Université Paris Saclay

Common Trunk - course 5

Image and signal processing

Face identification

Authors:
Issa Hammoud
Salim Tabarani

Supervisor:
Prof. Yohann Tendero

January 6, 2019

Contents

List of Figures

Introduction

Deep Learning is heavily used in computer vision nowadays, due to the fact that it performs well on these type of problems.However,these algorithms require millions of samples/images for training which are not always at our disposition.
So to deal with the lack of data issue,we want to compare one shot learning technique with classical approaches for face identification problem. For that,we will look into two techniques: Siamese Networks and Scale Invariant Feature Transformation (SIFT). We will compare their robustness on different kind of identification problems like change in contrast, rotation and brightness.
This report is divided as follow: in the first chapter, we will explain how SIFT works in details, then the proposed method for preprocessing in the literature, and what are the matching strategies in use.
In the second chapter, we will introduce deep learning approach for face identification, Siamese networks, and how it works. Finally, we present our dataset and the results of both methods.

1 Scale Invariant Feature Transformation(SIFT)

1.1 Introduction

Face identification is possibly one of the first cognitive processes used by humans to recognize familiar people. It is certainly a complex problem, but essentially can be reduced to a pattern classification task.[3]

SIFT is a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. The features are invariant to image scale and rotation, and are shown to provide robust matching across a a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive, in the sense that a single feature can be correctly matched with high probability against a large database of features from many images.[2]

The algorithm is divided into four sections:

- Scale-space extrema detection.
- Keypoint localization
- Orientation assignment.
- Keypoint descriptor.

1.2 Scale-space extrema detection

The first stage of keypoint detection is to identify locations and scales that can be repeatably assigned under differing views of the same object. Detecting locations that are invariant to scale change of the image can be accomplished by searching for stable features across all possible scales, using a continuous function of scale known as scale space.[2]

The fact of smoothing an image at all possible scales s > 0 leads to what is called a scale-space.

It may seem that there are enumerous ways to construct a scale-space. But surprisingly in case we set some simple requirements there is only one scale-space possible construction.

We are looking for a family of image operators L^s where s refers to the scale such that given the 'image at zero scale' f^0 we can construct the family of images $L^s f^0$

such that:

- L^s is a linear operator.
- L^s is translation invariant (all parts of the image are treated equally, i.e. there is no a priori preferred location).
- L^s is rotational invariant (all directions are treated equally, i.e. there is no a priori preferred direction).
- L^s is separable by dimension.
- L^s is invariant under change of scale (starting with a scaled version of the zero scale image we should in principle obtain the same scale-space but then offsetted in scale).

It can be proved that the unique scale space operator satisfying all the requirements is the Gaussian convolution, in our case the 2D Gaussian filter[6]:

$$G(x, y, s) = \frac{1}{2s^2} \times \exp\frac{-(x^2 + y^2)}{2s^2}$$

Therefore, the scale space of an image is defined as a function, $L(x, y, s)$, that is produced from the convolution of a variable-scale Gaussian, $G(x, y, s)$, with an input image $I(x, y)$:

$$L(x, y, s) = G(x, y, s) * I(x, y)$$

To efficiently detect stable keypoint locations in scale space, they have proposed using scale-space extrema in the difference-of-Gaussian function convolved with the image, $D(x, y, s)$, which can be computed from the difference of two nearby scales separated by a constant multiplicative factor k:

$$D(x, y, s) = (G(x, y, ks) - G(x, y, s)) * I(x, y) = L(x, y, ks) - L(x, y, s)$$

There are a number of reasons for choosing this function. First, it is a particularly efficient function to compute, as the smoothed images, L, need to be computed in any case for scale space feature description, and D can therefore be computed by simple image subtraction.

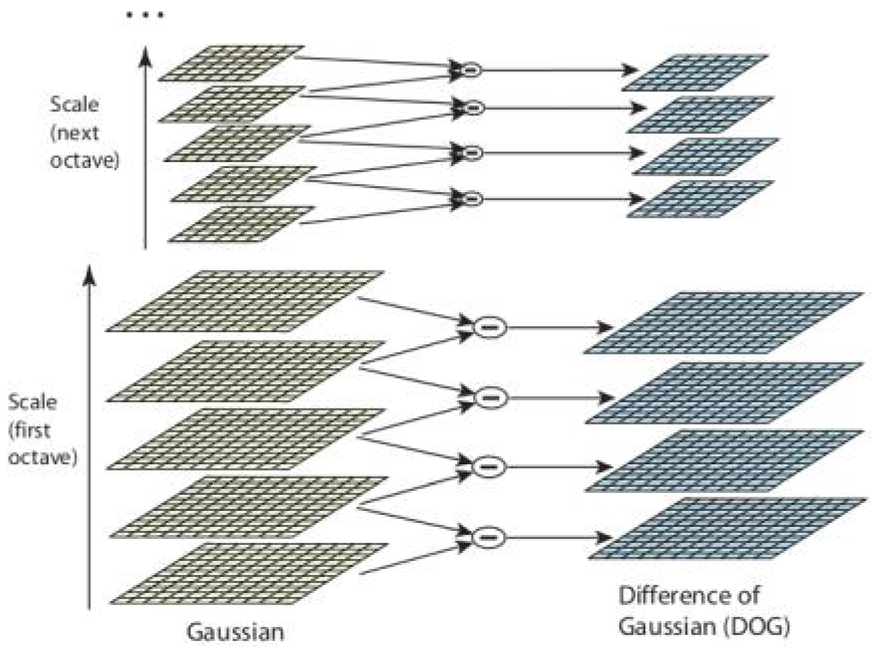

Figure 1.1: Scale-space and Difference of Gaussian

In addition, the difference-of-Gaussian function provides a close approximation to the scale-normalized Laplacian of Gaussian, $s^2\nabla^2 G$, found that its maxima and minima produce the most stable image features compared to a range of other possible image functions. The relationship between D and $s^2\nabla^2 G$ can be understood from the heat diffusion equation and the finite difference approximation to the derivative:

$$\frac{\partial G}{\partial s} = s\nabla^2 G \approx \frac{G(x, y, ks) - G(x, y, s)}{ks - s}$$

and therefore: $G(x, y, ks) - G(x, y, s) \approx (k - 1)s^2\nabla^2 G$.

The factor (k − 1) in the equation is a constant over all scales and therefore does not influence extrema location.

The figure above shows how to calculate the Difference of Gaussian (DoG). An octave is set of blurred images in way to double the scale, e.g if we want 3 images per octave and the first image is blurred with scale s, then the second one will be blurred with scale $\sqrt{2}s$, and the last one with $2s$.

Once a complete octave has been processed, we resample the Gaussian image that has twice the initial value of s by taking every second pixel in each row and column.

Then, in order to detect the local maxima and minima of $D(x, y, s)$, each sample point is compared to its eight neighbors in the current image and nine neighbors in the scale above and below. It is selected only if it is larger or smaller than all of these neighbors.[2]

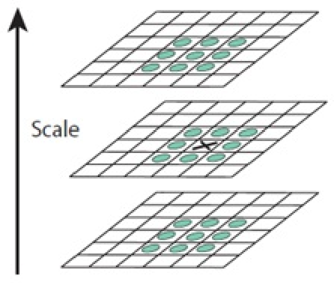

Figure 1.2: Extrema detection

However, a developed method for fitting a 3D quadratic function to the local sample points to determine the interpolated location of the maximum, showed a substantial improvement to matching and stability. His approach uses the Taylor expansion of the scale-space function, $D(x, y, s)$, shifted so that the origin is at the sample point. We denote the new extremum by $\hat{x}$.

1.3 Keypoint localization

Key points generated in the previous step produce a lot of key points. Some of them lie along an edge, or they don't have enough contrast (therefore sensitive to noise). In both cases, they are not useful as features.[6]

For low contrast features, we simply check their intensities.(in the original paper, the author rejected all extrema with a value of $|D(\hat{x})|$ less than 0.03 with image pixels between 0 and 1).

Moreover, it is not sufficient to reject only keypoints with low contrast. The difference-of-Gaussian function will have a strong response along edges, even if the location along the edge is poorly determined and therefore unstable to small amounts of noise.

A poorly defined peak in the difference-of-Gaussian function will have a large principal curvature across the edge but a small one in the perpendicular direction. The principal curvatures can be computed from a 2x2 Hessian matrix, H, at the location and scale of the keypoint, and its eigenvalues are proportional to them.[2]

However, we are only interested by the eigenvalues ratio, and we can prove easily that the trace's square divided by the determinant of H only depends on the ratio of the eigenvalues.

In the original paper, the author eliminates key points that have a ratio between the principal curvatures greater than 10.

1.4 Orientation assignment

By assigning a consistent orientation to each keypoint based on local image properties, the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation. The scale of the keypoint is used to select the Gaussian smoothed image, L, with the closest scale, so that all computations are performed in a scale-invariant manner. For each image sample, $L(x, y)$, at this scale, the gradient magnitude, m(x, y), and orientation, $\theta(x, y)$, is precomputed using pixel differences:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$
$$\theta(x,y) = tan^{-1}\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}$$

An orientation histogram is formed from the gradient orientations of sample points within a region around the keypoint. The orientation histogram has 36 bins covering the 360 degree range of orientations. Each sample added to the histogram is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a that is 1.5 times that of the scale of the keypoint. Peaks in the orientation histogram correspond to dominant directions of local gradients. The highest peak in the histogram is detected, and then any other local peak that is within 80 percent of the highest peak is used to also create a keypoint with that orientation. Therefore, for locations with multiple peaks of similar magnitude, there will be multiple key points created at the same location and scale but different orientations.

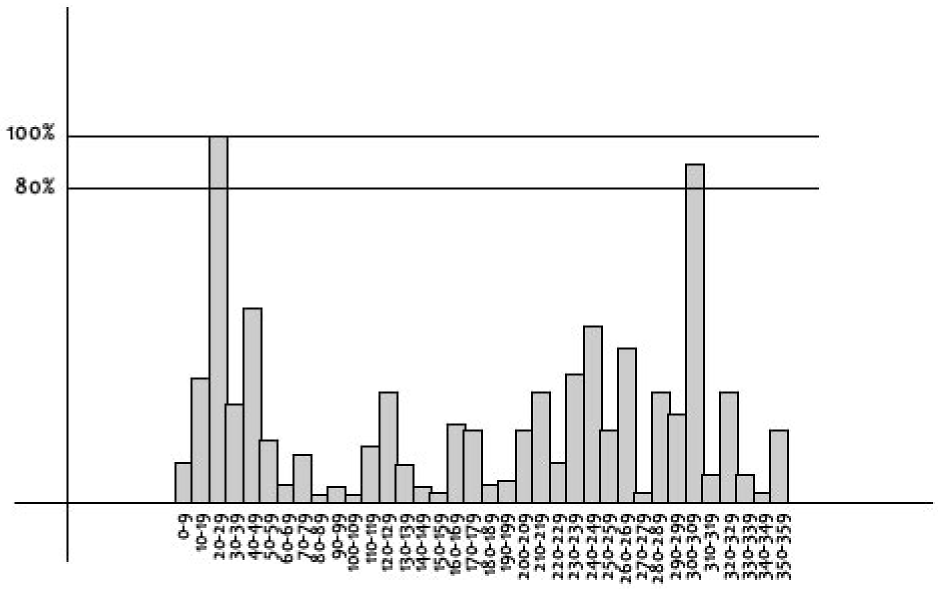

Figure 1.3: Orientation histogram

Finally, a parabola is fit to the 3 histogram values closest to each peak to interpolate the peak position for better accuracy.[2]

1.5 Key descriptors

The previous operations have assigned an image location, scale, and orientation to each keypoint. The next step is to compute a descriptor for the local image region that is highly distinctive to remaining variations, such as change in illumination or 3D viewpoint.

First the image gradient magnitudes and orientations are sampled around the keypoint location, using the scale of the keypoint to select the level of Gaussian blur for the image. In order to achieve orientation invariance, the coordinates of the descriptor and the gradient orientations are rotated relative to the keypoint orientation. A Gaussian weighting function with s equal to one half the width of the descriptor window is used to assign a weight to the magnitude of each sample point.

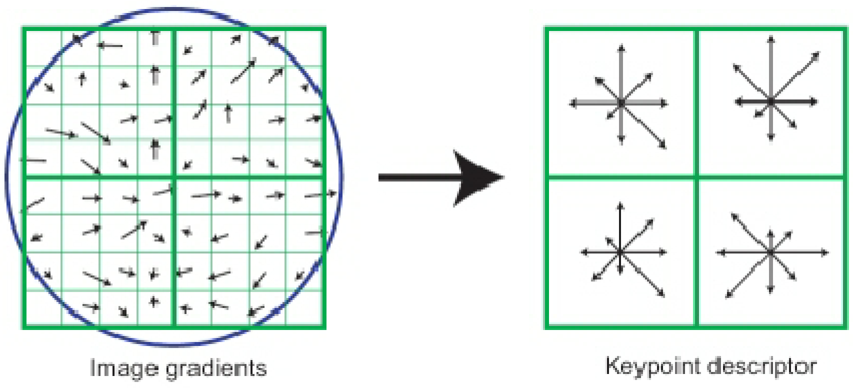

Figure 1.4: Keypoint descriptor

The purpose of this Gaussian window is to avoid sudden changes in the descriptor with small changes in the position of the window, and to give less emphasis to gradients that are far from the center of the descriptor, as these are most affected by misregistration errors.

The key point descriptor in the figure shows eight directions for each orientation histogram, with the length of each arrow corresponding to the magnitude of that histogram entry.

The descriptor is formed from a vector containing the values of all the orientation histogram entries, corresponding to the lengths of the arrows on the right side of the figure. It shows a 2×2 array of orientation histograms, whereas the experiments showed that the best results are achieved with a 4×04 array of histograms with 8 orientation bins in each. Therefore, the experiments in the original paper use a $4 \times 4 \times 8 = 128$ element feature vector for each keypoint.[2]

1.6 Preprocessing

The proposed method is CLAHE (Contrast Limited Adaptive Histogram Equalization). It aims to add a preprocessing stage to the traditional methodology of the SIFT algorithm, in order to improve its performance in adverse illumination conditions.

CLAHE is used to enhance the contrast of an image by changing values in the intensity, it operates in small areas, called tiles using bilinear interpolation to eliminate the boundaries of the region, so the small neighboring areas are smoothed.

It helps to dramatically reduce image noise and prevents saturation of brightness that can happen when performing a traditional histogram equalization.[4] The image is divided into small blocks called ”tiles”. Then each of these blocks are histogram equalized as usual. So in a small area, histogram would confine to a small region (unless there is noise). If noise is there, it will be amplified. To avoid this, contrast limiting is applied. If any histogram bin is above the specified contrast limit, those pixels are clipped and distributed uniformly to other bins before applying histogram equalization. After equalization, to remove artifacts in tile borders, bilinear interpolation is applied.[1]

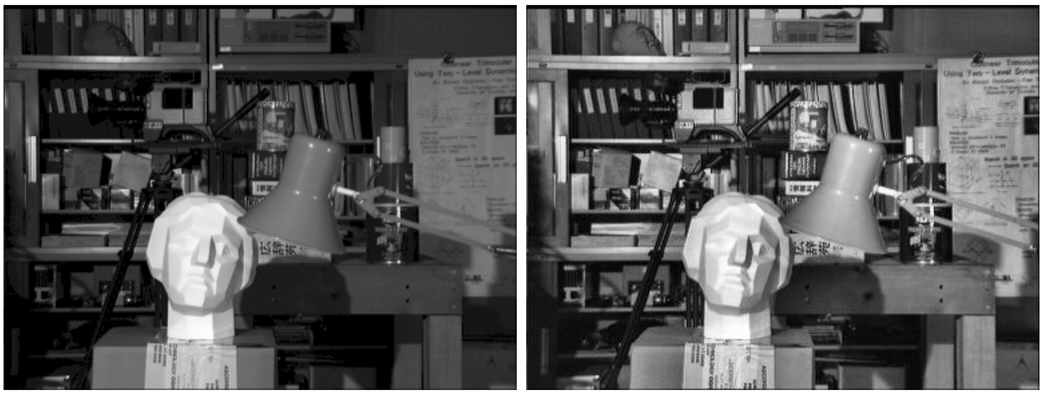

Figure 1.5: Before(left) and after(right) CLAHE preprocessing

1.7 Matching strategies

To authenticate a face, the SIFT features computed in the test image should be matched with the SIFT features of the template (images in the database).
In this section different matching methodologies are investigated:

1.7.1 Minimum pair distance(MPD)

This methodology is the simplest one: computing the distance between all pairs of keypoint descriptors in the two images and use as matching score the minimum distance. In this work the simple Euclidean Distance has been investigated, even if more complicated ones could be employed. The main idea under this method is that there is one point of the face which contains a very distinctive feature of the subject, which could be found in the testing image.

1.7.2 Matching eyes and mouth(EM)

This second method takes into account the fact that most part of the face information is located around the eyes and the mouth. Once the position of these landmarks is determined, a matching strategy can be driven to consider only SIFT features belonging to such image areas, neglecting other less informative points. Different techniques have been proposed for determining the position of eyes and mouth, here we assume that such positions are known. Given an image I, two sub images are extracted: one located around the eyes and one located around the mouth, Then the matching is performed in a pair-wise manner (using MDP). Finally the two distances are averaged.

1.7.3 Matching on a regular grid(RG)

The matching methodology presented in this paragraph subdivides the images in different sub-images, using a regular grid with overlapping. The matching between two images is then performed by computing distances between all pairs of corresponding sub-images, and finally averaging them. After a preliminary experimental evaluation done by the author of the paper[3] he found that sub-images of dimensions 1/4 and 1/2 of width and height, respectively, represent a good compromise between accuracy of localization and possibility of recovering from registration errors. The overlapping was set to 25 percent.[3]
In the results' section, we will only use MPD method to compare (RG is not suitable for our database).

2 Siamese Networks

2.1 Introduction

Siamese networks (Bromley, Jane, et al. "Signature verification using a" siamese" time delay neural network." Advances in neural information processing systems. 1994.) are neural networks containing two or more identical subnetwork components. A siamese network may look like this:

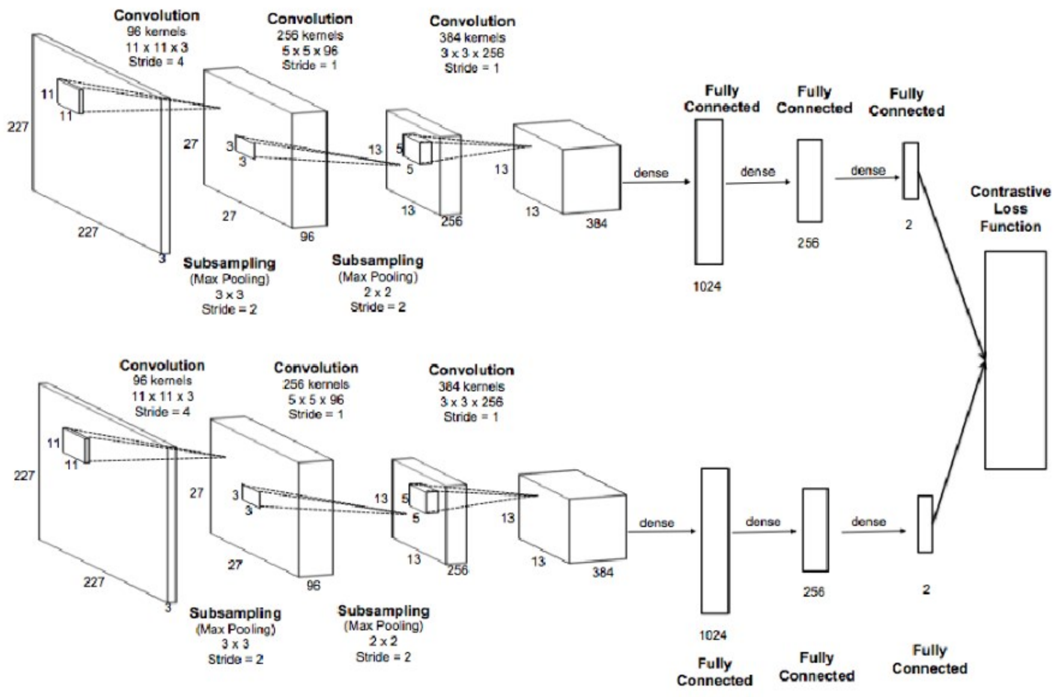

Figure 2.1: Siamese network architecture

It is important that not only the architecture of the subnetworks is identical, but the weights have to be shared among them as well for the network to be called "siamese". The main idea behind siamese networks is that they can learn useful data descriptors that can be further used to compare between the inputs of the respective subnetworks. Hereby, inputs can be anything from numerical data (in this case the subnetworks are usually formed by fully-connected layers), image data (with CNNs as subnetworks) or even sequential data such as sentences or time signals (with RNNs as subnetworks).

2.2 One shot Learning

Siamese networks have wide-ranging applications.In our case,we will be interested in one shot learning which consists in presenting a new training dataset is to the trained (classification) network, with only one sample per class. Afterwards, the classification performance on this new dataset is tested on a separate testing dataset. As siamese networks first learn discriminative features for a large specific dataset, they can be used to generalize this knowledge to entirely new classes and distributions as well.

2.3 Application

2.3.1 Model

we will apply transfer learning and use pre-trained weights of VGG Face model. Even though, imagenet version of VGG is almost same with VGG Face model, researchers feed dedicated training-set images to tune weights for face recognition.Also,we will consume the model as auto-encoder to represent images as vectors.

The structure of the VGG-Face model can be described as following:

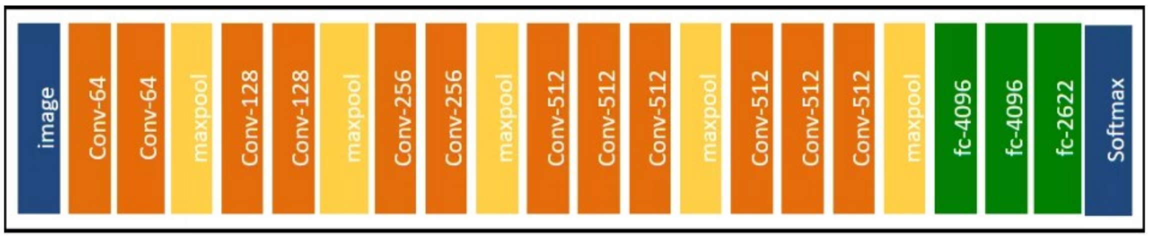

Figure 2.2: VGG-Face model

So,by using Keras,we can construct the model and load the pretrained weights.

2.3.2 Image Representation

The VGG model used expects 224x224x3 sized input images where the third dimension refers to number of channels. So the image representation which will be the output of the model will be a 2622 dimensional vector

2.3.3 Measure

We've seen that the used model enables us to represent input images as vectors.So now in order to compare two pictures,we need to compare their vector representations which can be achieved by computing the distance between these vectors.

In our case, we decided to go with the cosine distance to measure the distance between the vectors.It is equal to 1 minus cosine similarity. So it will enable us to find similarities between vectors.

Logically,if both images are same person, then measurement should be small. Otherwise, the measurement should be large if two images are different person. So we need to define a threshold value.

Based on our observations, we decided that the cosine similarity should be less than 0.25 in order to assume that the images represent the same person.

3 Results

3.1 Dataset

In order to test the performance of both algorithms.We used a dataset that contains 3 photos of each of the 35 persons. The dataset was separated in two parts:

- Train set:two labeled images of each person
- Test set: one image non labeled of each person

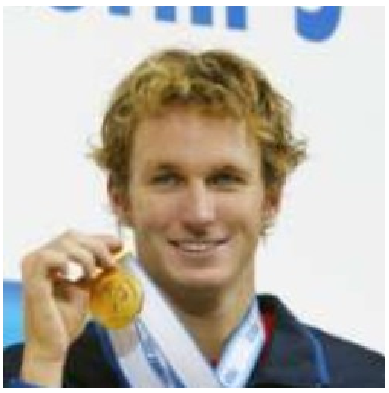
Figure 3.1: Train image 1

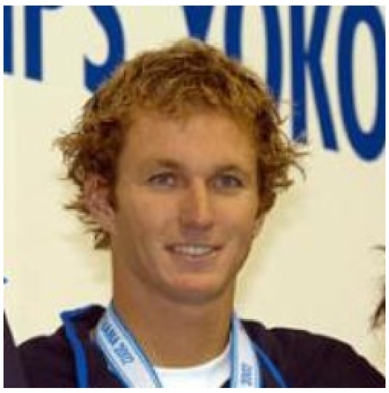
Figure 3.2: Train image 2

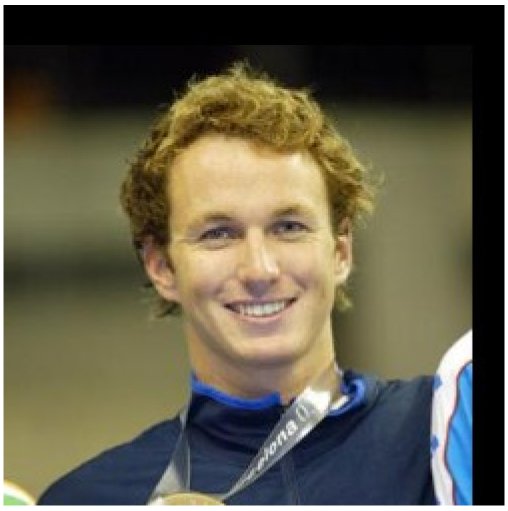
Figure 3.3: Test image

3.2 Evaluation

In order to measure the performance of our algorithms,the evaluation will be done as follows: for each image of the test set,we will measure its distance to each image of the train set which will give us to cases:

- Good match :if the distance is lower than the threshold and both images have the same label or if the distance is bigger than the threshold and the two images have different labels
- Bad match:if the distance is lower than the threshold and the two images have different labels or if the distance is bigger than the threshold and both images have the same label

3.2.1 Simple Image

For the SIFT algorithm,the measure used is the minimum pair distance and the threshold is equal to 30.

For the Siamese network,the measure used is the cosine similarity and the threshold is equal to 0.25

Algorithm	Good matches	Bad matches
SIFT	97.14	2.85
Siamese network	96	4

3.3 Algorithm Robustness

3.3.1 Contrast

In order to test the robustness of both algorithms,we created a new test set from the previous one by changing the contrast of the images. By following the same evaluation procedure,we obtain the following results:

Algorithm	Good matches	Bad matches
SIFT	97.14	2.85
Siamese network	96.65	3.34

3.3.2 Rotation

In order to test the robustness of both algorithms,we created a new test set from the previous one by applying a 30 degrees rotation to the images. By following the same evaluation procedure,we obtain the following results:

Algorithm	Good matches	Bad matches
SIFT	97.14	2.85
Siamese network	97.34	2.65

3.3.3 Brightness

In order to test the robustness of both algorithms,we created a new test set from the previous one by changing the brightness of the images. By following the same evaluation procedure,we obtain the following results:

Algorithm	Good matches	Bad matches
SIFT	97.14	2.85
Siamese network	94.44	5.55

4 Conclusion

We can see from the results obtained that SIFT algorith performs better in this case and we test the resistance of the algorithms to the change of contrast,rotation and brightness,we see that SIFT algorith is invariant whereas the results change slightly for the case of Siamese networks.

it's also interesting to note that techniques based on combining these two methods exist such as extracting the SIFT features as low level features and extract high level features from the convolutional neural network.This enables to make the best use of these two different type of features and fuse the extracted information to achieve a better accuracy [5].

References

[1] `https://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html`.

[2] David G. Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 2004.

[3] E. Grosso M. Tistarelli M. Bicego, A. Lagorio. On the use of sift features for face authentication. Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06), 2006.

[4] Jesus Carlos Pedraza Ortega Juan Manuel Ramos Arreguin Efrén Gorrostieta Hurtado Raúl David Palma Olvera, Elizabeth Martínez Zerón. A feature extraction using sift with a preprocessing by adding clahe algorithm to enhance image histograms. International Conference on Mechatronics, Electronics and Automotive Engineering, 2014.

[5] Hsueh-Ming Hang Shiuan Huang. Multi-query image retrieval using cnn and sift features. Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2017.

[6] Rein van den Boomgaard. `https://staff.fnwi.uva.nl/r.vandenboomgaard/IPCV20172018/LectureNotes/IP/ScaleSpace/index.html`.