# Assignment2

Yifan Li R0733300

## Question

### Q1: Now that we have some results, is the given similarity function a good metric to quantify the distance between two fingerprints? Is it reliable enough to incriminate a suspect? What are its limitations?

No. This similarity is not a good metric to incriminate a suspect. This similarity calculate the difference of all pixels. However, the pixel would change a lot even for a small shift, zoom or rotation. It is only useful when all the fingerprints have same positions and angle.

### Q2: Are all the key points matches accurate? Are they expected to be? Explain why.

Not all the key points matches are accurate. They  are not expected to all match to each other. Brutal force matcher will use hamming distance to find a best match for each point. But the distance is calculated using a local descriptor. Even though `crosscheck` is turned on, two different points will still match as long as the areas around them are similar. They might be at different position, but the descriptor might be similar. And all best matches will be returned.

### Q3: Choose a global feature similarity function, (e.g. you can start from euclidean distance between the reduced sets of KeyPoints and count the values above a threshold).

The similarity is function is acquired by calculating the number of image pairs of which the distance between the reduced sets of KeyPoints is less than a threshold.

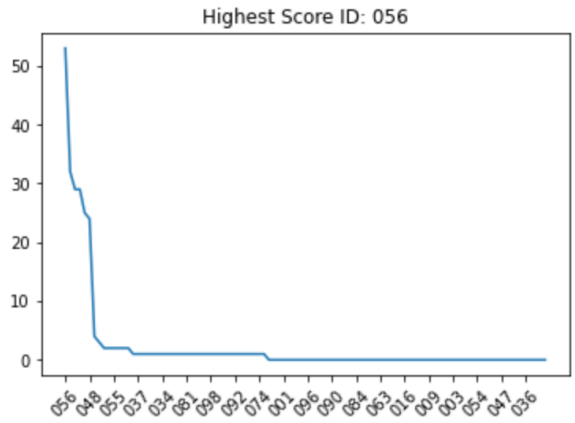
The code is shown below:

```
def global_img_similarity(matches, reg_kp1, kp2, threshold = 1):
    kp1_reduced,kp2_reduced = get_reduced_set(matches, reg_kp1, kp2)
    distances = np.sum(np.square(np.array(kp1_reduced) - np.array(kp2_reduced)), axis=
```

```
1)
    number = (distances < threshold).sum()
    return number
```

## Q4: Visualize the scores and determine a score threshold to discriminate the matching fingerprints. Explain how you determine the threshold.

We calculated the similarity of images by calculating the number of keypoints of which the distance is less than 1. The distance graph is drawn below:




The first 7 scores are 53, 32, 29, 29, 25, 24, 4. From the 6th point, the distance drop hugely from 24 to 4. The gap is huge. And distance of 4 means that there are only 4 point that have distance less than 1. It is fair to think the fingerprint doesn't match to the suspect. So the threshold should be around 20 between the 6th and 7th point.

## Q5: Choose a hybrid feature similarity function that makes use of both the geometric distance and the feature distance of the keypoints. Using this hybrid function, visualize and assess the matches.

We simply multiply local distance and global distance to get the hybrid distance.

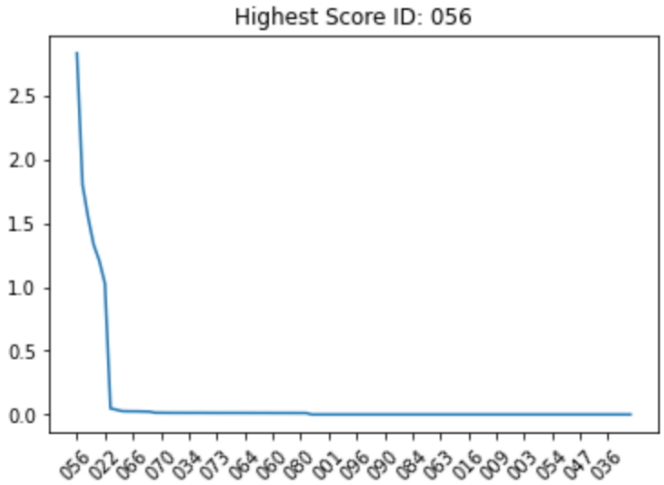
The local similarity is 1 divide by the top N distance between matching points.

The code of local similarity and hybrid similarity is shown below:

```
def local_img_similarity(matches, N = 10):
    local_pt_distance = np.sort(np.array([(match.distance) for match in matches]).T)
    topN_MeanDistance = np.mean(local_pt_distance[:N])
    return 1/topN_MeanDistance

def hybrid_img_similarity(matches, kp1_reg, kp2):
    localDistance = local_img_similarity(matches)
    globalDistance = global_img_similarity(matches, kp1_reg, kp2)
    return localDistance * globalDistance
```

The similarity graph is shown below:




The scores of first 7 scores are 2.83, 1.80, 1.54, 1.33, 1.21, 1.03, 0.05. It is easy to see that there is a huge gap between the 6th and 7th points. And by plotting the first 7th images, we can find that the first 6 fingerprints are identical. There are only some shifts and scaling. But the hybrid similarity is still robust to these changes. It proves this method is effective for verification.

## Q6: Check out iris_perpetrator.png. Where do you see difficulties? What kind of similarity measures do you expect to work best?

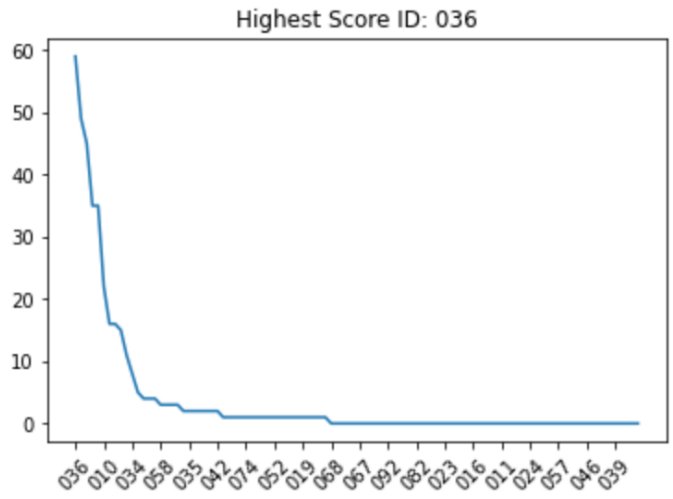
The eyelid will cover part of the iris. It will cover part of the features and make the segmentation more difficult. The eyelid and eyelash will generate noise feature. The illumination will interfere the feature extraction. The iris will also move. Therefore global feature should work better as it can better overcome the iris shift. It can also decrease the effect of eyelash.

If we stick to keypoint match as fingerprint, then we can use euclidean distance to calculate the similarity.

## Q7: Construct a similarity table with the iris images. Use local OR global matches in order to get scores. Use the scores you get to determine the perpetrator.

Global similarity is used.

The similarity graph is shown below:

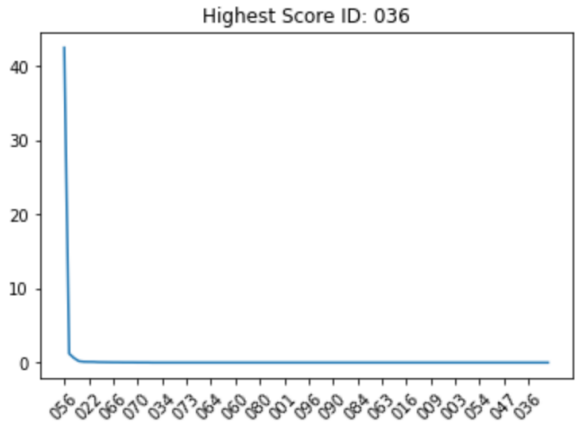



The first 12 scores are 59, 49, 45, 35, 35, 22, 16, 16, 15, 11, 8, 5. After plotting the first 12 iris, we find the first 11 iris are the same. So the perpetrator is inside the first 11 scores.

## Q8: Fuse your iris and fingerprint biometric system on the score level to solve the murder case! Do you feel confident in your prediction? How do you fuse the scores? Why?

Since the score of fingerprint varies between 2.8 to 0 and iris scores varies between 59 and 0, we can multiply the two scores to get a more accurate score. Besides, most non matchers have a score of 0, so multiply them will not increase their scores.

The first 5 score is 42.5,1.21, 0.61, 0.19, 0.11. It is very clear that the first match is much better than other in both level. So it is fair to assume the first match which corresponds to index 56 is the suspect.




If we look Iris and fingerprint separately, 56 exists at both the identical matches. If we assume the fingerprint and iris are both modified, then 56 is the only one that has been changed.  We are confident about it.

# Tasks

## 6. Iris Segmentation Using MRCNN

### 6.1 Introduction

In the original method, the iris is treated as a perfect circle. But in some situation, the eyelid would cover part of the iris and thus make it imperfect circle. The eyelid could become noise at later step.

In order to only segment the part of iris that is not covered by eyelid , MRCNN method is used. Compared with original method, it can segment irregular shape. Therefore it can exclude the influence of eyelid on iris.

### 6.2 MRCNN Principle

MRCNN is a combination of faster-RCNN and FCN. Faster-CNN is used for object classification and detection. FCN is then used to segment the object inside the box range.

CNN has been widely used for image classification. It can detect the feature of the image. However, due to convolution and pooling, the spatial information of image is lost which will cause problem for image segmentation. R-CNN(regions with CNN features) can more specifically point out the position of the classified object. The network first does region proposals search and compute the CNN features inside the region. Then it use classifiers such as SVM to classify the object inside the region. However, region proposal is the speed bottleneck of R-CNN. In order to increase the

speed of the network, Fast R-CNN first extract the features and then do region proposal search. Based on fast R-CNN, faster R-CNN went one step further. It combine the region proposal and CNN feature extraction into one step.

After using faster R-CNN to get the box outside the object, FCN(Fully Convolutional Network) is used to segment the object inside the box. It will decide each pixel inside the box belongs to the object or not.

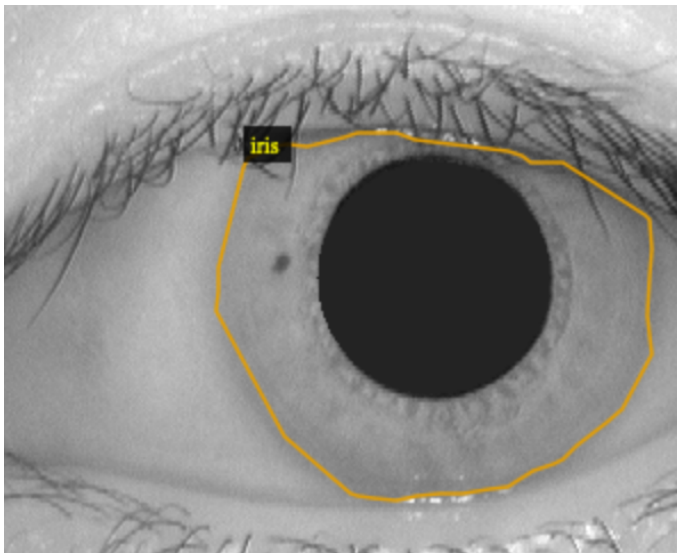
### 6.3 MRCNN on Iris Segmentation

Since the CASIA1 dataset already filled the area of pupil into black, we only need to detect the contour of iris. Pupil can be get rid of by using simple thresholding.

The whole process of iris segmentation include following step:

- Iris labeling and annotation

  VGG Image annotator is used for iris annotation. Only one class(Iris) need to be labelled. The labelled image is shown below:

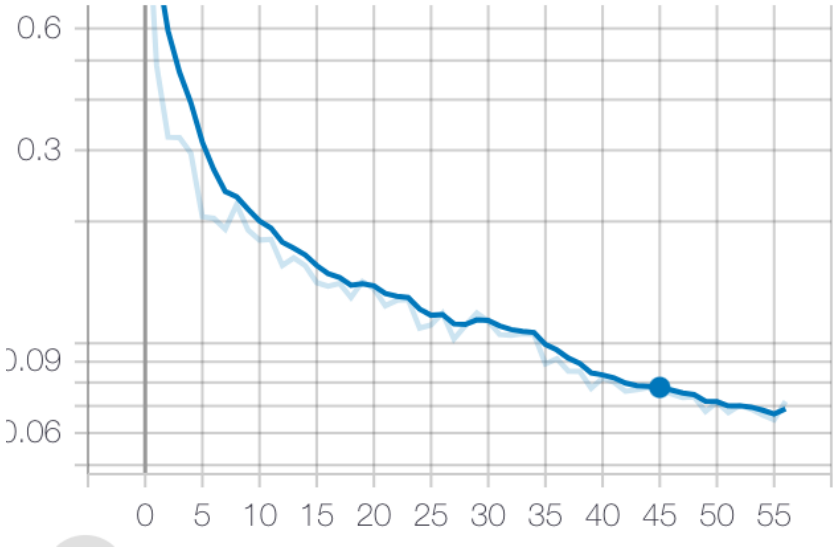



- Mask making

  Besides original image, MRCNN also requires the mask image. After getting the annotation file, `skimage.draw.polygon ` function is used to draw the mask image.

- Training

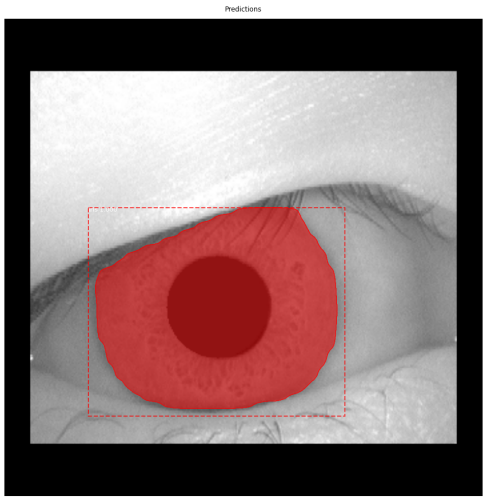
  The whole 100 images into two parts: 90 for training and 10 for validation. Pre-trained weights is loaded. Only the head layers are training in order to save time. After around 60 epochs, the training loss decrease from 1.54 to 0.07.

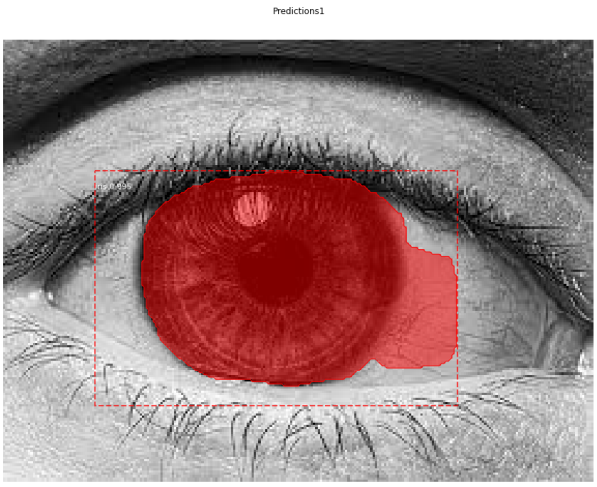



- Validation and testing

  One image from the CASIA1 dataset and one image from internet  is used for testing the training result.

  As is shown in the graph, the model can segment the iris of CAISA dataset fairly well. It can segment only the iris and avoid including eyelid into the segmentation. However, due to the limitation of training image, it can not segment random eye image well.




The code is saved at the MRCNN notebook.

# 7. Fingerprint feature extraction using HOG

## 7.1 HOG Descriptor Introduction

Hog descriptor is a method which extracts the gradient feature of an image. It can use gradient difference to detect the shape of object. The pixel gradient around the pixel will then be used as descriptor. In the situation of fingerprint, the ridges and valleys will cause a lot of sharp gradient. HOG descriptor is able to detect the both the gradient amplitude and direction. Thus it can detect the ridges and valleys. It will then save the gradient information to a histogram and then to a vector. By comparing the difference of two vectors, two image can be compared. By using HOG descriptor, the global edge feature can be directly extracted and compared.

The step of using HOG descriptor is shown below:

- **Gradient calculation**

  The gradient of an image can be calculated using sober detector. Sobel detector can calculate the gradient on both X and Y direction. The gradient amplitude can be calculated by dividing gradient on Y direction by X direction.

- **Histogram of gradient calculation**

  Then the image is divided into 8*8 cells. In each cell, the histogram of gradient will be calculated. Each bin has a range of 20 degree and thus the histogram has 9 bins(0-180 degree).

- **Block normalization**

  The gradient of image is strongly influenced by brightness. In order to make HOG robust to lighting, normalization is necessary. The normalization step will take 2*2 cells. L2 normalization is performed to shrink all values of the histogram to 0 and 1.

- **Calculate the Histogram of Oriented Gradients feature vector**

  The final feature vector is a result of concatenating vectors on all cells into a single vector. The single vector can represent the whole image and then used for comparison.

- **Feature Distance**

  The final single vector has very high dimension. This kind of vector can be compared using Cosine Distance. The formula is shown below:

  The larger the distance is, the more similar two images are.

### 7.2 Result

The hog descriptor can be easily used by importing `skimage.featuer` library. The code is shown below

```
def hog_transform(imgs):

imgSet = []

fdSet = []

for i in imgs:

fd, hogImage = hog(i, orientations=9, pixels_per_cell=(8, 8),

cells_per_block=(2, 2), visualize=True, feature_vector= True)

imgSet.append(hogImage)

fdSet.append(fd)

return imgSet, fdSet
```

The hog transformed image is shown below:

The highest 6 scores are 76, 48, 22, 49, 18, 56 which is the same as using hybrid feature and corresponds to the hacked fingerprint. It functions as well as the hybrid distance.

The code is saved at the end of assignment notebook.