

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR
CIÊNCIA DA COMPUTAÇÃO

THIAGO DA SILVA ALVES

isa

TUTORIAL HAAR CASCADE

Tutorial desenvolvido como tarefa parcial durante a realização do estágio obrigatório no Departamento Acadêmico de Computação (DACOM) da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Arnaldo Candido Junior.

Supervisor: Prof. Dr. Pedro Luiz de Paula Filho.

MEDIANEIRA
2017

Todos os exemplos foram desenvolvidos utilizando:

Opencv 3.1.0-dev.

Python 3.5.2.

1 - Download do conjunto de imagens.

Para que se possa fazer o treinamento com haar cascade, primeiramente serão necessários muitas imagens que mostre o objeto que se deseja reconhecer (positive samples), e ainda mais imagens que não contenham o objeto (negative samples).

Um site onde se pode encontrar vários bancos de imagens é o <http://image-net.org/>.

Para utilizá-lo basta buscar pelo objeto a ser reconhecido, logo após aparecerá várias imagens referentes a ele, você pode baixá-las uma a uma ou ir na sessão de downloads e clicar no link URLs **Figura 1**. Com isso, será apresentado um txt com os links de download para todas as imagens.

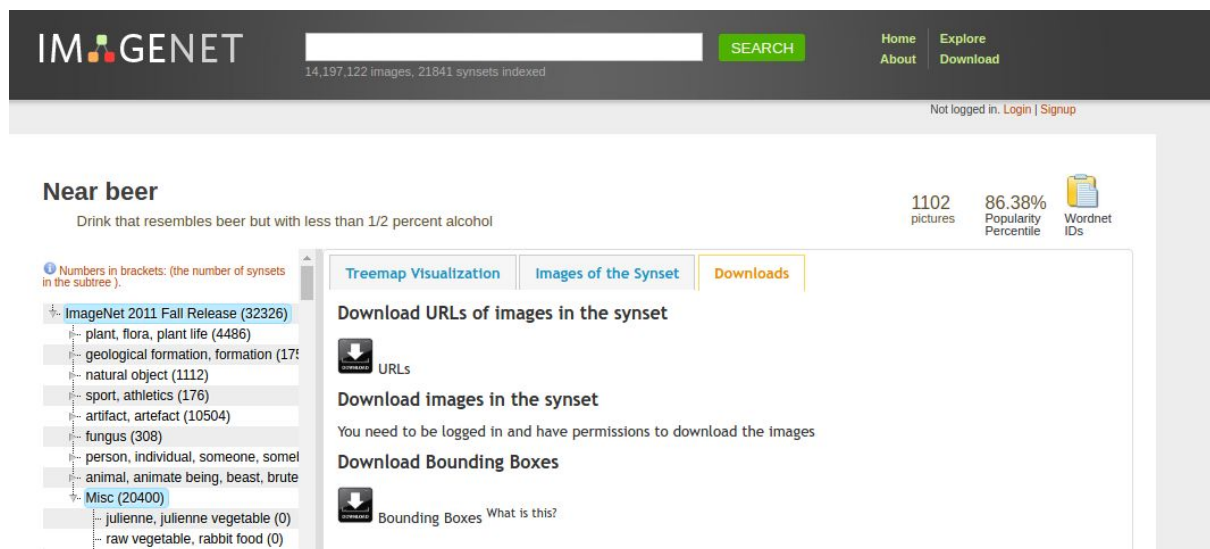


Figura 1

1.1 - Automatizando o download das imagens.

Na Universidade foi desenvolvido o software **Downloader.py**, com ele é possível por meio de uma lista de URLs, baixar de uma vez um conjunto de imagens e realizar um pré processamento sobre delas.

Os passos para a utilização do Downloader.py e efetuação dos downloads são os seguintes:

1.1.1 - Buscar uma lista de urls.

Como dito anteriormente, o website <http://image-net.org/> é uma ótima ferramenta para a obtenção de imagens de domínio público. Para utilizá-lo você deve buscar pelo conjunto de imagens que deseja, dentre os conjuntos de imagens encontrados, os chamados **synsets** **Figura 2**, escolha o que mais se enquadrar em suas necessidades.

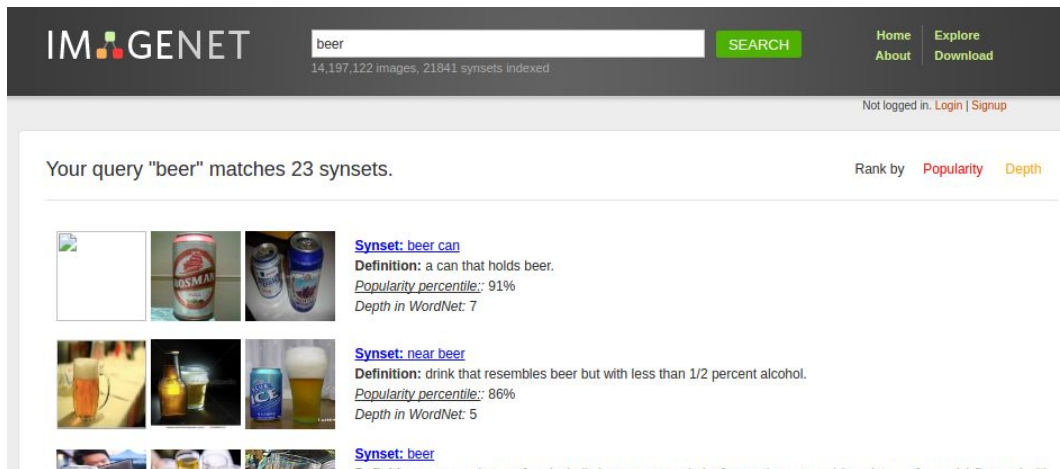


Figura 2

Para o exemplo, o primeiro link foi escolhido. Após escolher o seu, na aba Downloads, clique em URLs **Figura 3**.

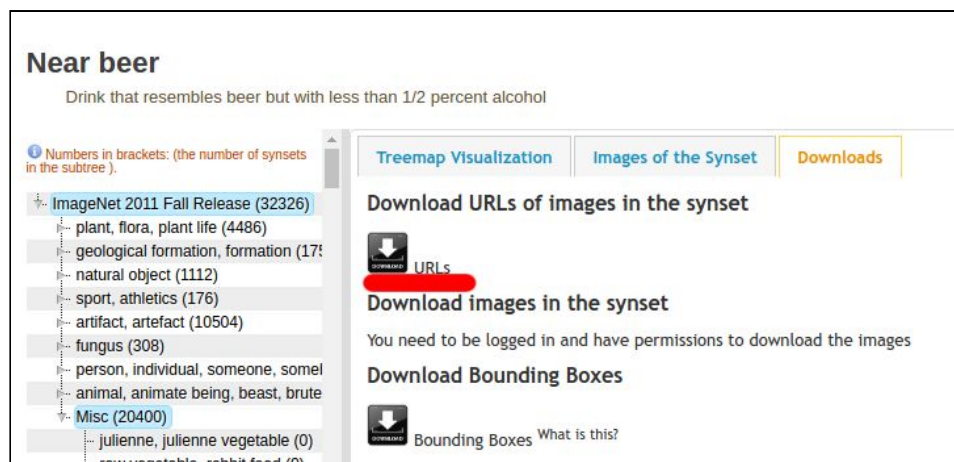


Figura 3

Clicando no link **URLs** por sua vez, redirecionará para um arquivo contendo um conjunto de URLs **Figura 4**, e são essas urls que o **Downloader.py** utilizará para conseguir as imagens necessárias para o treinamento.

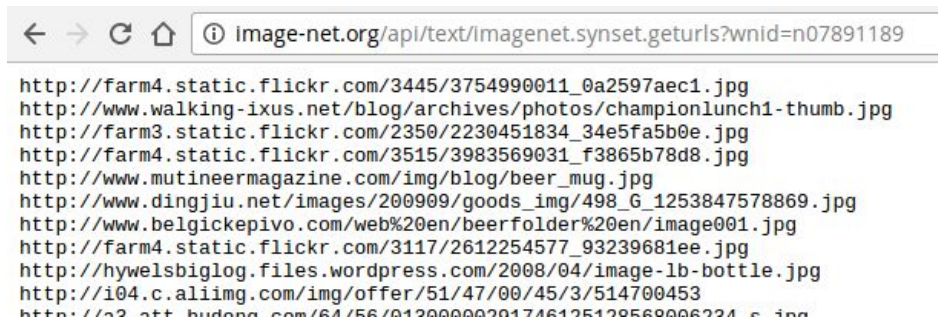


Figura 4

1.1.2 - Efetuar o download das imagens.

O comando utilizado para o uso do **Downloader.py** é o apresentado abaixo:

*Todos os argumentos que tiverem um * ao seu lado não são obrigatórios, o comando deve ser executado sem os *.*

\$ python Downloader.py

--urls=http://image-net.org/api/text/imagenet.synset.geturls?wnid=n07891189

--out="/imagens" --timeout=2* --img-extensions='(".jpg", ".jpeg", ".png", ".bmp")'

--prefix=img* --out-extension=".png"* --no-convert-gray* --no-resize*

--max-size=500.0* --no-std-names* --default-images="/defaults"

python:

Comando que diz que o programa a ser executado se trata de um software escrito em python. Você deve se atentar, pois em sua máquina pode ter várias versões do python instaladas ex: **Figura 5**. O software necessita de pelo menos a versão 3.5 da linguagem.

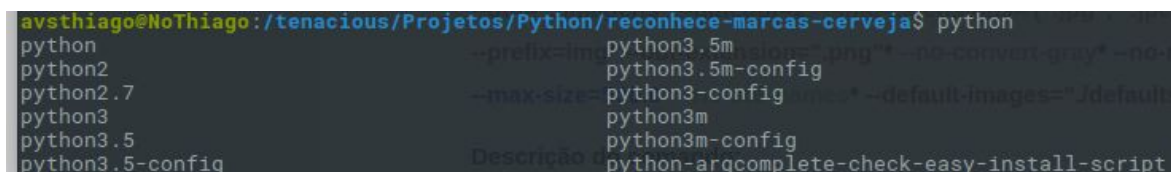


Figura 5

Downloader.py

O software em si, que está incumbido de efetuar o download das imagens contidas na url.

--urls=http://image-net.org/api/text/imagenet.synset.geturls?wnid=n07891189

Esse argumento é obrigatório. Pode ser passado para ele uma url com o da **Figura 4** ou um caminho até um arquivo de texto que contenha linha a linha o link das imagens que devem ser baixadas.

--out="./imagens"

Outro argumento obrigatório e tem por objetivo armazenar o diretório onde as imagens baixadas serão salvas.

--timeout=2

Quando se conecta a um servidor para fazer o download de uma imagem, ele pode ficar com a conexão aberta por muito tempo sem iniciar o download. Esse argumento tem por objetivo passar o tempo em **segundos** que uma conexão pode ficar aguardando o início do download, se esse tempo for atingido, o software passa para o download da imagem seguinte. O argumento não é obrigatório e tem por padrão o valor de 2 segundos.

--img-extensions='(".jpg", ".jpeg", ".png", ".bmp")'

Extensões aceitas para as imagens baixadas. Argumento não obrigatório que tem como padrão as seguintes extensões .jpg, .jpeg, .png, .bmp.

--prefix=img

Após o download das imagens estar completo, elas podem ter uma padronização de seu nome (argumento + um número sequencial **Figura 6**), o argumento não é obrigatório e por padrão o prefixo é **img**.

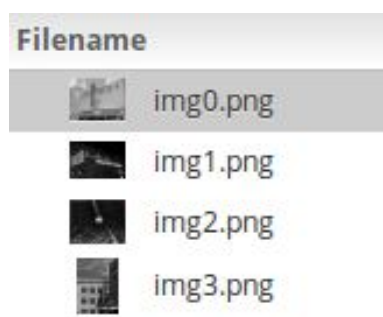


Figura 6

--out-extension=".png"

Após o processamento das imagens, a extensão delas é padronizada. A extensão pode ser escolhida, sendo que seu padrão é .png. Este argumento não é obrigatório.

--no-convert-gray

Por padrão, após o download as imagens são convertidas para a escala de cinza. Caso não queira essa ação, basta inserir esse argumento. Ele não possui valor.

--no-resize

Por padrão, após o download as imagens são redimensionadas, pelo motivo que o haar não necessita de imagens com uma alta resolução para operar. Adicione este argumento caso não queira que as imagens sejam redimensionadas. Caso queira alterar o tamanho máximo do redimensionamento altere o argumento **--max-size=500.0**.

--max-size=500.0

Esse argumento tem por objetivo definir o tamanho em pixels que o lado maior da imagem terá. Caso ela seja de 1000x500, após o seu redimensionamento ela passará ter o tamanho de 500x250. Sempre o lado maior será redimensionado para o valor informado e o menor será dimensionado proporcionalmente. O argumento não é obrigatório e tem seu valor padrão de 500px, caso não queira que as imagens tenham seu valor alterado adicione o argumento **--no-resize**.

--no-std-names

Caso não queira que o nome de suas imagens seja padronizado de acordo com o parâmetro **--prefix**, adicione esse argumento. Com ele o nome original das imagens será preservado.

--default-images="./defaults"

Quando são feitos downloads das imagens pode ser que muitas não existam mais, quando isso acontece os sites utilizam de uma imagem padrão para representar a antiga **Figura 7**. O Downloader.py também contém um removedor de imagens indesejadas, para que ele elimine essas imagens, coloque exemplos delas em um diretório, informe o diretório nesse

argumento, dessa forma, o software irá comparar as imagens baixadas com as do diretório, se forem semelhantes elas serão removidas. O argumento não é obrigatório.

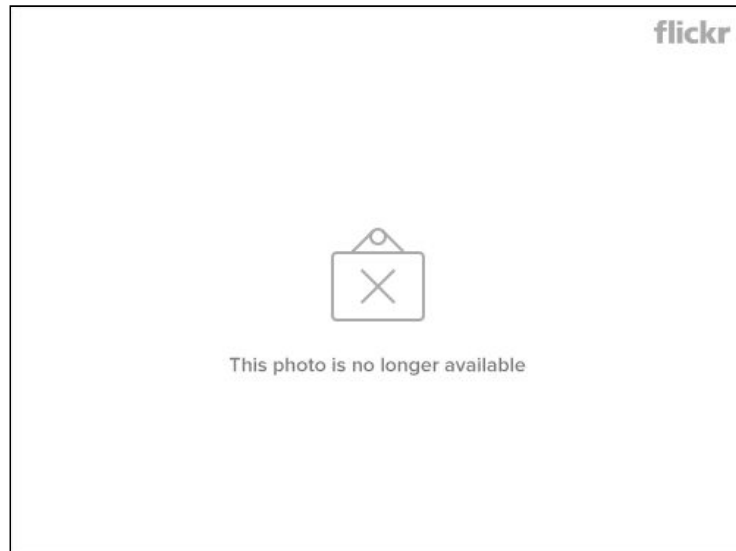


Figura 7

Após executar o comando as imagens começaram a ser baixadas e salvas no diretório escolhido **Figura 8**, o pré processamento delas será feito após ter feito o download de todas.

```
avsthiago@NoThiago:/tenacious/Projetos/Python/reconhece-marcas-cerveja$ python Downloader.py --urls=http://image-net.org/api/text/imagenet.synset.geturls?wnid=n07891189 --out="./imagens" --timeout=3 --img-extensions=".jpg", ".jpeg", ".png", ".bmp")", --prefix=img --out-extension=".png" --no-convert-gray --no-resize --max-size=500.0 --no-std-names
http://farm4.static.flickr.com/3445/3754990011_0a2597aec1.jpg
http://farm3.static.flickr.com/2350/2230451834_34e5fa5b0e.jpg
http://farm4.static.flickr.com/3515/3983569031_f3865b78d8.jpg
http://www.mutineermagazine.com/img/blog/beer_mug.jpg
```

Figura 8

Se precisar de ajuda para relembrar dos argumentos, basta digitar o seguinte comando e uma ajuda será exibida **Figura 9**:

\$ python Downloader.py -h

```
avsthiago@NoThiago:/tenacious/Projetos/Python/reconhece-marcas- cerveja$ python Downloader.py -h
usage: Downloader.py [-h] --urls URLS --out OUT [--timeout TIMEOUT]
                    [--img-extensions IMG_EXTENSIONS] [--prefix PREFIX]
                    [--out-extension OUT_EXTENSION]
                    [--default-images DEFAULT_IMAGES] [--no-convert-gray]
                    [--no-resize] [--max-size MAX_SIZE] [--no-std-names]

optional arguments:
  -h, --help            show this help message and exit
  --urls URLS           File where urls are, can be an site or a file.
  --out OUT             Where the images will be saved.
  --timeout TIMEOUT     Maximum time to download each image.
  --img-extensions IMG_EXTENSIONS
                        Tuple of image extensions that will be accepted.
  --prefix PREFIX       Name prefix to save images.
  --out-extension OUT_EXTENSION
                        Extension to use when sava an image.
  --default-images DEFAULT_IMAGES
                        Path that contains default images, these will be used
                        to verify if the downloaded image is a default image
                        and delete it.
  --no-convert-gray     Don't convert downloaded images to gray.
  --no-resize           Don't resizes each downloaded image.
  --max-size MAX_SIZE   Resizes the bigger side of an image to fit in this
                        param.
  --no-std-names        Don't standardize image names using --prefix
```

Figura 9

2 - Arquivos necessários para o treinamento

É preciso ter em mente que o treinamento pelo Haar requer uma certa estrutura de arquivos para poder ser executado. A estrutura é a seguinte **Figura 10**:



Figura 10

2.1 O que são imagens negativas?

Imagens negativas são toda e qualquer imagem que não contenha o objeto que deverá ser encontrado.

Sobre a quantidade de imagens negativas necessárias para o treinamento, fala-se sobre a seguinte proporção numPos:numNeg 1:2 [1], mas assim como também é dito na referência, a quantidade dependerá do propósito da aplicação.

2.2 Pasta contendo as imagens negativas

Caso não tenha ainda as imagens negativas, veja na seção 1 como pode baixar um conjunto de imagens genéricas e salvá-las em um diretório.

2.3 Arquivo que descreve o diretório onde as imagens negativas estão

As imagens negativas devem ser mapeadas por um arquivo de texto que em cada linha guarda o caminho até uma imagem.

Na imagem abaixo é possível ver a estrutura do arquivo de exemplo imagensNegativas.txt. Nele, em cada linha há o diretório para uma imagem negativa.

```
/imagensNegativas.txt  
img1.jpg  
img2.jpg  
img3.jpg
```

Figura 11

Para criar um arquivo como esse de exemplo pode-se utilizar o software desenvolvido **CreateImagesList.py**, o funcionamento dele se dá da seguinte forma:

No *bash* digite a linha de comando abaixo, mudando os argumentos de acordo com suas necessidades.

```
python CreateImagesList.py --path ./negativeImagesPath --out listOfNegativeImages
```

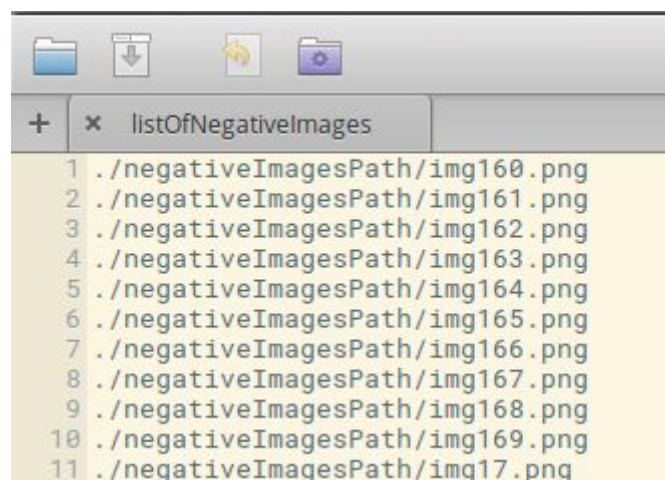
--path

Diretório que contém as imagens negativas.

--out

Nome do arquivo de saída que conterá o caminho para as imagens negativas.

Abaixo o arquivo criado pela linha de comando acima



The image shows a screenshot of a text editor window titled 'listOfNegativeImages'. The window contains a list of 11 file paths, each on a new line, numbered 1 through 11. The paths are all relative to the current directory and point to image files in the 'negativeImagesPath' directory. The files are named 'img160.png' through 'img169.png', with the last line showing 'img17.png' which might be a typo for 'img170.png'.

```
1 ./negativeImagesPath/img160.png  
2 ./negativeImagesPath/img161.png  
3 ./negativeImagesPath/img162.png  
4 ./negativeImagesPath/img163.png  
5 ./negativeImagesPath/img164.png  
6 ./negativeImagesPath/img165.png  
7 ./negativeImagesPath/img166.png  
8 ./negativeImagesPath/img167.png  
9 ./negativeImagesPath/img168.png  
10 ./negativeImagesPath/img169.png  
11 ./negativeImagesPath/img17.png
```

Figura 12

2.4 Arquivo .vec

O arquivo .vec necessário para o treinamento do haar, é um arquivo que contém miniaturas de todos samples positivos **Figura 13**. As imagens positivas em si não são utilizadas para o treinamento, somente o .vec que carrega suas características.



Figura 13

2.4.1 Visualizando as imagens dentro de um arquivo .vec

Um arquivo .vec não pode ser exibido pela maioria dos visualizadores de imagens, logo, o OpenCV têm uma ferramenta para a exibição de seu conteúdo. Para visualizá-lo, digite o seguinte comando em seu bash.

*Caso tenha informado valores diferentes do padrão (24x24) de tamanho, o novo tamanho também deve ser informado no comando abaixo.

opencv_createsamples -vec vetor.vec -show

-vec

Nome do arquivo .vec que você deseja exibir.

-show

Esse argumento diz ao programa **opencv_createsamples** que o conteúdo do arquivo .vec deve ser exibido.

Pressione enter e as imagens que o arquivo contém serão exibidas uma a uma **Figura 14**, pressione alguma tecla para passar para a próxima.

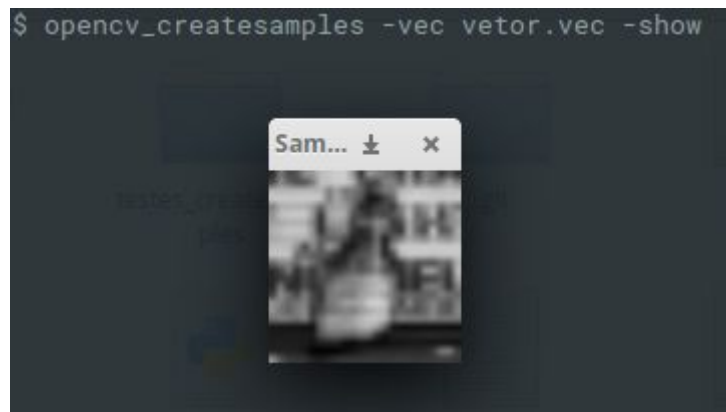


Figura 14

2.4.2 Imagens positivas

Imagens positivas são imagens que contém o objeto a ser reconhecido. É importante que a imagem seja um retângulo que engloba o objeto que se quer reconhecer. Caso exista muitos outros detalhes na imagem é importante recortar somente o objeto, isso será demonstrado mais adiante. Também é necessário que as imagens positivas generalizem ao máximo o objeto a ser reconhecido, exibindo-o em diversos ângulos, tamanhos e com variações de iluminação.

Sobre a quantidade de imagens positivas necessárias para o treinamento, não há um número fixo, pois depende do propósito do reconhecimento. Caso o objeto a ser encontrado apareça em ocasiões padronizadas, menos imagens positivas serão necessárias, mas se no momento de procurá-lo em imagens, ele aparece em diferentes ângulos, tamanhos ou variações de cor, um número maior de exemplos será necessário [1], podendo variar de centenas até milhares.

Como dito anteriormente, não será sobre as imagens positivas que o treinamento será feito e sim sobre o arquivo .vec que é criado sobre as imagens positivas, a mais adiante serão mostradas as formas de criá-lo.

2.4.3 Aquisição de imagens positivas.

Para conseguir as imagens positivas existem duas principais formas:

- 1) Encontrar um conjunto de imagens onde a aparece o objeto e recortar de cada uma um retângulo onde somente ele aparece (pode aparecer uma pequena parte do fundo, mas quanto menos melhor), isso pode ser feito com o auxílio da ferramenta **opencv_annotation** [3] na seção 2.4.4 será apresentado seu funcionamento.
- 2) Utilizar a ferramenta **opencv_createsamples** ou **CreateSamples.py** (essa que utiliza o **opencv_createsamples** para operar, mas simplifica várias tarefas para o usuário). Instruções sobre o funcionamento da ferramenta **opencv_createsamples** podem ser encontradas em [4], nas próximas seções será explicado o funcionamento do **CreateSamples.py**.

Se for escolhido a primeira opção, serão necessárias várias imagens do objeto. Caso tenha dificuldade de encontrá-las procure por softwares como [5] que extraem frames de vídeos, assim é possível gravar um vídeo de seu objeto e obter diversas imagens dele.

2.4.4 Funcionamento opencv_annotation

O **opencv_annotation** é instalado juntamente com o as outras ferramentas do OpenCV. Sua principal função é por meio de um conjunto de imagens que possuam o objeto a ser reconhecido, contar com o auxílio do usuário para marcar na imagem onde que os objetos estão **Figura 14** e a partir dessas marcações, gerar um arquivo que diga onde cada imagem está salva, quantos objetos ela possui e quais são os locais da imagem que apresentam o objeto **Figura 15**.

./positivas/img250.jpg 2 71 233 135 70 207 236 121 65

Localização da imagem no disco.

Quantidade de objetos existentes na imagem.

Coordenadas do primeiro objeto os números significam: Coordenada X do primeiro ponto (71), Coordenada Y do primeiro ponto (233), largura do retângulo em pixels (135), altura do retângulo em pixels (70).

O mesmo vale para as coordenadas do segundo objeto.



Figura 14

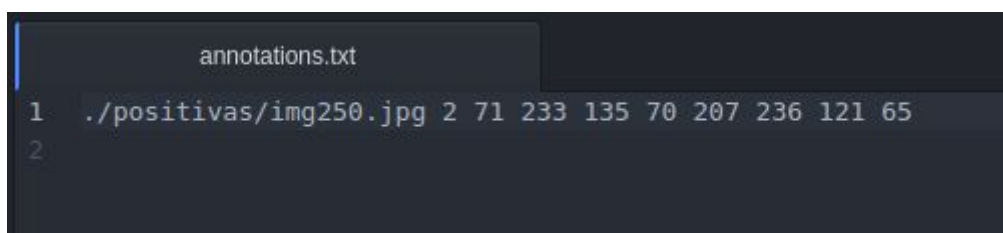


Figura 15

O uso do software se dá da seguinte forma [3]:

- 1 - Crie uma pasta com as imagens que serão marcadas.
- 2 - Execute o comando abaixo

A quantidade de - em frente ao argumento foi alterada para -- (dois "-") na versão 3.2.0 do OpenCV

opencv_annotation -images ./positivas/ -annotations annotations.txt

-images

Diretório onde se encontram as imagens positivas que serão utilizadas para selecionar os objetos. Dê preferência por caminhos absolutos.

-annotations

Arquivo de saída que conterà o caminho para cada imagem, a quantidade de objetos e suas localizações.

- 3 - A ferramenta abrirá as imagens da pasta uma a uma em uma ordem aleatória.

4 - Use o mouse para selecionar a região de interesse, mantendo as bordas o máximo perto do objeto que você deseja treinar. Quando estiver ok com a marcação pressione a tecla **c** de seu teclado para confirmar.

5 - Continue selecionando os objetos que você deseja.

6 - Pressione a tecla **n** de seu teclado para carregar a próxima imagem.

7 - Pressione a tecla **d** para remover a última marcação (versões mais recentes do OpenCV 3.2[12])

8 - Para parar pressione ESC.

Após o final desse processo você terá em mãos um arquivo de anotação semelhante a **Figura 15**. Com ele será possível criar o arquivo `.vec`. A criação do arquivo `.vec` está descrita na seção 2.4.6 deste documento.

Alternativas para o `opencv_annotation` são [8, 9, 11] essas são genéricas, então não terão um arquivo de saída semelhante ao reconhecido pelo criador do `.vec`. Uma alternativa que gera um arquivo reconhecível pode ser encontrado nesse artigo [10].

2.4.5 Criação de um conjunto de imagens positivas com o auxílio da ferramenta CreateSamples.py

A criação de imagens positivas pela forma descrita na seção anterior pode ser muito trabalhosa, já que será necessário recortar manualmente centenas ou milhares de objetos de imagens. Pensando nisso o OpenCV conta com a ferramenta `opencv_createsamples`. Com ela é possível pegar uma imagem do objeto a ser reconhecido **Figura 16** e inseri-lo de formas distorcidas em várias imagens antes negativas **Figura 17**, podendo dessa forma gerar uma infinidade de novas imagens positivas. Para trabalhar com várias imagens positivas geradoras ao invés de uma com o `opencv_createsamples` pode ser um pouco trabalhoso [6], pensando nisso foi criado o `CreateSamples.py`, existe uma ferramenta que faz um processo semelhante [7], mas o `CreateSamples.py` ainda o supera avsthiago@gmail.com na questão de facilitação do processo para o usuário.



Figura 16



SC

Figura 17

Para utilizar o CreateSamples.py utiliza o comando abaixo, os argumentos que contenham um * não são obrigatórios, mas caso não informe o --vec e também o --out-image-folder, nada será gerado. Não insira os * no comando quando for executar.

```
python CreateSamples.py --vec vetorpos.vec* --pos pos.txt --bg bg.txt --num 500
--bgcolor* 0 --bgthresh* 80 --inv* --randinv* --out-image-folder* ./saida/ --maxidev* 40
--maxzangle* 0.50 --maxxangle* 1.10 --maxyangle* 1.10 --width* 15 --height* 25
```

--vec

Arquivo que será criado para fazer o treinamento do cascade, não é obrigatório criá-lo agora, pois pode ser gerado posteriormente como será apresentado nas próximas seções, mas gerando-o agora poupará trabalhos futuros.

--pos

Um arquivo de texto que armazenará o caminho para as imagens positivas originais que servirão de base para a criação das novas imagens positivas, ele pode ser criado assim como na seção 2.3 com o CreateImagesList.py.

```
pos.txt
1 ./testes_create_samples/bud100x28.jpg
2 ./testes_create_samples/bud100xa28.jpg
3
```

Figura 18

As Imagens positivas devem ter proporções semelhantes e apresentar o objeto a ser reconhecido sob diferentes ângulos e iluminações **Figura 19**.

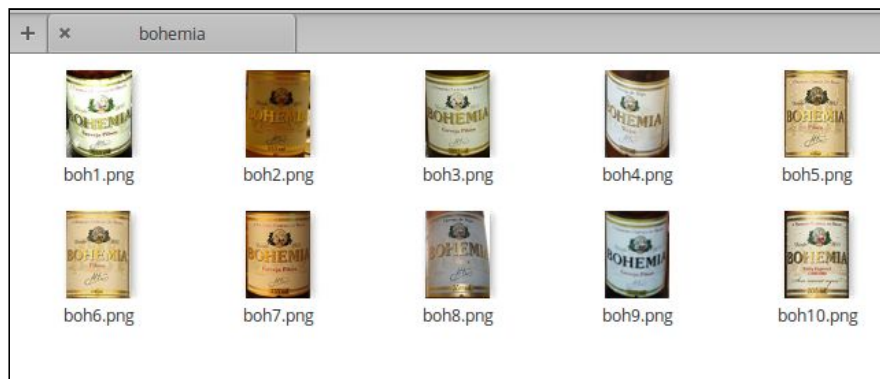


Figura 19

--bg

Um arquivo de texto que armazenará o caminho para um conjunto de imagens genéricas que não contenha o objeto, elas servirão de background **Figura 20**. As imagens positivas serão distorcidas e colocadas sobre elas **Figura 17**. Essas imagens devem ser um tamanho maior que as imagens contidas no **--pos**. O arquivo pode ser criado assim como na seção **2.3** com o **CreateImagesList.py**.

```

bg.txt
1 ./imagens/img160.png
2 ./imagens/img161.png
3 ./imagens/img162.png
4 ./imagens/img163.png
5 ./imagens/img164.png
6 ./imagens/img165.png
7 ./imagens/img166.png
8 ./imagens/img167.png
9 ./imagens/img168.png
10 ./imagens/img169.png
11 ./imagens/img17.png
12 ./imagens/img170.png

```

Figura 20

--num

Quantidade de imagens positivas que serão geradas.

--bgcolor

Cor de fundo, (atualmente imagens em tons de cinza são assumidas). A cor de fundo denota a cor transparente. A quantidade de tolerância de cores pode ser informada pelo **--bgthresh**. Todos os pixels entre **bgcolor-bgthresh** e **bgcolor+bgthresh** range serão interpretados como transparente. Não é obrigatório e tem seu valor padrão Não é obrigatório e tem seu valor padrão **0**.

--bgthresh

Threshold para remover o background da imagem positiva, quando a **Figura 16** foi inserida na **Figura 17**, seu fundo preto foi removido setando o --bgthresh como 1, pelo fundo ser preto "0" ele passou a ser transparente a partir de sua inserção na imagem. Não é obrigatório e tem seu valor padrão **80**.

--inv

Inverte as cores da imagem antes de inseri-la no background.

--randinv

Randomiza as cores que irá inverter antes de inserir a imagem no background.

--out-image-folder

Após processar as imagens, caso esse argumento tenha sido informado, todas as imagens geradas e o arquivo com as marcações das imagens positivas serão salvos do diretório informado **Figura 21**.

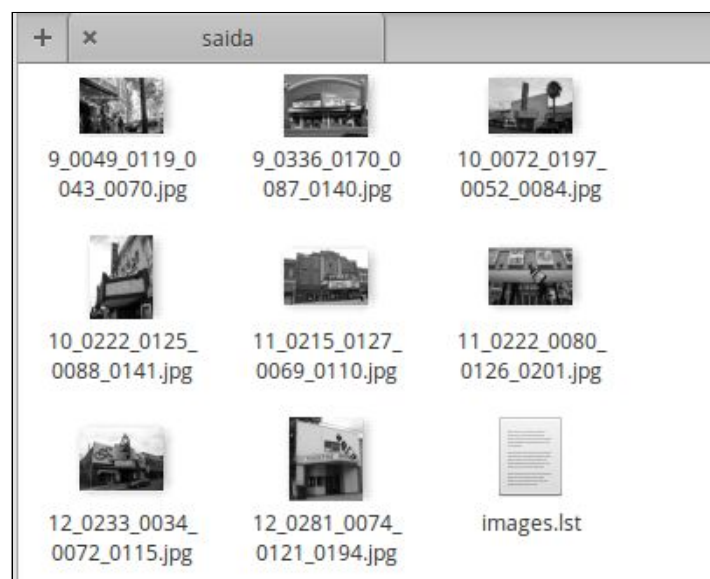


Figura 21

O arquivo images.lst armazena a descrição das imagens que foram geradas, quantos objetos há em cada imagem e a localização deles, assim como o arquivo gerado pelo opencv_annotation na seção **2.4.4**.

--maxidev

Máximo de variação de luminosidade, não é obrigatório e têm seu valor padrão de **40**.

--maxzangle

Rotação máxima no eixo **Z**. O valor deve ser em radianos. Não é obrigatório e têm seu valor padrão de **0.5**.

--maxxangle

Rotação máxima no eixo **X**. O valor deve ser em radianos. Não é obrigatório e têm seu valor padrão de **1.1**.

--maxyangle

Rotação máxima no eixo **Y**. O valor deve ser em radianos. Não é obrigatório e têm seu valor padrão de **1.1**.

--width

Largura da imagem positiva que será inserida no background. Esse tamanho não deve ser grande, pois pode aumentar muito o tempo para treinar o reconhecedor já que terão muitas características para analisar. O valor padrão para ele é de 24px, mas o mais importante é ele ser proporcional a imagem original.

--height

Altura da imagem positiva que será inserida no background. Esse tamanho não deve ser grande, pois pode aumentar muito o tempo para treinar o reconhecedor já que terão muitas características para analisar. O valor padrão para ele é de 24px, mas o mais importante é ele ser proporcional a imagem original.

2.4.6 Criação do arquivo .vec

E por fim a criação do arquivo .vec caso ainda não tenha sido criado enquanto na seção 2.4.5.

A criação é feita pelo comando abaixo, ele utiliza o opencv_createsamples:

```
opencv_createsamples -info info/info.lst -num 600 -w 22 -h 80 -vec positives.vec
```

-info

Um arquivo que contém uma lista de imagens positivas em cada linha, acompanhada pela quantidade de objetos e a localização de cada um, assim como nas **Figuras 15 e 18**. Esse é o arquivo que é criado pelo opencv_annotations, que também é o mesmo que se encontra no **--out-image-folder** quando esse argumento é informado ao CreateSamples.py.

-num

Quantidade de imagens que serão colocadas dentro do arquivo .vec, essa quantidade deve ser menor ou igual ao número de arquivos descritos no -info.

-w

Largura da imagem positiva que será inserida vetor de saída. Esse tamanho não deve ser grande, pois pode aumentar muito o tempo para treinar o reconhecedor já que terão muitas características para analisar. O valor padrão para ele é de 24px, mas o mais importante é ele ser proporcional a imagem original.

-h

Altura da imagem positiva que será inserida vetor de saída. Esse tamanho não deve ser grande, pois pode aumentar muito o tempo para treinar o reconhecedor já que terão muitas características para analisar. O valor padrão para ele é de 24px, mas o mais importante é ele ser proporcional a imagem original.

-vec

Arquivo de saída que será gerado.

3 Treinamento do classificador pelo opencv_traincascade:

O próximo passo é o treinamento baseado nos datasets positivos e negativos construídos anteriormente.

O treinamento será realizado com o auxílio da ferramenta opencv_traincascade. Abaixo estão os argumentos que ela suporta. Eles estão agrupados de acordo com o seu propósito.

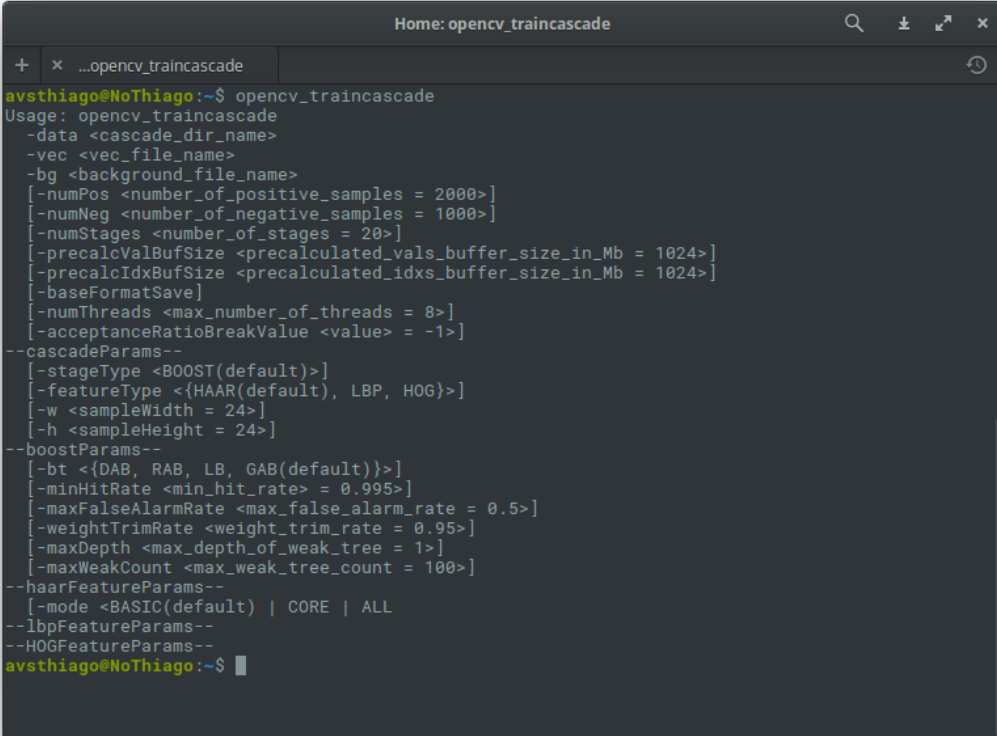
- Argumentos frequentes:
 - **-data <cascade_dir_name>** : Onde o classificador treinado será armazenado. Essa pasta deve ser previamente criada.
 - **-vec <vec_file_name>** : Arquivo .vec, criado pelo CreateSamples.py ou pelo opencv_createsamples.
 - **-bg <background_file_name>** : Arquivo que lista as imagens negativas. Como visto na seção 2.3.
 - **-numPos <number_of_positive_samples>** : Número de imagens positivas que serão utilizadas em cada estágio da classificação. Nem todas as imagens dentro do arquivo .vec são utilizadas, algumas são removidas no processo. Então não colocar o número máximo de imagens no arquivo vec para esse argumento. Segundo [17], um bom valor seria $0.9 * \text{o número de imagens no .vec}$. Ou para maior garantia 0.8. Mais informações em [16].
 - **-numNeg <number_of_negative_samples>** : Número de imagens negativas que serão utilizadas em cada estágio da classificação.
 - **-numStages <number_of_stages>** : Número de estágios que serão utilizados para treinar o classificador. Em cada estágio um número maior de características são escolhidas. O tempo entre um estágio e outro no treinamento é exponencialmente maior. De 10 a 15 estágios bons resultados já são conseguidos.
 - **-precalcValBufSize <precalculated_vals_buffer_size_in_Mb>** : Quantidade de memória para o buffer para o pré-cálculo de características. Tamanho (em Mb). Quanto mais memória você atribuir, mais rápido será o

treinamento. Cuidado para que `-precalcValBufSize` e o `-precalcIdxBufSize` combinados não ultrapassem o total de memória disponível no sistema.

- **-precalcIdxBufSize** **<precalculated_idx_buffer_size_in_Mb>** : Quantidade de memória para o buffer para o pré-cálculo de índices das características. Tamanho (em Mb). Quanto mais memória você atribuir, mais rápido será o treinamento. Cuidado para que `-precalcValBufSize` e o `-precalcIdxBufSize` combinados não ultrapassem o total de memória disponível no sistema.
- **-baseFormatSave** : O arquivo de saída será gerado no formato antigo caso esse argumento seja informado.
- **-numThreads** **<max_number_of_threads>** : Número máximo de threads a serem utilizadas durante o treinamento. É necessário ter compilado o OpenCV com suporte a TBB para essa opção.
- **-acceptanceRatioBreakValue** **<break_value>** : Esse argumento é utilizado para determinar o quanto preciso o modelo deve se manter para aprendendo e quando deve parar. Uma dica é não treinar acima de $10e-5$, para assegurar que seu modelo não fique super treinado em sua base de treinamento.
- Parâmetros cascade:
 - **-stageType** **<BOOST(default)>** : Tipo de estágios. Até o momento atual, somente classificadores *boosted* suportam essa característica.
 - **-featureType****<{HAAR(default), LBP}>** : Tipo de características que serão extraídas. HAAR - Haar-like features ou LBP - local binary patterns. O LBP tem um treinamento muito mais rápido que o HAAR
 - **-w** **<sampleWidth>** : Largura das imagens contidas no arquivo .vec (em pixels). Devem ter o mesmo tamanho de quando o .vec foi criado.
 - **-h** **<sampleHeight>** : Altura das imagens contidas no arquivo .vec (em pixels). Devem ter o mesmo tamanho de quando o .vec foi criado.

- Parâmetros para *Boosted classifiers*:
 - **-bt <{DAB, RAB, LB, GAB(default)}>** : Tipo de *boosted classifiers* a ser utilizado: DAB - Discrete AdaBoost, RAB - Real AdaBoost, LB - LogitBoost, GAB - Gentle AdaBoost [20].
 - **-minHitRate <min_hit_rate>** : Mínima quantidade de acerto desejada para cada estágio do treinamento. [21] §4.1.
 - **-maxFalseAlarmRate <max_false_alarm_rate>** : Número máximo de falsos positivos desejados para cada estágio do treinamento. [21] §4.1.
 - **-weightTrimRate <weight_trim_rate>** : Especifica se o trimming deve ser utilizado como peso. Uma boa escolha é de 0.95.
 - **-maxDepth <max_depth_of_weak_tree>** : Profundidade máxima da *weak tree*.
 - **-maxWeakCount <max_weak_tree_count>** : Quantidade máxima de weak trees para cada estágio do cascade. O classificador *boosted* terá a quantidade de weak trees ($\leq \text{maxWeakCount}$), que for necessário para alcançar o -maxFalseAlarmRate que foi informado.
- **Parâmetros para características Haar:**
 - **-mode <BASIC (default) | CORE | ALL>** : Selecione o tipo de características Haar que serão utilizadas no treinamento. BASIC somente usa características horizontais e verticais, já o ALL utiliza além das verticais e horizontais, também utiliza características rotacionadas 45 graus. Veja mais detalhes em [22].
- **Parâmetros para o Local Binary Patterns:** Local Binary Patterns não possui parâmetros especiais.

Valores padrão caso não informe algum argumento

A terminal window titled 'Home: opencv_traincascade' showing the help text for the 'opencv_traincascade' command. The text is displayed in a dark-themed terminal with yellow and green highlights for the prompt and section headers. The help text includes usage instructions, command-line options for data, vector, background, stages, buffers, threads, acceptance ratio, cascade parameters, boost parameters, Haar feature parameters, and LBP/HOG feature parameters.

```
avsthiago@NoThiago:~$ opencv_traincascade
Usage: opencv_traincascade
  -data <cascade_dir_name>
  -vec <vec_file_name>
  -bg <background_file_name>
  [-numPos <number_of_positive_samples = 2000>]
  [-numNeg <number_of_negative_samples = 1000>]
  [-numStages <number_of_stages = 20>]
  [-precalcValBufSize <precalculated_vals_buffer_size_in_Mb = 1024>]
  [-precalcIdxBufSize <precalculated_idx_buffer_size_in_Mb = 1024>]
  [-baseFormatSave]
  [-numThreads <max_number_of_threads = 8>]
  [-acceptanceRatioBreakValue <value> = -1]
--cascadeParams--
  [-stageType <BOOST(default)>]
  [-featureType <{HAAR(default), LBP, HOG}>]
  [-w <sampleWidth = 24>]
  [-h <sampleHeight = 24>]
--boostParams--
  [-bt <{DAB, RAB, LB, GAB(default)}>]
  [-minHitRate <min_hit_rate> = 0.995]
  [-maxFalseAlarmRate <max_false_alarm_rate = 0.5>]
  [-weightTrimRate <weight_trim_rate = 0.95>]
  [-maxDepth <max_depth_of_weak_tree = 1>]
  [-maxWeakCount <max_weak_tree_count = 100>]
--haarFeatureParams--
  [-mode <BASIC(default) | CORE | ALL>]
--lbpFeatureParams--
--HOGFeatureParams--
avsthiago@NoThiago:~$
```

Figura 22

Exemplos de uso:

opencv_traincascade -data classifier -vec samples.vec -bg bg.txt -numStages 14 -minHitRate 0.999 -maxFalseAlarmRate 0.5 -numPos 1000 -numNeg 2000 -w 22 -h 30 -mode ALL -precalcValBufSize 1024 -precalcIdxBufSize 1024

Logo após o início do treinamento, será informado no console as opções que foram escolhidas, e em seguida começará o estágio 0 do treinamento **Figura 23**. Para cada estágio é mostrado o número de imagens positivas e negativas que estão sendo utilizadas, o HR que é a porcentagem de acerto e FA que é a quantidade de False Alarms encontrados.

```

PARAMETERS:
cascadeDirName: lbp/
vecFileName: vetosSaida.vec
bgFileName: bg.txt
numPos: 430
numNeg: 900
numStages: 10
precalcValBufSize[Mb] : 1024
precalcIdxBufSize[Mb] : 1024
acceptanceRatioBreakValue : -1
stageType: BOOST
featureType: HAAR
sampleWidth: 13
sampleHeight: 20
boostType: GAB
minHitRate: 0.999
maxFalseAlarmRate: 0.5
weightTrimRate: 0.95
maxDepth: 1
maxWeakCount: 100
mode: ALL

===== TRAINING 0-stage =====
<BEGIN
POS count : consumed    430 : 430
NEG count : acceptanceRatio    900 : 1
Precalculation time: 1
+---+---+---+---+
|  N  |   HR   |   FA   |
+---+---+---+---+
|  1  |    1    |    1    |
+---+---+---+---+
|  2  |    1    |    1    |
+---+---+---+---+
|  3  |    1    |    1    |
+---+---+---+---+
|  4  |    1    |    1    |
+---+---+---+---+
|  5  |    1    | 0.516667|
+---+---+---+---+
|  6  |    1    | 0.523333|
+---+---+---+---+
|  7  |    1    | 0.445556|
+---+---+---+---+

```

Figura 23

4 Visualização dos classificadores cascade gerados com o opencv_visualization:

O OpenCV oferece uma ferramenta chamada `opencv_visualization` para que você possa visualizar seu cascade após treinado, com isso você poderá visualizar quais características ele escolheu no treinamento e quanto complexo os estágios estão. A ferramenta fará uma animação mostrando as características animadas como visto em [23] ou na **Figura 25**.

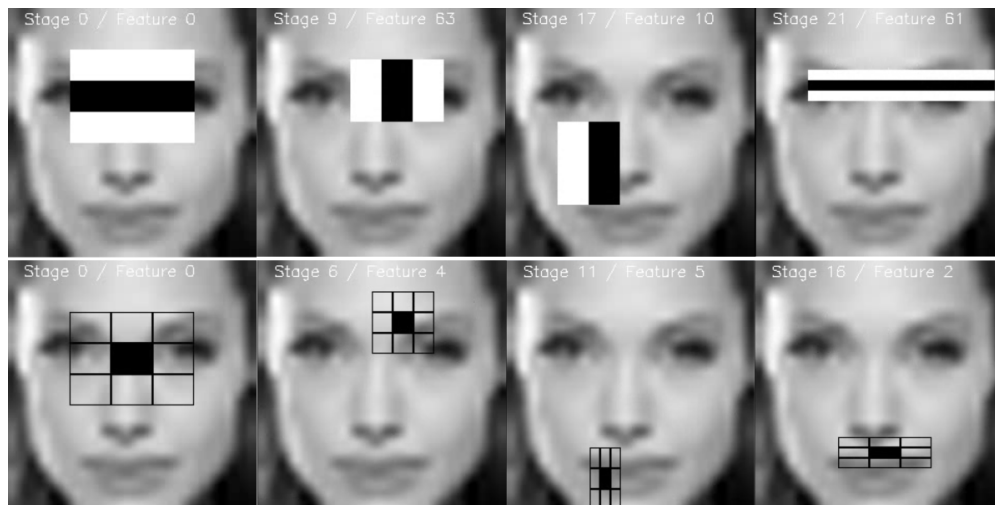


Figura 25

O funcionamento da ferramenta se dá da seguinte forma:

```
opencv_visualization    --image=/data/object.png    --model=/data/cascade.xml  
--data=/data/result/
```

--image

Uma imagem do objeto a ser encontrado, precisa ter o mesmo tamanho que foi utilizado para o treinamento do cascade nos argumentos `-w` e `-h`.

--model

Arquivo de modelo `.xml` que o `opencv_traincascade` gerou. Deve estar no diretório `-data` que foi utilizado no `opencv_traincascade`.

--data (opcional)

Um diretório que será utilizado para salvar um vídeo de saída que a ferramenta irá gerar.

5 Escolhendo os melhores parâmetros para a detecção do objeto

Se tratando de objetos pelo método Haar Cascade, no momento do reconhecimento dois parâmetros são importantes para encontrar o objeto com o mínimo de falsos positivos/negativos. O método do qual eles pertencem é o `detectMultiScale` e os argumentos são:

scaleFactor:

Esse fator de escala fará que em cada iteração, o algoritmo reduza o tamanho da imagem, assim ele pode se tornar invariante a escala. Se o valor dele for por exemplo 1.05, a cada iteração a imagem será reduzida em 5%. Quanto menor esse valor, maior é chance de encontrar o objeto em questão, mas uma quantidade maior de falsos positivos pode ser encontrada, além de necessitar de um maior processamento da máquina. Esse é o motivo de sempre buscar o melhor `scaleFactor`.



Figura 26

minNeighbors:

Esse parâmetro especifica quantos vizinho cada objeto encontrado deve ter para que ele seja retido. A resposta dada para a questão em [24] explica com vários exemplos como esse parâmetro deve ser utilizado.

5.1 O software:

Para que se possa escolher os melhores valores para esses dois parâmetros, foi desenvolvido o **FindSFandMinNeigh.py** (Find Sf and MinNeigh). Ele itera por um conjunto de imagens, e busca em cada uma delas onde o objeto está, logo após compara a

localização de onde ele foi localizado com onde ele deveria ter aparecido utilizando *Intersection Over Union (IoU)* [25]. Esse método retorna um valor de acurácia da predição.

São testadas várias combinações de *scale factor* e *min neighbors*, sendo que para cada combinação é iterado sobre todas as imagens. A escolha da melhor combinação de parâmetros será dada pela maior soma de todos os valores de *IoU* obtidos.

Pode ser visto na **Figura 27** que o software itera por diversas combinações de argumentos e mostra as que tiveram a melhor soma IoU. A última a ser apresentada será a melhor escolha.

```
$ python FindSFandMinNeigh.py --cascade-file bud.xml --list-of-images
IoU/images.lst
imagens carregadas
4.73314885508 1.01 1
5.88503880488 1.01 2
7.2079385907 1.01 3
8.80795153927 1.01 4
9.57211938021 1.02 2
11.7258944855 1.02 3
13.1668356099 1.02 4
14.1168498606 1.03 4
14.7776541796 1.04 4
15.9101164375 1.05 3
16.2761567599 1.05 4
```

Figura 27

5.1.1 Significado do resultado:

16.2761567599 1.05 4

Somado IoU

Scale Factor

Min Neighbors

5.1.2 Exemplo de uso do software:

```
python FindSFandMinNeigh.py --cascade-file bud.xml --list-of-images IoU/images.lst
```

--cascade-file bud.xml

Arquivo .xml criado pelo opencv_traincascade como visto na seção 3.

--list-of-images IoU/images.lst

Um arquivo de texto onde há uma lista de imagens que contenha o objeto e a localização dele **Figura 15**.

Outros argumentos não obrigatórios:

--max-obg-img 10

Número máximo de objetos que as imagens contém. caso a combinação de parâmetros esteja encontrando mais objetos do que deveria, ela é penalizada.

--iwi 5

Iterations Without Improvement. Conforme o valor tanto do minNeighbors quanto do *scaleFactor* são aumentados menos objetos são encontrados, assim sendo, caso haja uma certa quantidade de iterações sem que haja um melhoramento no resultado, o software é interrompido.

--step 1

Passo entre os valores dos argumentos testados. Por padrão esse argumento recebe 1, assim o scale factor muda da seguinte forma: 1.01, 1.02, 1.03... já o minNeighbors incrementa em 1 normalmente.

--visualize

Com esse parâmetro informado, é possível ver cada imagem que é processada. Indicado somente para testes, pois essa tarefa pode ser muito tediosa.

--resize-predict (depreciado)

Ainda é necessário testar um pouco mais esse argumento, pois o seu uso não interferiu muito nos resultados obtidos.

6 Utilizando os classificadores para reconhecer objetos por imagens ou uma Webcam:

Para essa tarefa foi desenvolvido o **ObjectFinder.py** com ele é possível utilizar um conjunto de classificadores de uma vez só, dessa forma, pode se buscar diversos objetos em vídeos ou imagens.

Um exemplo do uso dele é: (O único argumento obrigatório é o **--file-cascade**).

```
python ObjectFinder.py --file-cascade Cascades.csv --angle 0 --max-iou-ratio 0.3
```

--file-cascade Cascades.csv

Esse é um dos argumentos mais importantes, ele é formado por um arquivo que contém uma lista classificadores (Cascade.xml) com seus parâmetros. Uma dica para ele é criar um arquivo csv (Editável pelo excel) **Figura 28**. Pois o software lê os parâmetros separados por vírgula no seguinte formato:

Local Arquivo,Nivel,Tam. Minimo X,Tam. Minimo Y,Min Vizinhos,Fator de Escala,Nome
bud.xml,1,10,10,4,1.08,Budweiser
bohemia.xml,3,10,10,3,1.06,Bohemia
stella.xml,0,10,10,8,1.09,Stella

	A	B	C	D	E	F	G
1	Local Arquivo	Nivel	Tam. Minimo X	Tam. Minimo Y	Min Vizinhos	Fator de Escala	Nome
2	bud.xml	1	10	10	4	1.08	Budweiser
3	bohemia.xml	3	10	10	3	1.06	Bohemia
4	stella.xml	0	10	10	8	1.09	Stella

Figura 28

Explicação sobre as colunas na imagem:

Local arquivo:

Diretório onde o classificador está.

Nivel:

Por padrão informado como 0. Caso dois objetos diferentes sejam encontrados no mesmo local da imagem, aquele com o valor de nível maior será escolhido como o real. (Esse parâmetro foi desenvolvido para testar casos onde primeiro é treinado um reconhecedor de

objeto e tempos depois é treinado um segundo, mas nesse segundo o conjunto de imagens negativas contém o primeiro objeto. Isso faz com que seja mais difícil confundir ele com o primeiro treinado. Dessa forma, se o software encontra ambos em um mesmo lugar o segundo objeto tem uma probabilidade maior de ser o correto.) A máxima porcentagem que os objetos podem se sobrepor é definida pelo argumento **--max-iou-ratio**.

Tam. mínimo (x, y):

Não serão procurados objetos menores que esse tamanho.

Min Vizinhos:

Valor que pode ser calculado assim como visto na seção 5.1.

Fator de Escala:

Valor que pode ser calculado assim como visto na seção 5.1.

Nome:

Nome que será apresentado quando o objeto for identificado na cena.

--angle 0

Quando esse argumento é informado, a imagem será rotacionada 360° e o passo será o valor passado para ele. Dessa forma o algoritmo poderá detectar o objeto em diferentes ângulos. Informando 0 a imagem não é rotacionada.

--max-iou-ratio 0.3

Argumento não obrigatório, mas caso ele seja informado, toda vez que um objeto sobrepor um segundo em mais de 30%(valor do exemplo) somente o com o valor maior de nível descrito no **--file-cascade** irá ser exibido. Caso ambos pertençam a um mesmo classificador ou tenha o valor de nível igual, ambos serão exibidos.

Outros argumentos não presentes no exemplo:

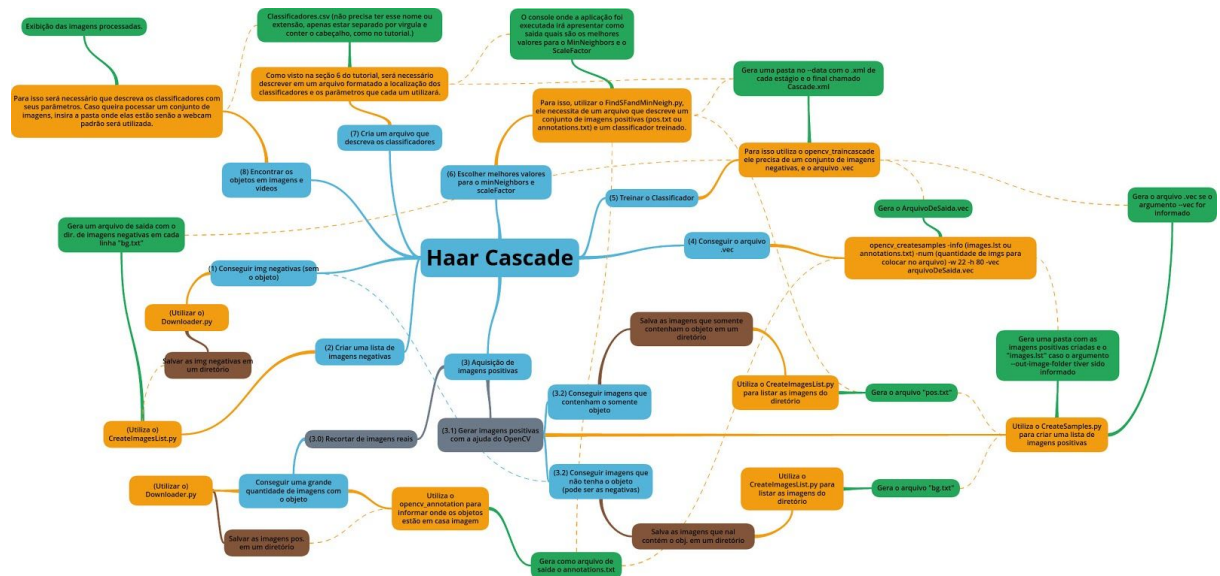
--num-camera 0

As Webcams têm números para sua identificação, variando de 0 até o número de Webcams instaladas -1. O padrão é 0, mas caso queira escolher outra, basta alterar o valor desse argumento.

--images-folder

Caso não queira utilizar uma câmera e sim um conjunto de imagens. Informe nesse argumento uma pasta onde as imagens a serem reconhecidas estão armazenadas. Assim, uma a uma será processada e exibida.

Para uma visualização do processo completo, foi criado esse mindmap:



Pode ser encontrado com uma maior resolução no seguinte link:

https://www.goconqr.com/en-US/p/7303708-Haar-Cascade-mind_maps

REFERÊNCIAS

[1] opencv_traincascade Parameters explanation, image sizes etc.

http://answers.opencv.org/question/39160/opencv_traincascade-parameters-explanation-image-sizes-etc/

[2] Rapid Object Detection Using a Boosted Cascade of Simple Features.

<http://www.merl.com/publications/docs/TR2004-043.pdf>

[3] Documentation for opencv_annotation.

http://answers.opencv.org/question/75083/documentation-for-opencv_annotation/

[4] Positive Samples.

http://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html#positive-samples

[5] Vlc export frames.

<https://www.isimonbrown.co.uk/vlc-export-frames/>

[6] Creating training set as a collection of PNG images.

http://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html#creating-training-set-as-a-collection-of-png-images

[7] Github: opencv-haar-classifier-training.

<https://github.com/mrnugget/opencv-haar-classifier-training>

[8] Github: FastAnnotationTool.

<https://github.com/christopher5106/FastAnnotationTool>

[9] Image annotation tool with image masks.

https://lear.inrialpes.fr/people/klaeser/software_image_annotation

[10] Creating a Cascade of Haar-Like Classifiers: Step by Step.

https://www.cs.auckland.ac.nz/~m.rezaei/Tutorials/Creating_a_Cascade_of_Haar-Like_Classifiers_Step_by_Step.pdf

[11] Label Images for Classification Model Training.

<https://www.mathworks.com/help/vision/ug/label-images-for-classification-model-training.html>

!

[12] Cascade Classifier Training.

http://docs.opencv.org/master/d8/d88/tutorial_traincascade.html

[13] Video: VIOLA JONES FACE DETECTION EXPLAINED

https://www.youtube.com/watch?v=_QZLbR67fUU

[14] Video melhor: Viola Jones face detection and tracking explained.

<https://www.youtube.com/watch?v=WfdYYNamHZ8>

[15] Exemplos Haar Cascade Github.

<https://github.com/opencv/opencv/tree/master/data/haarcascades>

[16] Motivo de selecionar menos positivos no treinamento do haar.

<http://answers.opencv.org/question/4368/traincascade-error-bad-argument-can-not-get-new-positive-sample-the-most-possible-reason-is-insufficient-count-of-samples-in-given-vec-file/#4474>

[17] Erro causado pela falta de samples no arquivo vec.

<http://answers.opencv.org/question/776/error-in-parameter-of-traincascade/?answer=792#post-id-792>

[18] Video: HAAR face model visualization based on OpenCV 2.4

<https://www.youtube.com/watch?v=zLBAJ93-AEQ&spfreload=5>

[19] Outras referências.

http://docs.opencv.org/master/d0/de3/citelist.html#CITEREF_Viola04

[20] Vídeo explicação Adaboost.

https://youtu.be/_QZLbR67fUU?t=13m33s

[21] Robust Real-Time Face Detection.

<https://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf>

[22] An Extended Set of Haar-like Features for Rapid Object Detection.

<https://pdfs.semanticscholar.org/72e0/8cf12730135c5ccd7234036e04536218b6c1.pdf>

[23] Video HAAR face model visualization based on OpenCV 2.4

<https://www.youtube.com/watch?v=zLBAJ93-AEQ&spfreload=5>

Marca	Corretos	Corretos (%)	Incorretos
Budweiser	366	73%	361
Bohemia	301	60%	289
Devassa	348	69.6%	458
Heineken	315	63%	223
Stella Artois	308	61.6%	286

asd ads			
------------	--	--	--

[24] OpenCV detectMultiScale() minNeighbors parameter.

<http://stackoverflow.com/questions/22249579/opencv-detectmultiscale-minneighbors-parameter>

[25] Intersection over Union (IoU) for object detection.

<http://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>

