Projeto Object Detection

Universidade de Aveiro

João Gameiro , Pedro Pereira, Marco Ramos,
Tiago Pedrosa

VERSAO

Projeto Object Detection

LABI - DETI

Universidade de Aveiro

João Gameiro , Pedro Pereira, Marco Ramos, Tiago Pedrosa

16-05-2019

Resumo

Este documento serve como relatório para um projeto realizado no âmbito da disciplina de Laboratórios de Informática. O programa construído tem como objetivo tornar possível a visualização de uma biblioteca de imagens. Esta mesma biblioteca é utilizada por um programa que é capaz da procura e identificação de objetos em imagens.

Ao longo do relatório, é analisado e explicado o código do programa e do website que o acompanha, expondo todas as funções e características do projecto. Após isto, é efectuada uma demonstração do funcionamento do programa e do modo correto de interacção com a interface do mesmo e o relatório é concluído com algumas reflexões sobre o projeto e o contributo que este teve para o desenvolvimento das nossas capacidades.

Conteúdo

Capítulo 1

Introdução

O programa desenvolvido tem como objectivo permitir a visualização de uma biblioteca de imagens. Acompanhado de outra ferramenta, torna-se possível a procura e identificação de objetos das imagens armazenadas nessa mesma biblioteca.

O relatório está dividido em 5 capítulos. Depois deste Capítulo 1 (Introdução), encontra-se o Capítulo 2 (Desenvolvimento) onde é apresentado e explicado o código que foi desenvolvido. No Capítulo 3 (Resultados) é demonstrado o funcionamento do programa e no Capítulo 4 apresentam-se as conclusões do projeto. Para acabar, é comentada a divisão do trabalho entre os elementos do grupo no Capítulo 5.

Capítulo 2

Desenvolvimento

2.1 Python

2.1.1 Módulos

Os módulos importados foram os seguintes:

- Image : usado na manipulação e obtenção de informação de imagens;
- Sqlite3 : usado para operações relacionadas com a base de dados (incluindo criação e métodos);
- cherrypy : usado para o desenvolvimento da aplicação;
- json : funções de retorno de informação e para obtenção da mesma;
- requests : utilizado para a obtenção de informação do processador fornecido pelos docentes;
- hashlib : sintetização dos nomes das imagens;
- os.path : obtenção de informações sobre directórios.

O programa começa com a definição de um dicionário que contribui para a configuração da aplicação.

2.1.2 Base de Dados

Como podemos verificar na Figura 2.1 a criação da base de dados é feita sob a condição de que se não existir já uma é criada. São sintetizadas duas tabelas uma (Imagem) para guardar informações sobre a imagem original (id e nome original da mesma) e outra (Objecto) para guardar informações como: nome da imagem com o objecto extraído, nome do objecto, a cor do objecto, a confiança usada e o id da imagem original.

Posteriormente à criação da base são definidos vários métodos com a sua função descrita na Tabela 2.1

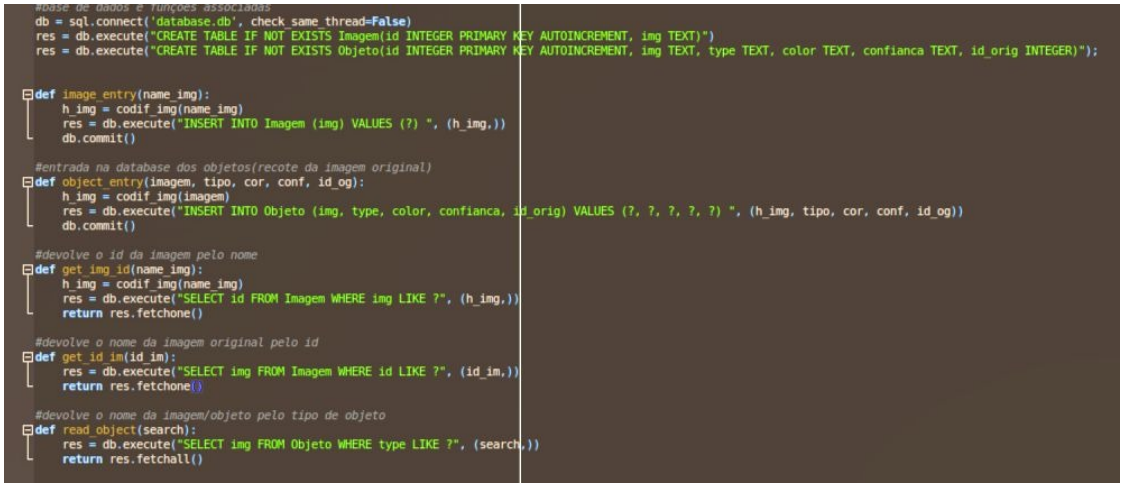

Figura 2.1: Criação da Base de Dados e alguns métodos

Tabela 2.1: Métodos da Base de dados

Métodos	Função
`imag_entry`	Inserir o nome da imagem na tabela Imagem
`objetc_entry`	Inserir dados em todas as entradas da tabela Objeto
`get_im_id`	Obter o id de uma imagem inserindo-se o nome da mesma
`get_id_im`	Oposto da anterior (obter nome inserindo-se o id)
`read_object`	Devolve o nome da imagem pequena com o objeto extraído inserindo-se o objeto
`read_obj_col`	Devolve o nome da imagem pequena tendo em conta o tipo de objeto e cor inseridos
`return_obj`	Devolve todos os objetos

2.1.3 Funções Adicionais

`codif_im`

Esta função serve para sintetizar um nome "encriptado"de uma imagem inserida como argumento. É neste caso que é usado o módulo hashlib.

`belongs_box` e `object_extraction`

A função `object_extraction` depende da primeira na medida em que é nestas duas que é realizado o processo de criação de uma nova imagem com o objeto extraído. A primeira verifica se um dado pixel pertence a uma certa área definida na informação que é fornecida pelo processador do docente. A segunda função analisa todos os píxeis de uma imagem e se estes pertencerem à área definida (usando a função anterior) então vão ser guardados numa nova imagem, caso contrário são descartados.

`color_detetion`

Esta última função serve para detectar a cor predominante numa imagem. Analisando a imagem usando o modo rgb, verifica-se a intensidade de cada um dos canais e dependendo do valor obtido irá dar retorno à cor predominante. Devido à dificuldade em estabelecer intervalos específicos para cada cor esta ferramenta não é totalmente confiável podendo devolver cores diferentes do esperado.

2.1.4 Interligação das Páginas e Listas

Posteriormente passámos à interligação das páginas que constituíam a interface da aplicação. No entanto na Figura 2.2 aparece a definição de um conjunto de listas a anteceder a interligação das páginas. Essas listas serviram para auxiliar no armazenamento de informação e em comentário pode ser encontrada a definição seu conteúdo.

```
lstimn = []             #nomes das imgens originais
lstobj = []             #nomes dos objetos extraídos
lstbox = []             #coordenadas onde estão contidos os objetos de cada imagem
lstcfn = []             #confiança
lstcol = []             #cores dominantes
lstimo = []             #lista com o nome das imagens com os objetos extraídos
lstcod = []             #lista com os nomes das imagens originais mas sintetizados
cnt1    = 0


class Root(object):
    #interligação de todas as 6 páginas
    @cherrypy.expose
    def index(self):
        return open('html/main.html', 'r')

    @cherrypy.expose
    def dynamic1(self):
        return open('html/page.html', 'r')

    @cherrypy.expose
    def dynamic2(self):
        return open('html/page2.html', 'r')
```

Figura 2.2: Interligação das Páginas e Listas

2.1.5 Função *list*

Esta representa a função principal da aplicação sendo que é através da mesma que é recebida e tratada a informação para o sistema.

Através do expose da função *put* é enviada uma imagem para o sistema. Na função *list* a informação proveniente do processador é armazenada nas listas e na base de dados e posteriormente exposta ao cliente em diferentes locais da aplicação.

Na Figura 2.3 podemos visualizar a inserção da informação proveniente do processamento da imagem na base de dados (usando métodos já anteriormente descritos) e em listas para auxiliar no tratamento de informação e em revelação de erros e bugs.

```
#adição à base de dados das imagens com objetos extraídos e da cor detetada
if len(s):
    image_entry(names)

    for i in range(0, len(s)):
        name_obj = object_extraction(names, s[i]['box']['x'], s[i]['box']['y'], s[i]['box']['x1'], s[i]['box']['y1'])
        lstimo.append(name_obj)
        lstcol.append(color_detetion(name_obj))
        object_entry(name_obj, s[i]['class'],  color_detetion(name_obj), str(s[i]['confidence']), get_img_id(str(names))[0])
```

Figura 2.3: Inserção de informação na Base de Dados

Na Figura 2.4 já podemos encontrar a devolução da informação obtida pelo processamento da imagem em forma de JSON ao cliente.Toda esta informação é apresentada ao carregar nos vários botões espalhados pela aplicação.

Um dos casos de devolução é o nome dos objectos detectados até ao momento.

Os outros dois casos dependem de uma pesquisa efectuada pelo cliente, tendo num deles em conta apenas o tipo de objecto de detectado e noutro o tipo e a cor. Ou seja o cliente pesquisa pelo tipo de objecto e a aplicação devolve o nome das imagens (encriptado) com objectos já extraídos. O outro caso processa-se da mesma forma mas a pesquisa envolve também a cor.

```
        #Devolve os objetos detetados
        jsono = '['
        for obj in return_obj():
            jsono = jsono+'{"Objeto": "'+str(obj)+'"},'
        jsono = jsono[0:(len(jsono)-1)] + ']'
        jsono0 = json.loads(jsono)
        cherrypy.response.headers["Content-Type"] = "application/json"
        return json.dumps(jsono0).encode('utf-8')


    #Devolve um json de acordo com o objeto especificado
    elif(name!=None):
        try:
            jsono = '['
            for obj in read_object(name):
                jsono = jsono+'{"image": "'+str(obj)+'"},'
            jsono = jsono[0:(len(jsono)-1)] + ']'
            jsono0 = json.loads(jsono)
            cherrypy.response.headers["Content-Type"] = "application/json"
            return json.dumps(jsono0).encode('utf-8')
        except:
            return "ERRO: Nenhum objeto Encontrado"


    #Devolve um json de acordo com o objeto e com a cor
    elif(nome!=None and color!=None):
        try:
            jsono = '['
            for obj in read_obj_col(nome, color):
                jsono = jsono+'{"image": "'+str(obj)+'"},'
            jsono = jsono[0:(len(jsono)-1)] + ']'
            jsono0 = json.loads(jsono)
            cherrypy.response.headers["Content-Type"] = "application/json"
            return json.dumps(jsono0).encode('utf-8')
        except:
            return "ERRO: Nenhum objeto encontrado"
```

Figura 2.4: Devolução de informação sob a forma JSON

2.1.6 Configuração e porta TCP

Finalizando este capítulo temos na Figura 2.5 uma configuração da aplicação e da porta tcp usada.

```
cherrypy.server.socket_port = 10006
cherrypy.server.socket_host = "0.0.0.0"
cherrypy.tree.mount(Root(), "/", conf)
cherrypy.engine.start()
cherrypy.engine.block()
```

Figura 2.5: Configuração e ligação ao servidor

2.2 HTML, CSS e JAVASCRIPT

Na construção do website foi implementado uma barra de navegação no topo da página que permite acesso ao resto do conteúdo. Para além disto, foram costumizados o fundo e o tipo de letra e adicionado um rodapé para identificar os autores do trabalho. Todo o conteúdo utilizado na construção do website está disponibilizado nos urls seguintes.

- Tipo de letra : http://fonts.googleapis.com/css?family=Ubuntu:bold e http://fonts.googleapis.com/css?family=Vollkorn
- Fundo do site : https://www.toptal.com/designers/subtlepatterns/

Concluindo este conteúdo não nos pertence apenas o usamos no desenvolvimento do estilo do nosso site e declaramos que o mesmo não é da nossa autoria.

O javacript usado serve para a inserção e apresentação de uma imagem na página html quando é inserida no sistema.

Capítulo 3

Resultados

Na Figura 3.1 podemos ver uma apresentação da página inicial da aplicação web desenvolvida, onde se pode observar a barra de tarefas que permite ligação entre páginas e um rodapé com os nomes dos autores.

3.1 Aspecto Gráfico

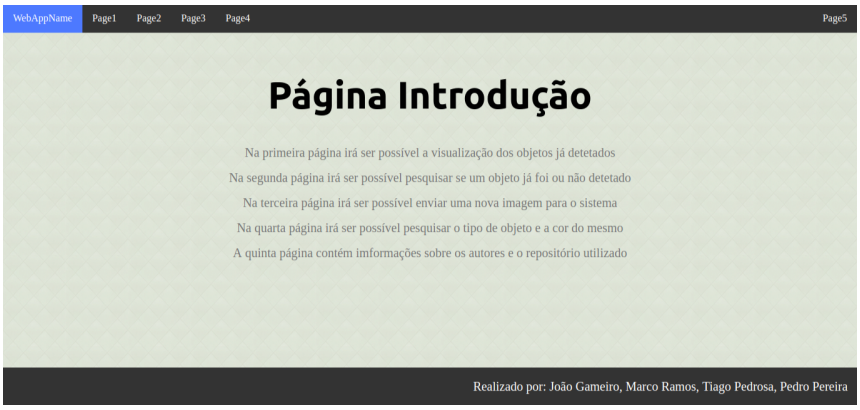

Figura 3.1: Aspeto Gráfico da Aplicação e Teste Funcional

Se seleccionarmos a página 3 vamos obter à interface gráfica visível na Figura 3.2. A partir desta página é possível enviar uma imagem nova, sendo para esse efeito apenas necessário carregar no botão "Choose File". Após isso a imagem será apresentada e seguidamente ao processamento da mesma vamos poder observar um JSON que identifica o objecto detectado. Na Figura 3.2 pode-se visualizar um exemplo que é uma imagem contendo um carro amarelo e o JSON devolvido pode ser visto na figura Figura 3.3. Este JSON é nos devolvido

numa nova página em branca com o único objectivo de devolver este pedaço de informação.

Figura 3.2: Aspeto Gráfico da Página de envio de Imagem

[{"Objeto": "('car',)"}]

Figura 3.3: JSON Devolvido

Se formos ao directório da aplicação vamos poder comprovar que se encontra lá uma nova imagem criada a partir da anterior mas apenas com o carro (o objecto extraído, Figura 3.4).

Após verificar, testando as outras páginas, passando à pagina 1, vemos um botão que diz JSON. Ao carregarmos nele somos levados para uma nova página branca que devolve o objecto detectado, o nome da imagem original sintetizado, tal como o nome da nova imagem e a confiança usada (Figura 3.5).

Passando agora à página 2 em que se encontra uma barra de pesquisa. Insere-se o nome do objecto (neste caso "car") e somos direccionados para uma página em branca que possui um JSON com o nome da nova imagem sintetizado. Figura 3.6.

Novamente mudando de página mas desta vez para a 4 encontramos duas barras de pesquisa, desta vez uma para o objecto e outra para a cor. Se inserirmos em cada barra "car"e "red"respectivamente vamos ser direccionados para outra página em branco que terá o mesmo conteúdo que a anterior visto que o sistema apenas tem uma imagem (um JSON com o nome da nova imagem

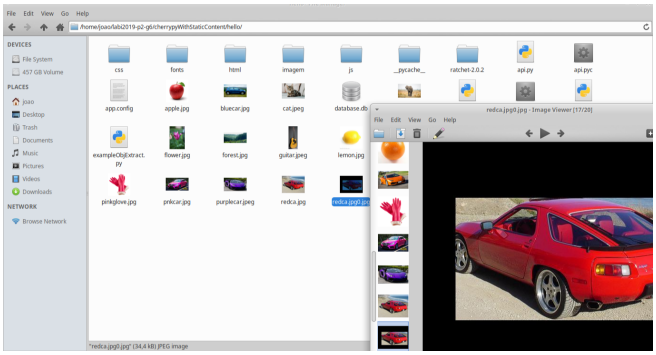

Figura 3.4: Nova imagem criada apenas com o objeto


```
[{"Objeto": "car", "original": "35e842b5f147f19bbba7d2d07047712b", "imagem": "a1a89769ddf1a1c0d375eea843de88fe", "confianca": "0.878"}]
```


Figura 3.5: JSON com o objeto, imagens original e nova e confiança

sintetizada), Figura 3.6). Ou seja esta página serve para pesquisar das imagens obtidas as que contém o objecto e cor designados pelo cliente.


```
[{"image": "('a1a89769ddf1a1c0d375eea843de88fe',)"}]
```


Figura 3.6: Nome sintetizado de uma nova imagem

Testando agora o detetor de cor. Enviando uma nova imagem, mas desta vez com um carro amarelo,verificamos se o sistema reconheceu ou não o objeto, e em caso afirmativo passamos para a página 4. Pesquisamos "car"e "yellow"(Figura 3.7) e verificamos o resultado obtido.

Como podemos comprovar a o nome sintetizado na Figura 3.8 é diferente do sintetizado na Figura 3.6, ou seja a aplicação devolveu a figura com o carro a amarelo e não a figura com o carro vermelho.

3.2 Objetivos não cumpridos

Listar objetos na página destinada a isso e apresentação de imagens nas páginas 2 e 4 e uma das funções que devolve JSON possui erros.

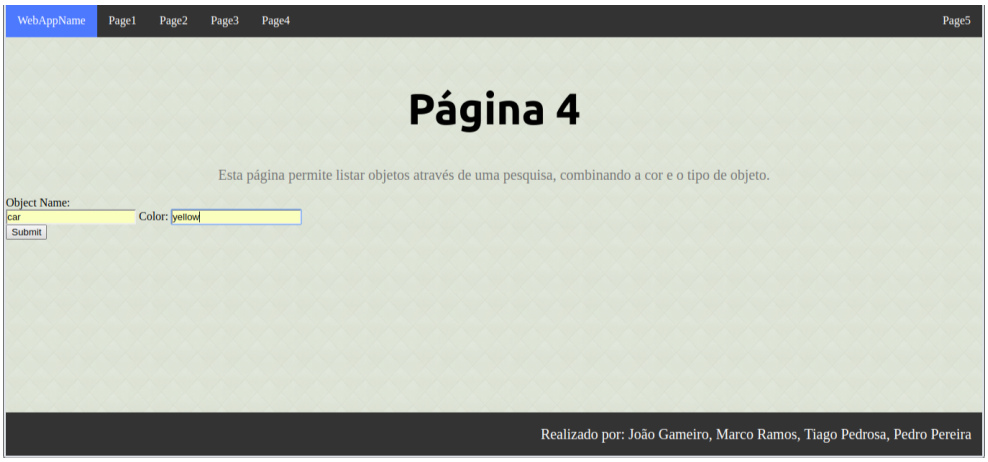

Figura 3.7: Interface da Página 4

Figura 3.8: Resultado da Pesquisa yellow car

Capítulo 4

Conclusão

Não foi possível implementar todas as funcionalidades exigidas no enunciado, no entanto achamos que este projecto contribui bastante para a valorização do nosso percurso académico, tendo em conta que adquirimos competências nas áreas do desenvolvimento de aplicações web, manipulação de imagens e bases de dados. Reconhecemos também que contribuiu positivamente para a nossa capacidade de trabalhar em equipa, discussão, e aceitação de ideias provenientes de outros.

Concluindo, sabemos que não atingimos todos os objectivos esperados neste projecto, mas concluímos que foi uma experiência que enriqueceu o nosso percurso académico, e reconhecemos também que trabalhámos arduamente para o que apresentamos aqui.

Capítulo 5

Divisão do trabalho entre os quatro elementos do grupo

A divisão do trabalho foi feita da seguinte forma:

João Gameiro (JG) e Marco Ramos (MR) desenvolveram em conjunto o python e a base de dados presente neste projecto, discutindo, planeando e estruturando o código juntos.

Pedro Pereira (PP) e Tiago Pedrosa (TP) assumiram responsabilidade pelo design e aspecto gráfico da aplicação, tendo também ambos discutido e planeado a sua parte em conjunto.

É importante referir que apesar desta divisão todo o projecto foi discutido, planeado e estruturado pelos 4 membros e tal como descrito na aplicação concluímos que todos participaram igualmente neste projecto logo as percentagens representativas do trabalho de cada autor são:

- JG - 25%
- MR - 25%
- PP - 25%
- TP - 25%

O relatório foi escrito por PP e por JG. PP escreveu o resumo, introdução e o capítulo sobre HTML e CSS. O conteúdo restante foi escrito por JG.

Foi usado um repositório na plataforma Code UA para o qual todos os autores contribuiu com commits frequentes. O link para esse repositório encontra-se na página 5 da aplicação desenvolvida.

Acrónimos

JG João Gameiro

MR Marco Ramos

PP Pedro Pereira

TP Tiago Pedrosa