Reconhecimento de Imagens

Universidade de Aveiro

Diogo Amaral,Guilherme Pereira,Luís Costa

1.0

Reconhecimento de Imagens

Departamento de Electrónica, Telecomunicações e Informática

Universidade de Aveiro

Diogo Amaral,Guilherme Pereira,Luís Costa
(93228) diogomra7@ua.pt, (93134) guilherme.pereira@ua.pt,(92996) joselcosta@ua.pt

15 de junho de 2019

Resumo

Hoje em dia, é um aspeto bastante óbvio a utilização de novas tecnologia em todas as atividades, e com isto, muitas áreas científicas têm aumentado em termos de informação e eficácia no seu determinado objetivo e função.

Com isto surge a ciência referente ao reconhecimento e classificação de imagens. O nosso projeto tem como objetivo criar um sistema em que seja possível dar *upload* a imagens, pesquisar objetos, mostrar a lista de objetos(por imagem ou por nome) da nossa base de dados.

Para a resolução deste projeto foi utilizado algumas tecnologias já prefeitas, pois alguns conceitos só vão ser abordados nas futuras unidades curriculares. Para além da utilização destas funcionalidades foi desenvolvida uma*Interface Web* que tem como finalidade ser a ligação entre todo o sistema. Foi também elaborado um ficheiro *Python* que funciona como o "cérebro"do sistema e apresenta diversos métodos.

Estes ficheiros e funcionalidades são melhores descritos no decorrer do relatório.

Conteúdo

1 Introdução **1**

2 Objetivos **2**
2.1 Interface Web 2
2.1.1 Listagem dos Objetos(por nome) 2
2.1.2 Listagem dos Objetos(por imagem) 3
2.1.3 Upload de imagens 4
2.1.4 Pesquisa de imagens 4
2.2 Aplicação Web 5
2.2.1 /list?type=names: 5
2.2.2 /list?type=detected: 5
2.2.3 /list?type=detected&name=NAME: 6
2.2.4 /list?type=detected&name=NAME&color=COLOR: . . . 6
2.2.5 /put: 6
2.2.6 /get?id=IDENTIFIER: 6
2.3 Persistência 7
2.4 Processador de imagens 7

3 Bónus **8**

4 Testes efetuados **9**

5 Resultados **11**

6 Conclusões **12**

Capítulo 1

Introdução

Este relatório está dividido em 5 principais partes. A primeira é relativa a este Capítulo 1, a introdução, fora esta componente existe o Capítulo 2 em que é descrita todo o projeto detalhadamente, desde as páginas HyperText Markup Language (HTML), Cascading Style Sheets (CSS), JavaScript (JS) até ao ficheiro *Python*.

No Capítulo 3 é apresentada a nossa ideia, para assim obter o bónus no nosso projeto. No Capítulo 4 é descrito alguns testes feitos para assim verificar a correção das funções. No penúltimo capítulo, o Capítulo 5 são analisados alguns resultados obtidos.

Para finalizar no Capítulo 6 é exposta uma pequena conclusão acerca deste trabalho e posteriormente as contribuições dos autores.

Capítulo 2

Objetivos

2.1 Interface Web

Tal como foi explicitado anteriormente, a *Interface Web* tem a função de interligar todas as funções do sistema, sendo esta dividida em 5 páginas(descritas separadamente de seguida).

Para a realização destas páginas foi utilizado HTML para criar a estrutura das páginas, CSS para que o aspeto das páginas ficasse mais apelativo e dinâmico para o utilizador e JS para conciliação com o ficheiro em *Python*.

2.1.1 Listagem dos Objetos(por nome)

Uma das páginas mostra a listagem dos objetos guardados na base de dados, sendos estes já detetados ou imagens introduzidas no sistema.

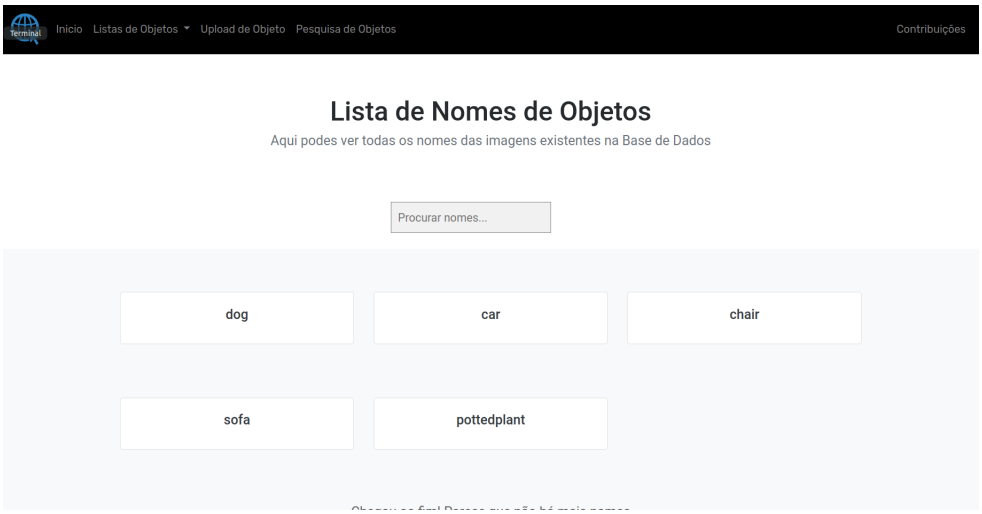

Figura 2.1: Demonstração do site para listar objetos por nome

2.1.2 Listagem dos Objetos(por imagem)

Esta página mostra a listagem dos objetos guardados na base de dados, mas neste caso, lista por imagem e não pelo nome(como o anterior).

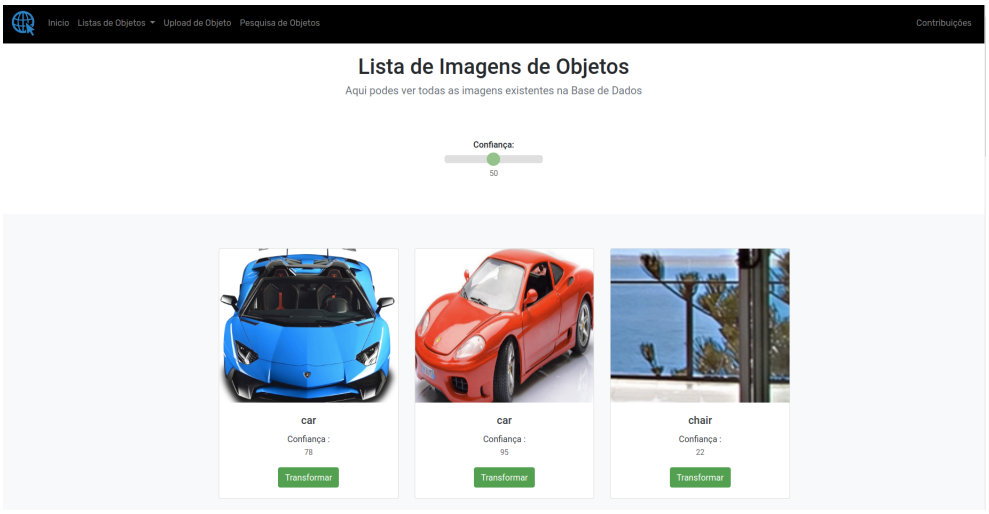

Figura 2.2: Demonstração do site para listar objetos por imagem

2.1.3 Upload de imagens

Esta página é bastante importante, pois oferece a possibilidade de o utilizador dar *upload* a uma imagem para a base de dados do sistema.

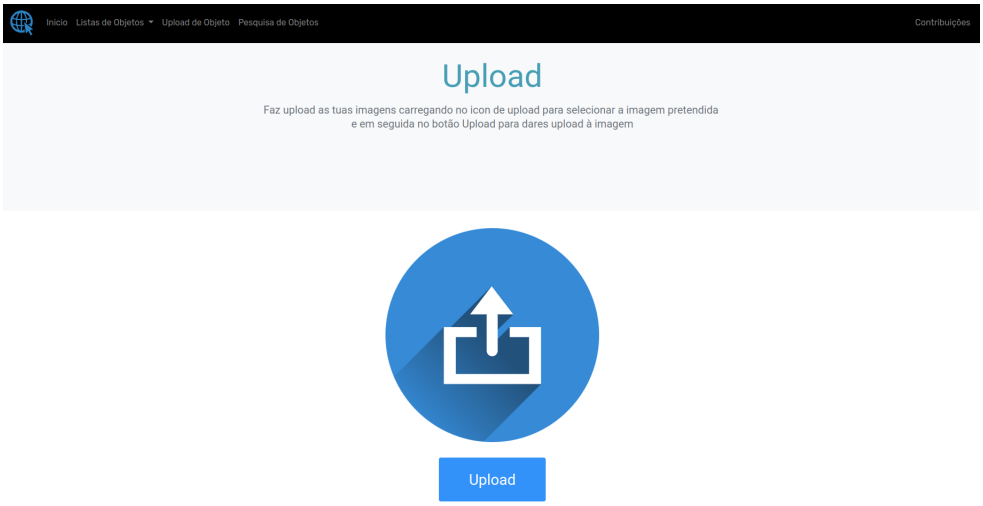

Figura 2.3: Upload de uma imagem

2.1.4 Pesquisa de imagens

Esta penúltima página permite ao utilizador a pesquisa de um objeto de acordo com a cor e o seu tipo. Por exemplo, "red car", o sistema vai procurar imagens com grandes quantidades da cor vermelha, e do tipo descrito e vai mostra-las.

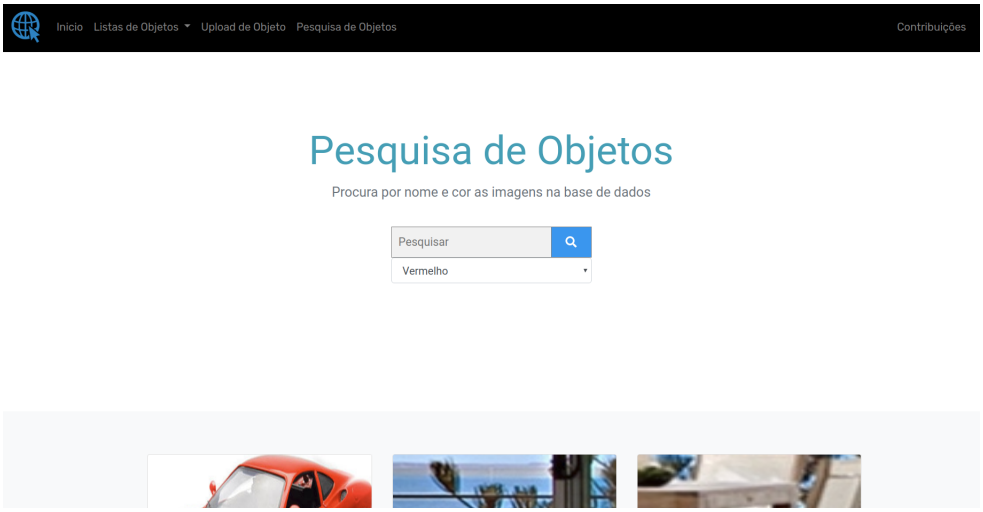

Figura 2.4: Upload de uma imagem

2.2 Aplicação Web

2.2.1 /list?type=names:

Esta função devolve um JavaScript Object Notation (JSON) contendo todos os objetos que forram detetados pelo sistema.

Listing 2.1: getAllNames

```
1  def getAllNames():
2      result = db.execute("Select Nome From Base WHERE IsObject = 'true' ")
3      rows = [rows[0] for rows in result.fetchall()]
4      data = {}
5      rows = list(dict.fromkeys(rows))
6      data['Nomes'] = rows
7      return json.dumps(data, indent=4, sort_keys=True)
```

2.2.2 /list?type=detected:

Esta função retorna uma lista com pequenas imagens recortadas com objetos extraídos.

Listing 2.2: getDetectedObjects

```
1  def getDetectedObjects():
2    names = db.execute("Select Nome From Base WHERE IsObject = 'true'")
3    allNamesRows = [rows[0] for rows in names.fetchall()]
4    singleNamerows = list(dict.fromkeys(allNamesRows))
5    data = {}
6    for singleNames in singleNamerows:
7      data[singleNames] = []
8      ids = db.execute("Select ObjId From Base WHERE Nome = ?", (singleNames,))
9      for id in ids:
10       imgHash = db.execute("Select ImgId From Base WHERE ObjId = ?", (id))
11       acc = db.execute("Select Acc From Base WHERE ObjId = ?", (id))
12       objHash = db.execute("Select ObjId From Base WHERE ObjId = ?", (id))
13       idDic = {}
14       idDic["image"] = [rows[0] for rows in objHash.fetchall()][0]
15       idDic["confidence"] = [rows[0] for rows in acc.fetchall()][0]
16       idDic["original"] = [row[0] for row in imgHash.fetchall()][0]
17       data[singleNames].append(idDic)
18 return json.dumps(data, indent=4, sort_keys=True)
```

2.2.3 /list?type=detected&name=NAME:

Esta função retorna uma lista com pequenas imagens recortadas com objetos específicos. Este método é igual ao getDetectedObjects mas com a diferença de confirmar o nome do objeto.

Listing 2.3: getDetectedObjectsByName

```
1 def getDetectedObjectsByName(name):
2 (...)
3 if singleNames == name :
```

2.2.4 /list?type=detected&name=NAME&color=COLOR:

Esta função retorna uma lista com pequenas imagens recortadas com objetos específicos. Este método é igual ao getDetectedObjects mas com a diferença de confirmar a cor do objeto.

Listing 2.4: getDetectedObjectsByColor

```
1 def getDetectedObjectsByColor(name, color):
```

2.2.5 /put:

Esta função permite adicionar uma nova imagem:

Listing 2.5: addToDatabase

```
1  def upload(self, ufile):
2    upload_file = os.path.normpath(os.path.join("img/", ufile.filename))
3    if(len(ufile.filename) != 0) :
4      file = open(upload_file, 'wb')
5      while True:
6        data = ufile.file.read(8192)
7        if not data:
8          break
9        file.write(data)
10     msg = allImageProcessing("img/" + (...))
```

2.2.6 /get?id=IDENTIFIER:

Esta função permite, com o *Id* de uma imagem, obter uma certa imagem:

Listing 2.6: getById

```
1 def getById(id):
2   try:
3     img = db.execute("Select objPath From Base WHERE ObjId = ?", (id,))
```

```
4       result = [row[0] for row in img.fetchall()][0]
5       return json.dumps(result, indent=4, sort_keys=True)
```

2.3 Persistência

Esta parte do projeto tem como função guardar e aceder às informações numa base de dados. Para a produção da base de dados foi utilizado o *sqlite3*. Estes métodos vão ser utilizados pela *Aplicação Web* para obter as cores, a identificação dos objetos, registar imagens, etc...

2.4 Processador de imagens

Esta função é dada pelos professores da unidade curricular, e tem como objetivo processar uma imagem para o sistema.

Listing 2.7: getImageInfo

```
1  def getImageInfo(path):
2    session = requests.Session()
3    URL="http://image-dnn-sgh-jpbarraca.ws.atnog.av.it.pt/process"
4    with open(path, 'rb') as f:
5      file = {'img': f.read()}
6      r = session.post(url=URL, files=file, data=dict(thr=0.01))
7      if r.status_code == 200:
8        imgValues = r.json()
9      else:
10       imgValues = "null"
11
12     return imgValues
```

Capítulo 3

Bónus

Com a finalidade de o sistema oferecer mais funcionalidades ao utilizador, implementámos uma função para que seja permitido alterar uma imagem, usando filtro, neste caso cinzento ou sépia.

Ao clicar na *Listagem de Imagens de Objetos* irá aparecer o botão *Transformar*(tal como é demostrado no ponto 2.1.2) que nos levará para outra página onde é possível escolher entre os dois filtros.

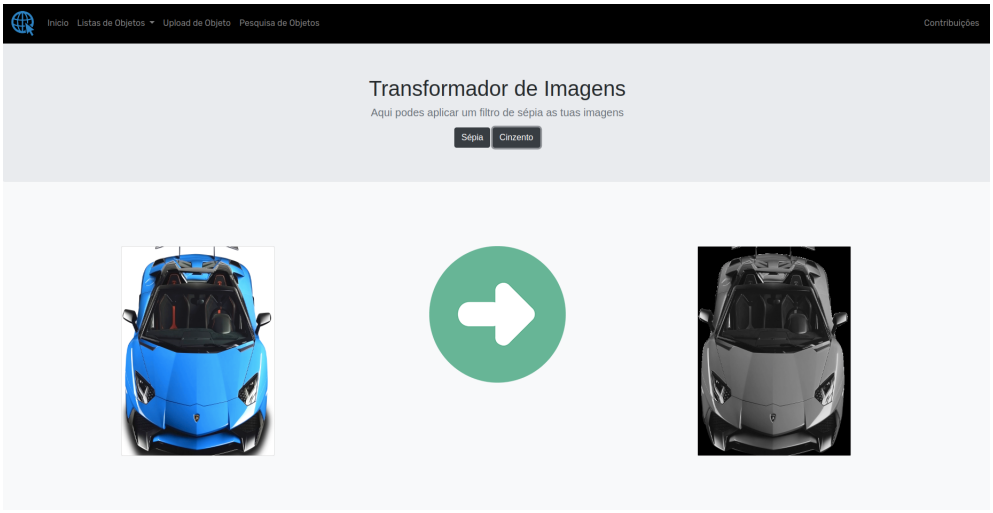

Figura 3.1: Exemplo, usando o filtro cizento

Capítulo 4

Testes efetuados

Um dos testes efetuados, para verificar o bom funcionamento do sistema, é por exemplo, introduzir uma imagem e verificar as funções do sistema.

Introduzir a imagem:

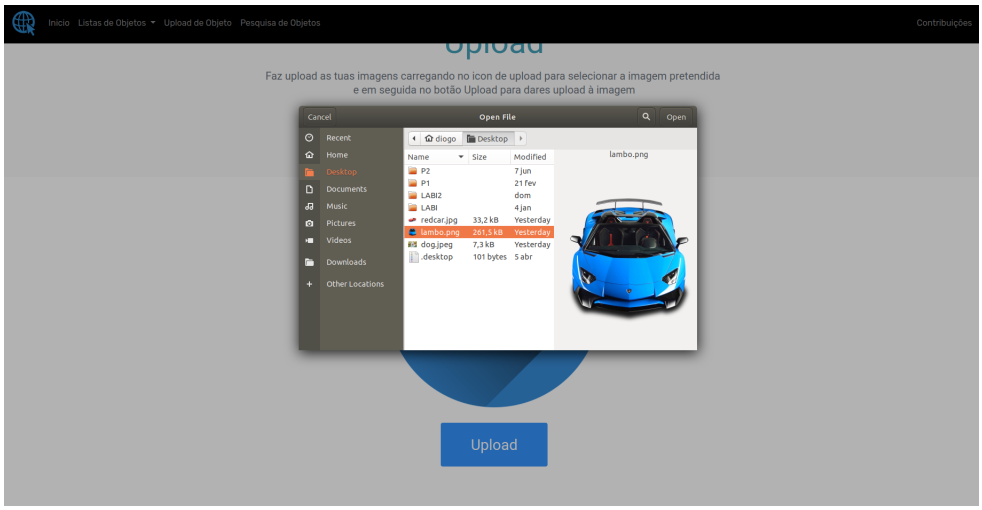

Figura 4.1: Upload

Se pesquisarmos "car"e escolhermos a cor "azul", irá aparecer, tal como suposto, a imagem introduzida. Também nas listas vai acontecer tudo como o previsto, assim demonstrando um bom funcionamento do sistema (comprovado nas seguintes imagens).

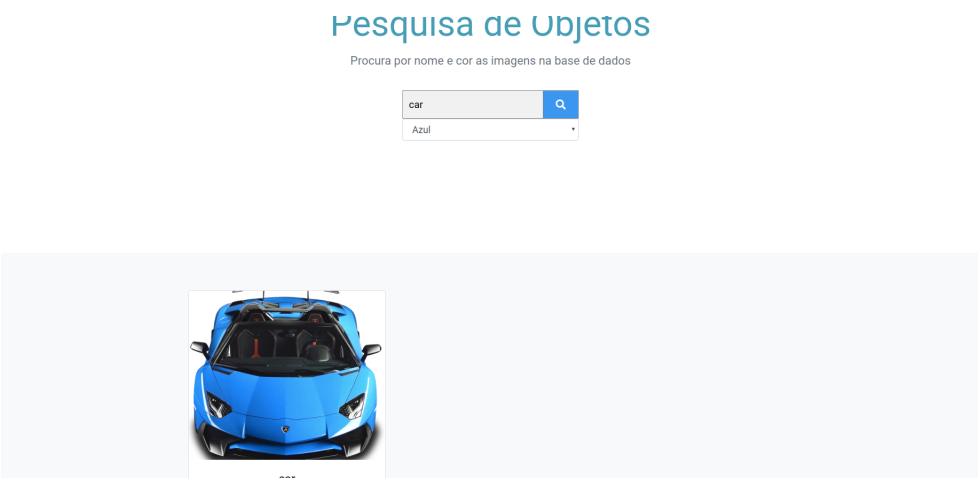

Figura 4.2: Demonstração de pesquisa por cor e objeto

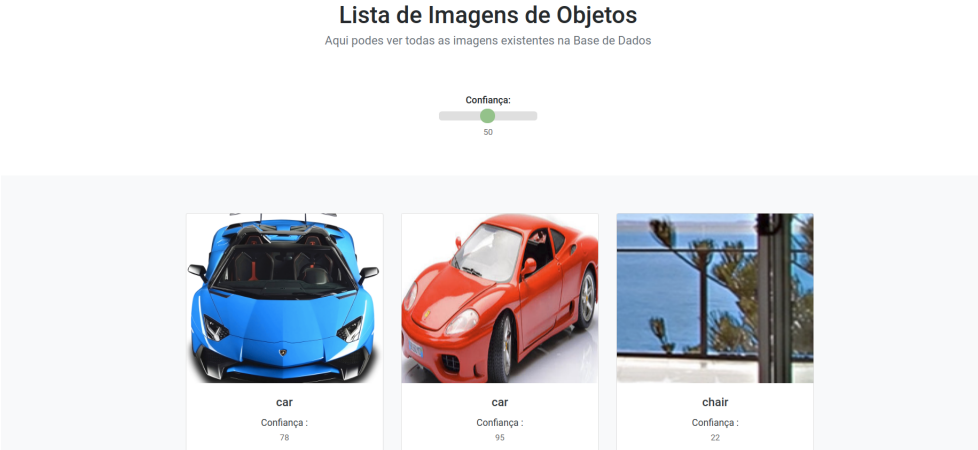

Figura 4.3: Demonstração da listagem da base de dados

Capítulo 5

Resultados

Com base no Capítulo 4 podemos analisar o funcionamento do sistema. Ao ser introduzido a imagem na base de dados, e posteriormente com a listagem desta, podemos verificar que a imagem foi introduzido com sucesso.

A pesquisa por cor e objeto, do mesmo modo que a listagem, ajuda-nos a perceber que o sistema está a correr como o suposto pois para selecionar apenas as imagens com o objeto escolhido e a cor escolhida é necessário verificar os itens na base de dados e em seguida mostrar apenas aqueles que correspondem aos requisitos do utilizador.

Os restantos serviços a dispor do utilizador, são mais uma vez, uma confirmação do funcionamento correto do sistema dado que fazem tudo o que é suposto.

Capítulo 6

Conclusões

Neste trabalho concluímos que é bastante importante a classificação e determinação de objetos para diversas áreas cientificas, sendo cada vez mais, por exemplo, com o avanço da inteligência artificial. Desta forma, o nosso projeto pode abrir novos horizontes para algumas aplicações e sistemas.

Como este projeto é o projeto final, tivemos uma especial atenção em realizar tudo com as maior competências e correção possível para assim o projeto ficar na sua melhor versão, e achamos que foi conseguido com clareza.

Contribuições dos autores

Para a produção deste projeto final, todos os membros do grupo esforçaram-se para que este trabalho fosse concluído com o maior sucesso. O Guilherme Pereira na realização do site e do ficheiro python, o Luís Costas mais focado na base de dados e no ficheiro python e o Diogo Amaral na produção do site e do relatório final.

Assim atribuindo a mesma percentagem a cada membro do grupo, 33% ao Guilherme Pereria, 33% ao Diogo Amaral e 33% ao Luís Costa.

CodeUA: `http://code.ua.pt//projects/labi2019-p2-g4`

Xcoa: `https://xcoa.av.it.pt/labi2019-p2-g4/index.html`

Acrónimos

HTML HyperText Markup Language

CSS Cascading Style Sheets

JS JavaScript

JSON JavaScript Object Notation