Lebanese University Faculty of Sciences Hadat Campus

# OCR

OCR-like application that reads the information of the FS1 student ID

Supervised by:
Dr. Ihab Sbeity

Done by:

# TABLE OF CONTENTS

# ACKNOWLEDGEMENTS

It’s always a pleasure to commend the fine people of the Lebanese University for the sincere guidance we received to uphold our skills in this domain. Whatever we have achieved in this project would not have been without the kind support and help of many individuals. As so we would like to extend our sincere thanks to them. We are highly indebted to all the doctors for their guidance and constant supervision as well as for providing necessary information for completing this project.

We express our profound gratitude and indebtedness to Dr. Ihab Sbeity, Department of Computer Science, for introducing the present topic and for his inspiring intellectual guidance, constructive criticism and valuable suggestions throughout the project work.
We are also thankful to Dr. Mohammad Dbouk, Dr. Bassem Haidar and other staffs in the Department of Computer Science for motivating us as well as for their comments and advices.

# ABSTRACT

In this project we will take you behind the scenes on how we built an Optical Character Recognition (OCR) – based software to detect the ID number on our Lebanese University ID card. We used concepts of computer vision and deep learning as well as image recognition and classification. Our project is implemented using Python language and openCV library. This project is the first step in designing a student identification software for a new automated student attendance system which will replace the current traditional one at the Lebanese University.

**Keywords** : Optical Character Recognition; Computer Vision; Deep Learning; Image Processing; Python; OpenCV; Identification

# CHAPTER 0: INTRODUCTION

## 1. Project Background

The process of monitoring attendances in any institution is an important part in determining the efficiency of the services offered by the institution and the general interest of those who benefit from the institution's services, in our case, the students.

Nowadays, most universities and colleges are still using the traditional attendance system which requires students to sign on a piece of paper every time they attend a class throughout the whole semester.

## 2. Problem Statement

Using the traditional attendance system and mechanisms, attendance is taken by various means but all just as flawed and defective and thus we are faced by some serious problems that need to be looked into.

### 2.1 Buddy Signing:

One of the means of taking attendance is by signing a passed attendance sheet. This is obviously flawed as course mates can "help" those who did not attend the class by signing their attendance which is also known as buddy-signing.

### 2.2 Frauds:

As the student id of the university is understandably not a smart chip card, a fraud can forge a student id with whatever picture or id of choice. This presents a serious security breach as anyone can do that and enter the campus under false id. Moreover, some "helpful" senior class students may enter exams instead of their junior friends and will most probably get away with it because during exams the invigilator, who is usually not the lecturer, does not have a picture of the id holder.

### 2.3 Time consumption:

To avoid buddy signing the lecturer/assistant/invigilator has to personally mark attendance by painstakingly either calling out for every enrolled student or visiting each present student and making sure he/she is signing the right slot.

# 3. Project Objectives

Our primary goal is to help the lecturers/assistants/invigilators improve and organize the process of tracking and managing student attendance. Additionally, we seek to:

- Provide a valuable attendance service for both teachers and students.
- Reduce manual process errors by providing automated and a reliable attendance system.
- Increase privacy and security where students cannot present themselves or their friends while they are not present.

# 4. Motivation

Having a system that can automatically capture student's attendance by flashing their student card at the camera can really save all the formerly mentioned troubles.

Looking at a bigger picture, deploying the system throughout the academic faculty will benefit the academic management as students' attendance to classes is one of the key factor in improving the quality of teaching and monitoring their students' performance.

# 5. Project Scope

In our project we have aimed to develop the first step in the student identification software that will be able to identify the student who flashes his/her id in front of the camera. In this step the software will detect the flashed university card, segment its id, recognize the numbers of the id and identify the holder by searching the data found in the university database.

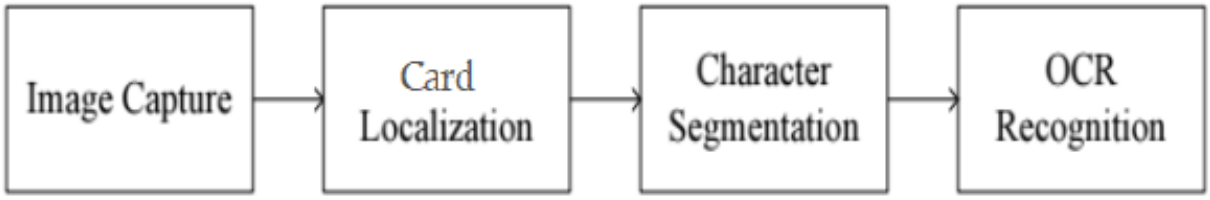



*Figure 1 Architecture of ID Recognition System*

# CHAPTER I: STATE OF ART

In this chapter we are going to introduce briefly main concepts that we used throughout the project and have been central in the whole implementation process.

## 1. Machine Learning

Artificial intelligence is one field that is using machine learning. Machine learning is a science discipline that focuses on creating algorithms that can learn from data. This is interesting since we will need to use algorithms that make use of this to make image recognition usable. This could also be used to boost performance of algorithms.

### 1.1 Training

Using training data in an algorithm can improve both the algorithms' execution time and accuracy.

This can for example be a filter to remove uninteresting parts of a dataset. Such a filter can be created by sorting good and bad data which then is used as input for the algorithm.

## 2. Computer Vision

The field of computer vision includes both analyzing, processing and understanding images. One of the themes has been to try to replicate the human vision ability with computers. This has been tried to be achieved through machine learning and finding shapes within images. It can be said that image processing is a sub field to computer vision.

### 2.1 Recognition

Within computer vision there are several fields. The most interesting one out of these for this project is recognition. Recognition is when you make a computer study an image in such a way that it can determine what is in the picture. If this could be used within this report, we could determine whether for one the user actually is holding a Lebanese University ID which would make our task easier and for two recognize the numbers of the ID. When implementing the latter, it is necessary to implement a machine learning algorithm.

# 3. OpenCV - Open source Computer Vision

In the area of image processing there is one established library that makes performing appropriate operations easier. OpenCV (Open Source Computer Vision) was officially started by Intel in 1998 and have since evolved to the defacto image recognition library. Today it is maintained by the company Itseez with contributions from many of the big players in the field, including Intel, Google and NVidia.

OpenCV is a library of components available under a BSD-license and contains more than 2500 optimized algorithms. It covers both basic and state-of-the-art computer vision and machine learning algorithms, including histogram comparison, template and feature matching, which is particularity interesting in the field of image comparison.

These are some reasons of why we preferred OpenCV over Matlab.

## 3.1 Specific

OpenCV was made for image processing. Each function and data structure was designed with the Image Processing coder in mind. Matlab, on the other hand, is quite generic. You get almost anything in the world in the form of toolboxes. All the way from financial tool boxes to highly specialized DNA tool boxes.

## 3.2 Speedy

Matlab is just way too slow. Matlab itself is built upon Java. And Java is built upon C. So when you run a Matlab program, your computer is busy trying to interpret all that Matlab code. Then it turns it into Java, and then finally executes the code.

If you use C/C++ you don’t waste all that time. You directly provide machine language code to the computer, and it gets executed. So ultimately you get more image processing, and not more interpreting.

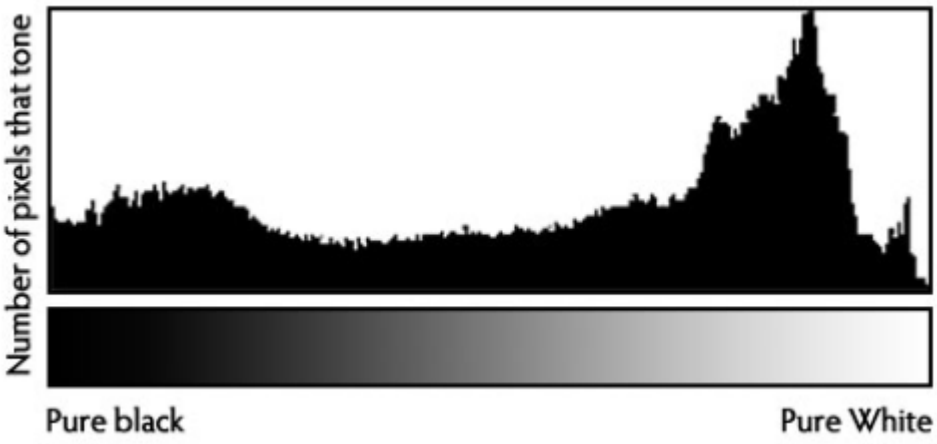
Sure you pay the price for speed – a more cryptic language to deal with, but it’s definitely worth it. You can do a lot more. You could do some really complex mathematics on images with C and still get away with good enough speeds for your application.

### 3.3 Efficient

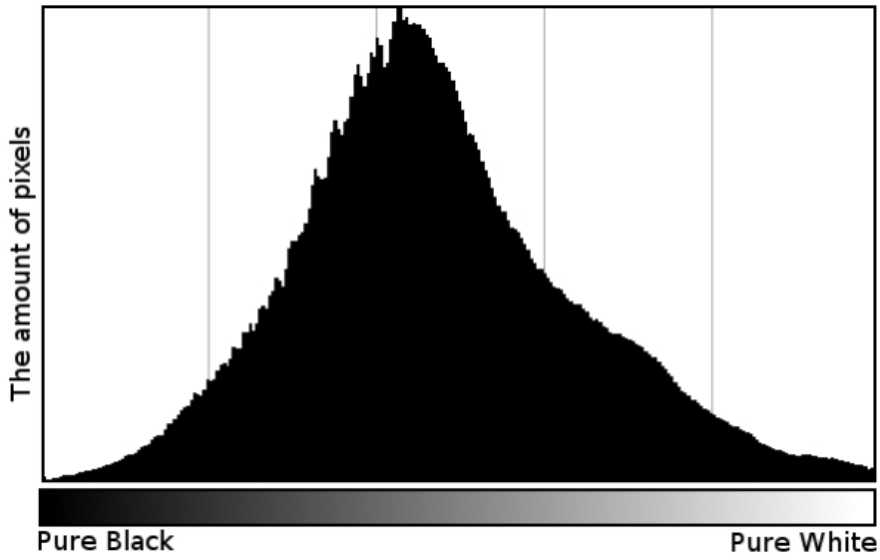
Matlab uses just way too much system resources. With OpenCV, you can get away with as little as 10mb RAM for a real-time application. But with today's computers, the RAM factor isn't a big thing to be worried about. You do need to take care about memory leaks, but it isn't that difficult.

## 4. Image histogram

A histogram is a graph that shows the distribution of light, where the x-axis is pixel brightness and y-axis is the amount of pixels, see figure 1.




Everyone wants their pictures to be as pretty as possible. If an image is too dark or too bright the picture losses clearness and sharpness. The task of determining if an image is to bright or to dark was only possible by humans until the histogram was invented. Thanks to histograms it is possible for a computer to determine the light distribution. The ideal histogram is when the light distribution looks like a Normal Distributed graph, see figure 2.




### 4.1 Normalization

If there is a picture that is too bright or too dark, a technique that is called normalization can be used to make the picture's pixels be pushed within another distribution span. This can make it easier to compare histograms if you have two images that are similar but have different brightness levels.

## 5. Keypoints and descriptors

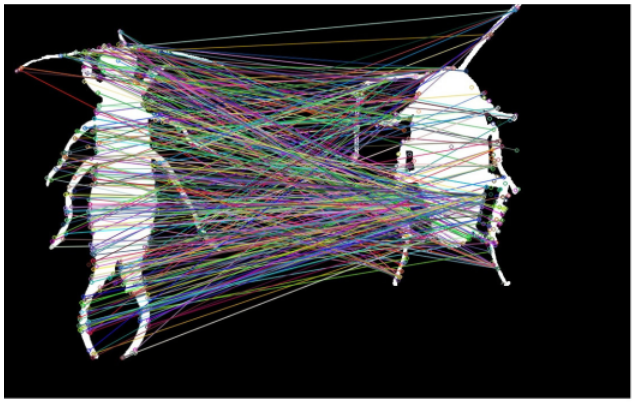
When trying to describe important parts of an image, the keypoints and descriptors are often used. Keypoints are used to describe position and size of an interesting point. An interesting point is not a shape in an image, but rather a characteristic point in the image. This might be for example be an edge on the ID card, that could be significant when matching. The descriptors are more advanced. A descriptor can be many things and is sometimes called features and feature descriptors. Often it tries to estimate the direction the pattern has in the keypoint, which is defined as numerical value. This can be used to match with other descriptors.

### 5.1 Histograms of Oriented Gradients (HOG)

HOG is a keypoint detection algorithm. OpenCV includes a framework which allows for training of the HOG algorithm.

# 6. Scale-invariant feature transform (SIFT)

SIFT is an algorithm that finds and describes high relevant features in a picture. It can also determine relevance degree. These features are described with so called descriptors. The descriptors can be used to match different keypoints to each other, see figure 3. To find the keypoints within the images, SIFT makes use of HOG. This keypoint matching could potentially be used to match an already saved picture of the Lebanese University Student ID and any recognized look-alike in the captured image.



## 6.1 Key point Localization

Once potential key points locations are found, they have to be refined to get more accurate results.

*Figure 5 Approximate key point location*

## 6.2 Orientation Assignment

Now an orientation is assigned to each key point to achieve invariance to image rotation. It creates key points with same location and scale, but different directions. It contributes to stability of matching.

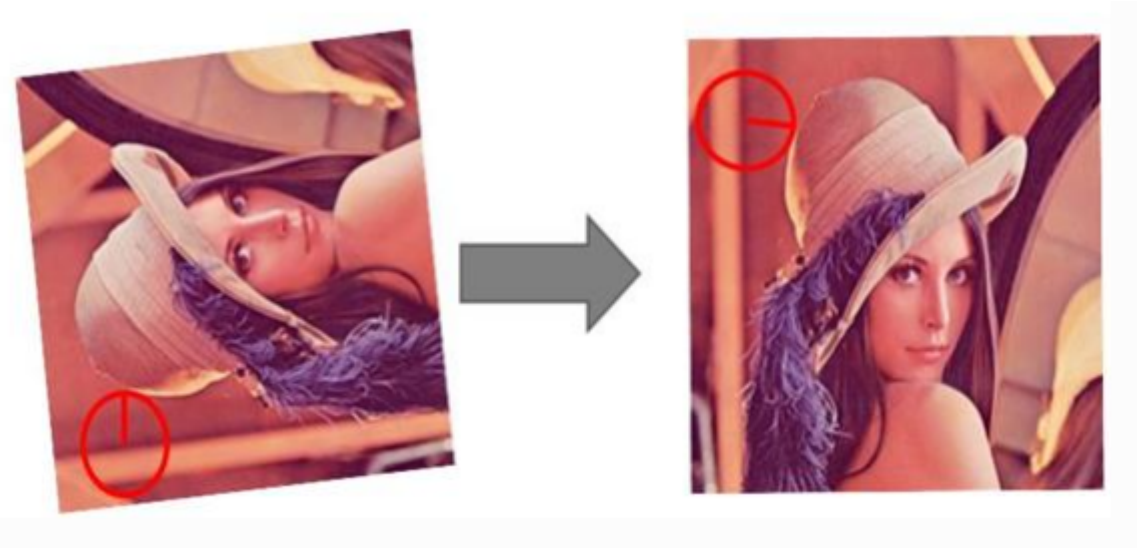


*Figure 6 Orientation Assignment*

## 6.3 Key Point Descriptor

Now key point descriptor is created. A 16x16 neighborhood around the key point is taken. It is divided into 16 sub-blocks of 4x4 size. For each sub-block, 8 bin orientation histograms are created. So total of 128 bin values are available. It is represented as a vector to form key point descriptor. In addition to this, several measures are taken to achieve robustness against illumination changes, rotation etc.

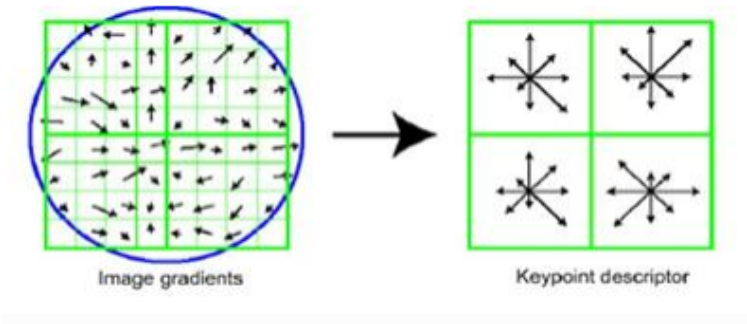



*Figure 7 Key point Descriptor*

## 6.4 Key point Matching

Key points between two images are matched by identifying their nearest neighbors. But in some cases, the second closest-match may be very near to the first.

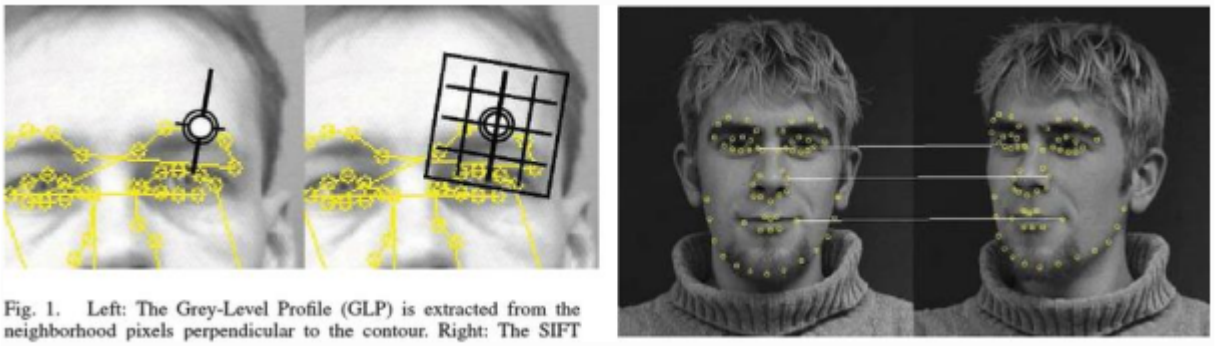


Fig. 1. Left: The Grey-Level Profile (GLP) is extracted from the neighborhood pixels perpendicular to the contour. Right: The SIFT

*Figure 8 Key point Matching*

# 7. FLANN - Fast Library for Approximate Nearest Neighbors

When matching descriptors to each other, OpenCV offers a library called FLANN. This is faster than using the brute force matching algorithm, because it approximates the best match. It contains a collection of algorithms optimized for fast nearest neighbor search in large datasets and for high dimensional features.

# CHAPITRE II: CONTRIBUTION

In this chapter we are going to explain our approach to achieve our projects goals by explaining them step by step.

## 1. Card Detection

The card detection step greatly influences the accuracy of the next steps. The input is an image containing none or one Lebanese University ID and the output should be the portion of the image containing the ID only. A plethora of algorithms have been proposed the last years to solve this kind of challenging problem. The brute-force approach of processing every pixel in an image gives a very high processing time. Instead, the common approach is to use prominent features in the ID card and only process the pixels having these features reducing the processing time considerably. Hence we have adopted the SIFT algorithm.

### 1.1 Capture of Images Through a Webcam

If the webcam is properly installed, OpenCV would easily detect it and OpenCV capture functions can be used to get the images from the camera. The VideoCapture() function of cv2 initializes capturing video from the camera and stores it in the CvCapture structure, which is the video capturing structure. The VideoCapture () function may be passed an index number to identify which camera is to be the source of images. In case there is only 1 camera, the parameter (0) is used. Thereafter, the read() function may grab the frame (image) from the CvCapture structure. From there, the image can be displayed on the screen by the imshow () function in a window. For video streaming the process of capture and display may be placed inside a loop.

Below is the code fragment for capturing and displaying images through the webcam:

```
cam=cv2.VideoCapture(0)
ret, QueryImgBGR=cam.read()
cv2.imshow('result',QueryImgBGR)
cam.release()
```

```
cv2.destroyAllWindows()
```

## 1.2 SIFT feature extractor

The SIFT feature extractor will extract some distinct features from images as key points for our object recognition. To be able to use SIFT cv2 library needs to be imported. After that, the feature extractor SIFT needs to be initialized using the function SIFT_create() of xfeatures2d of the cv2 library. Now, a training image which is the image of the object to be recognized, in our case the Lebanese University ID, has to be loaded. Next the feature extractor is used to detect features and store them in two variables one is trainKP which is the list of key points / coordinates of the features, and the other in trainDesc which is list of descriptions of the corresponding key points. The former and latter will be needed to find visually similar objects in captured input image.

Below is the code fragment for initializing the SIFT feature extractor and using it on the training image:

```
import cv2
import numpy as np
detector = cv2.xfeatures2d.SIFT_create()
trainingImage = cv2.imread('TrainingData/TrainImg.jpeg', 0)
```

## 1.3 FLANN feature matcher

A feature matcher will match the features from the sample/ training image with the input image from the webcam. For FLANN based matcher, two dictionaries need to be passed which specify the algorithm to be used, its related parameters etc. First one is IndexParams. For various algorithms, the information to be passed is explained in FLANN docs. For algorithms like SIFT, SURF etc. the following can be passed “dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)”. Second dictionary is the SearchParams. It specifies the number of times the trees in the index should be recursively traversed. Higher values give better precision, but also take more time.

Corresponding code:

```
kp1, des1 = sift.detectAndCompute(trainingImage, None)
kp2, des2 = sift.detectAndCompute(inputImage, None)

FLANN_INDEX_KDTREE = 0
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
search_params = dict(checks=50)

flann = cv2.FlannBasedMatcher(index_params, search_params)

matches = flann.knnMatch(des1, des2, k=2)
```

Here FLANN is using KNN to match the features with k=2, so we will get 2 neighbors.

## 1.4 Matching

Now that the FLANN feature matcher has been used to match the features in both images, and the matches results have been stored in matches variable, the former have to be filtered to avoid false matches. This is done by checking distance from the nearest neighbor m and the next neatest neighbor n and considering the match a good one if the distance from point m is less the 70% of the distance on point n and appending that point to goodMatch array variable.

```
goodMatch=[]
    for m,n in matches:
        if(m.distance<0.75*n.distance):
            goodMatch.append(m)
```

Moreover, to make sure that enough feature matches are there to call these a match, a threshold MIN_MATCH_COUNT is set.

For the following, numpy library needs to be imported. The coordinates of the good matches are appended to separate arrays for training image and input image correspondingly. These coordinated are then converted to numpy lists to translate image points from training to input by finding the transformation constant H using the function findHomography() of cv2.

```
if(len(goodMatch)>=MIN_MATCH_COUNT):
```

```
        tp=[]
        qp=[]
        for m in goodMatch:
            tp.append(trainKP[m.trainIdx].pt)
            qp.append(queryKP[m.queryIdx].pt)

        tp,qp=np.float32((tp,qp))

        H,status=cv2.findHomography(tp,qp,cv2.RANSAC,3.0)
```

### 1.5 Adjustment

Using the transformation constant H from earlier the coordinates from training image are translated to input image by the function perspectiveTransform() which performs the perspective matrix transformation of vectors. Now by using the function wrapPerspective() the detected card in the input is adjusted and flattened to look just like the training image.

```
h,w=trainImg.shape
trainBorder=np.float32([[[0,0],[0,h-1],[w-1,h-1],[w-1,0]]])
queryBorder=cv2.perspectiveTransform(trainBorder,H)
M1 = cv2.getPerspectiveTransform(queryBorder, pts)
queryBorder = cv2.warpPerspective(inputImage, M1, (w, h))
```

## 2. Segmentation

Features extracted from entire image are known as the global features and those features extracted from blocks identified during segmentation or from subdivision of the number plate are known as local features.

In this part further image processing is done on extracted ID card to remove unnecessary data. After character segmentation, the card has only those characters that belong to ID number.

The final output image of the previous step is resized using the resize() function of cv2 to static dimensions. Then the region of the id number is extracted using the width height ratios matching. After that, each number is saved separately by segmenting the id region.

```
imS = cv2.resize(queryBorder, (960, 540))  # Resize image
height, width = img.shape[:2]
#(top Left of cropping rectangle)
start_row, start_col = int(height * .47), int(width * .72)
#ending pixeL coordinates (bottom right)
end_row, end_col = int(height * .55), int(width * .83)
# SimpLy use indexing to crop out the rectangle we desire
cropped = img[start_row:end_row, start_col:end_col]
cv2.imwrite("Images/numbers.jpeg", cropped)
```

# 3. Number Recognition

This is going to be the last stage, it's at this stage we introduce the concept of machine learning. Machine learning can simply be defined as the branch that deals with data and processes it to discover patterns that can be used for future predictions. The major categories of machine learning are supervised learning, unsupervised learning and reinforcement learning. Supervised learning makes use of a known dataset (called the training dataset) to make predictions. We'll be taking the path of supervised learning because we already have an idea of how 0s, 1s and all the numbers look like. Supervised learning can be divided into two categories; classification and regression. Character recognition belongs to the classification category.

All we need now is to get a training data set, train a model, and then use the model for prediction.

## 3.1 Training Data Set

Image sets of the 10 digits from different cards have to be created and classified into a folder each. Then the images in each folder will be visited, preprocessed, and appended with its corresponding digit into two separate lists that will be saved as text files for later use.

## 3.2 Predicting Numbers

There are several classifiers we can use with each of them having its advantages and disadvantages. KNN is one of the simplest of classification algorithms available for supervised learning. The idea is to search for closest match of the test data in feature space.

First we need to initiate the KNN algorithm using the function KNearest_create() and pass the training data and responses to train the KNN (It constructs a search tree) by the function train(). Then the extracted sample digits will each be read, preprocessed, and compared with the model to find the nearest match using the function findNearest().

```
samples = np.loadtxt('generalsamples.data', np.float32)
responses = np.loadtxt('generalresponses.data', np.float32)
responses = responses.reshape((responses.size, 1))

model = cv2.ml.KNearest_create()
model.train(samples, cv2.ml.ROW_SAMPLE, responses)

    for each in numbers:
        img_dir = "Images/number" + each + ".jpeg"
        im = cv2.imread(img_dir)
        gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
        blurred = cv2.GaussianBlur(gray, (1, 1), 0)
        thresh = cv2.adaptiveThreshold(blurred, 255, 1, 1, 11, 2)
        roismall = cv2.resize(thresh, (10, 10))
        roismall = roismall.reshape((1, 100))
        roismall = np.float32(roismall)

        retval, results, neigh_resp, dists =
model.findNearest(roismall, k=1)
        # print(results)
        string = str(int(float(results[0])))
```

```
        fu11_number.append(string)
```

The predicted numbers will then be joined and displayed to the user.

## 4. Connecting to the Database

This is the last step where software connects to the database and checks for the existence of the detected id number to display the corresponding information.

First, the library pymyql has to be imported to connect using its function connect() that takes 4 parameters: server name, host name, password, and database name. the sql statements are then executed using an initiated cursor.

```
db = pymysql.connect("localhost", "root", "root", "test")
    # prepare a cursor object using cursor() method
    cursor = db.cursor()
    sql = "SELECT * FROM users WHERE ID='%d'" % (number)
    try:
        # Execute the SQL command
        cursor.execute(sql)
        # Fetch all the rows in a list of lists.
        results = cursor.fetchall()
        if cursor.rowcount == 1:
            for row in results:
                ID = row[0]
                Name = row[1]
                print(ID, Name)
```

# CHAPTER III: RESULTS AND DISCUSSION

# 1. Card Detection

## 1.1 Capturing

The camera is started by pressing the “start” button, and an image is captured by the “capture”


*Figure 9 Capturing*

## 1.2 SIFT – ing

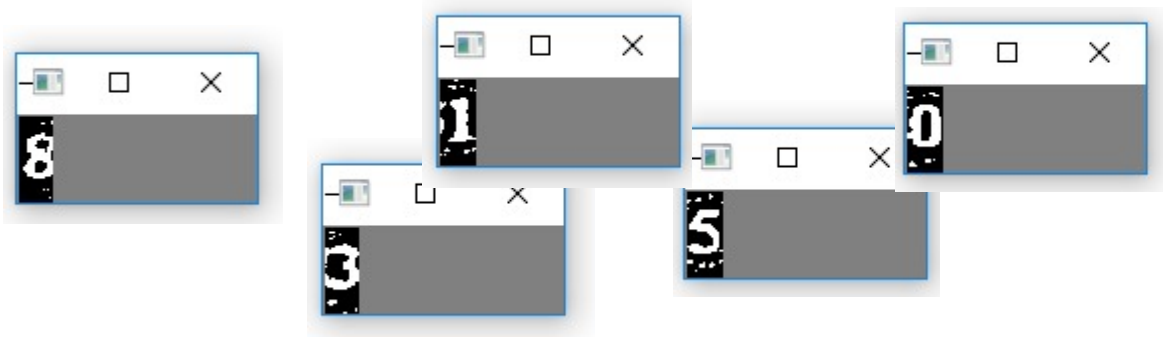
The key point descriptors on the training sample image of the Lebanese University ID card:


*Figure 10 KP-descriptors of ID*

## 1.3 Matching


*Figure 11 Detected Card*

# 2. Segmentation



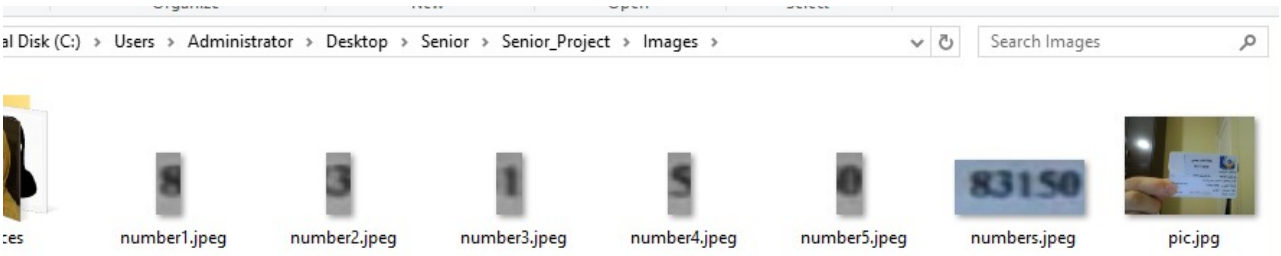



*Figure 13 Images Saved*

# 3. Number Recognition and Connection to Database

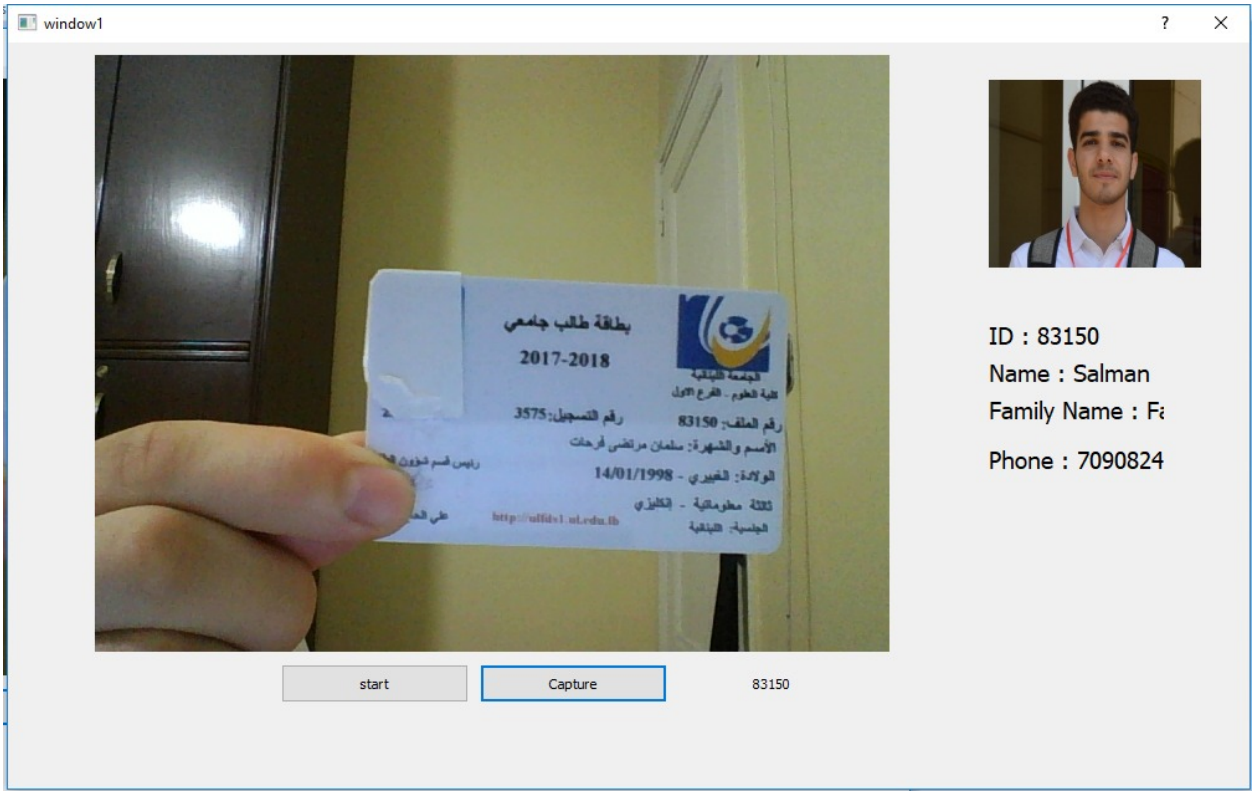



*Figure 14 Final Result*

# CHAPITRE IV: CONCLUSION AND FUTURE WORK

## 1. Conclusion

In this report, a training based approach has been discussed for the Lebanese University card's id number recognition. The implemented software is the first step in designing an automated student attendance monitoring system that can automatically capture students' attendances by only flashing their student card at the camera. This system will save a lot of troubles mentioned in the beginning of the report. The results of the proposed algorithm signify the success of our approach.

## 2. Future Work

As previously mentioned, this project is only the first step of a proposed system. As so, for the future we hope to implement the whole system where there will be facial detection and recognition to ensure the validity of the flashed id. Moreover, we can develop the system to be able to detect multiple students and their IDs and manage their attendance.

# LIST OF ABBREVIATIONS

OCR: Optical Character Recognition

ID: Identification

HOG: Histograms of Oriented Gradients

SIFT: Scale-Invariant Feature Transform

KP: Key Points

FLANN: Fast Library for Approximate Nearest Neighbor

KNN: K-Nearest Neighbor

# LIST OF FIGURES