCV. Em seguida, são percorridos ambos os *arrays* obtidos, procurando e descartando todos os que tiverem contornos internos, ou seja, buracos.



Em seguida, foi necessário retirar todos os objectos que não fossem redondos como moedas. A função *removeNonRound* percorre todos os contornos existentes nesta fase de processamento e a partir da sua área e perímetro calcula a circularidade. Para as moedas, observou-se que os valores de circularidade estavam distribuídos entre 14 e 15, então o algoritmo remove todos os contornos cujo valor de circularidade seja maior que 15 (quanto maior o valor, menos redondo é considerado o objeto).



2.3. Classificação

Por fim, é apenas necessário pegar nos restantes contornos, e assumir que todos eles são moedas, classificando-as de acordo com o seu tamanho. A partir da observação de todas as imagens, foi desenvolvido um dicionário que tem como chave os nomes das moedas, associados a valores de área mínima, máxima, e valor da moeda em cêntimos.

Os contornos são processados pela função *classObjectos*, que comparam o valor da área de cada contorno aos valores do dicionário e assim classificam a moeda com o valor correcto, podendo então fazer o somatório do valor monetário das moedas observadas, e apresenta-lo no ecrã.

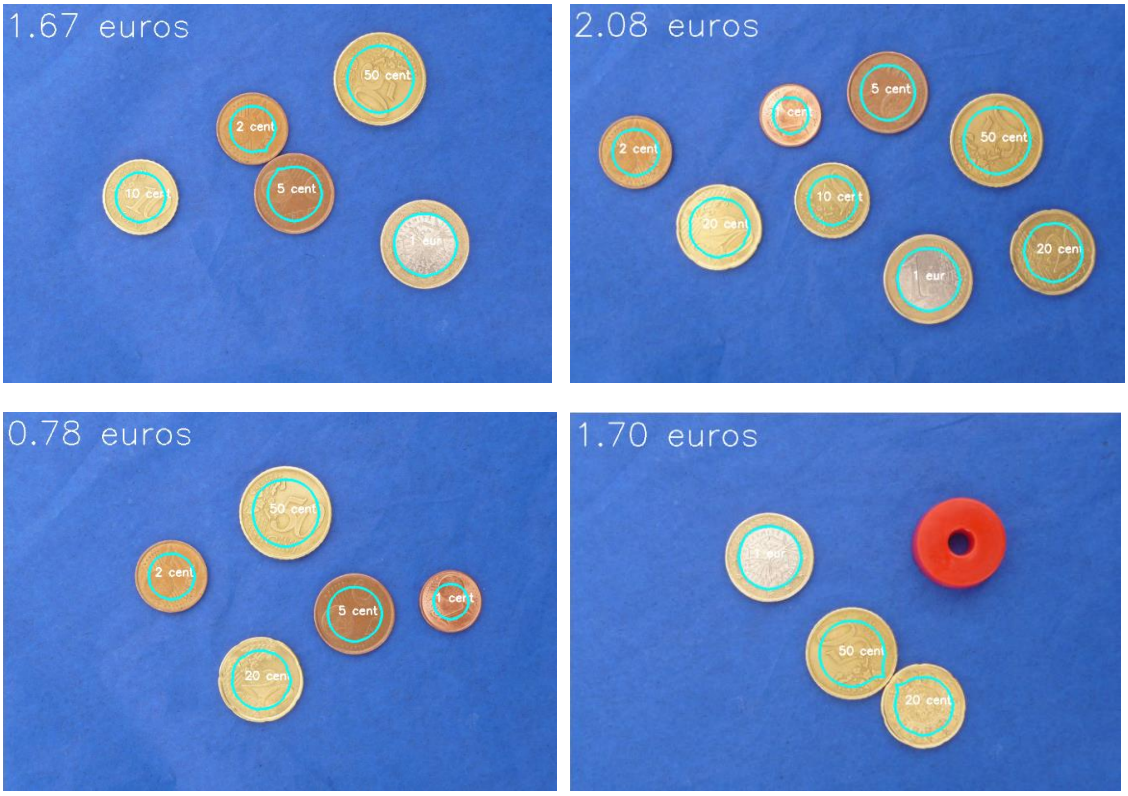
Resultados

Ao correr o *script* de Python, é percorrido um array de imagens, que actualmente contém o conjunto de treino, e que pode ser eventualmente substituído por outro. Por cada imagem, é mostrado:

- Imagem original
- Imagem binarizada
- Melhoramento da imagem binarizada
- Contornos dectados
- Contornos rejeitados
- Contagem de moedas

São ignorados todos os objectos do conjunto de treino que não são moedas, incluindo objectos redondos com dimensões inferiores à moeda mais pequena e maiores que a maior moeda. É provável que se for incluída uma imagem que contenha uma moeda de 2 euros esta seja "confundida" com uma moeda de 50 cêntimos, dado que o conjunto de treino não contém nenhuma, e como tal não foi prevista na criação do classificador.

Os resultados obtidos foram os seguintes:




1.62 euros
2 cent
10 cent
50 cent
1 eur


0.67 euros
2 cent
50 cent
10 cent
5 cent


1.30 euros
10 cent
1 eur
20 cent


1.33 euros
2 cent
10 cent
1 cent
20 cent
1 eur

Conclusão

Com este trabalho concluiu-se que hoje em dia a criação de algoritmos para sistemas de classificação está mais acessível, devido à existência de bibliotecas como o *OpenCV*, que por exemplo possui métodos para a extracção de componentes de imagem.

Prevê-se que o classificador projectado seja robusto o suficiente para processar qualquer imagem que seja tirada em condições semelhantes, e até algumas não previstas como objectos redondos com tamanhos muito diferentes das moedas que estão previstas. No entanto, não prevê objectos semelhantes com texturas ou símbolos diferentes. É provável que, por exemplo, uma réplica de moeda em plástico seja contada como uma moeda, se também for semelhante no tamanho.

No que toca a imagens que contenham apenas porções de uma moeda poderão existir erros, se se tratar de uma parte grande de uma moeda igualmente grande. A moeda parcial que aparece numa das imagens de treino é ignorada devido à sua área ser menor do que a de uma moeda de 1 cêntimo.