**Differentiate between documents, photos and rx images**

OK we start easy with image recognition, we have a bunch of images which can be either colour photos, rx images or scanned documents. The aim of this project is to classify them into three classes.

Since the three classes are well separated we opted for a model which uses features extracted from images. Other methods based e.g. on Convolutional Neural Network (CNN) would work but we thought it could have solved with a simpler method.

Since our model works on the extracted features from images, the choice of the features is key to have a good model.

We selected this set of features …

**Recognise car parts**

Another good project we are dealing with is the recognition of car parts in a set of photos of cars.

Nowadays more and more DBs of various classes of images are becoming available (e.g. ImageNet) and many people decide to use huge datasets and train sophisticated CNN models.

To solve this specific issue we tried to use a softer approach. So we asked ourself the question: is it possible to extract features from this kind of photos?

Our answer is a firm YES!

There are a number of different algorithms used in the past at the time when training sets were not so big and computation power was an issue. SIFT, SURF and HOG descriptors are maybe the most famous.

We will consider here the HOG descriptor.

######### beginning code

xpix = 128

ypix = 128

cell\_size = 16

num\_ori = 20

block\_size = 1

df\_wheel\_macro = pd.DataFrame(columns = ['HOG','type'])

nimg = 0

file = file\_0

folder = folder\_0

scaling = [.8,.9,1.]

mode\_s = [['q','c'],['q','c'],['c']] # mode\_scaling, q: quadrants, c: centre

#mode\_s = [['c'],['c'],['c']] # mode\_scaling, q: quadrants, c: centre

naug = [0]\*len(scaling)

for imode in range(0,len(mode\_s)):

if ('q' in mode\_s[imode]):

naug[imode] += 4

if ('c' in mode\_s[imode]):

naug[imode] += 1

naug = sum(naug)\*2. # number of images (augmentation) for each image (with h. flipping)

for ifil in range(0,len(file)):

img = Image.open(folder+os.sep+file[ifil])

img = img.convert('RGB')

img = img.convert('L')

imgm = ImageOps.mirror(img)

imgwidth, imgheight = img.size

for iscale in range(0,len(scaling)):

width = int(imgwidth \* scaling[iscale])

height = int(imgheight \* scaling[iscale])

for imode in range(0,len(mode\_s[iscale])):

for iquad in range(0,4):

if (mode\_s[iscale][imode] == 'c'):

if (iquad > 0):

continue

x0 = int((imgwidth - width)/2.0)

y0 = int((imgheight - height)/2.0)

else:

x0 = int(imgwidth - width)\*(iquad%2) # x0 = 0 if iquad in [0,2]

y0 = int(imgheight - height)\*int(iquad/2.) # y0 = 0 if iquad in [0,1]

box = (x0, y0, x0+width, y0+height)

subimg = img.crop(box)

subimgm = imgm.crop(box)

subimg = subimg.resize((xpix,ypix), PIL.Image.ANTIALIAS)

subimg = np.array(subimg)

roi\_hog\_fd = hog(subimg, orientations=num\_ori, pixels\_per\_cell=(cell\_size, cell\_size),

cells\_per\_block=(block\_size,block\_size))

df\_wheel\_macro.loc[nimg, 'HOG'] = list(roi\_hog\_fd)

df\_wheel\_macro.loc[nimg, 'type'] = 'wheel'

nimg += 1

subimgm = subimgm.resize((xpix,ypix), PIL.Image.ANTIALIAS)

subimgm = np.array(subimgm)

roi\_hog\_fd = hog(subimgm, orientations=num\_ori, pixels\_per\_cell=(cell\_size, cell\_size),

cells\_per\_block=(block\_size,block\_size))

df\_wheel\_macro.loc[nimg, 'HOG'] = list(roi\_hog\_fd)

df\_wheel\_macro.loc[nimg, 'type'] = 'wheel'

nimg += 1

if ((int(nimg/naug))%50 == 0):

print('I have analysed ',(int(nimg/naug)), ' images out of ',len(file))

######### end code

The HOG descriptor devides the input image into different cells (in this case of size 16x16 pixels.

In each of these cells the HOG computes the gradient at different angles.

To do so the image is first resampled to the size of 128x128 pixels.