MIDDLE EAST TECHNICAL UNIVERSITY

ELECTRICAL-ELECTRONICS ENGINEERING DEPARTMENT

EE300 SUMMER PRACTICE REPORT

Student Name:Berk Alperen Bener
Student ID:2165991
SP Company Name: Amani Technologies Limited
SP Date: 12.06.2019 - 02.08.2019
Submission Date: 25.09.2019

TABLE OF CONTENT

1.Introduction

I have performed my second year EE300 Summer Practice in Amani Technologies Limited, which is reg tech startup based in UAE. My practice lasted totally 8 weeks, starting from 12.06.2019 and ending in 02.08.2019. The division that I work was dealing with KYC system. Means know your client. Division was trying to make a module that can be used to verify customer identity using image of their identity documents where there I learn many image processing algorithms, ways and using machine learning to classify documents.

The report starts with brief history of company followed my main project which has subgroups that in each I narrated my design step by step. After conclusion there is references that I used to make design decision and appendices that includes all the codes that I wrote throughout the internship which I used Python.

,

2.Description of the Company

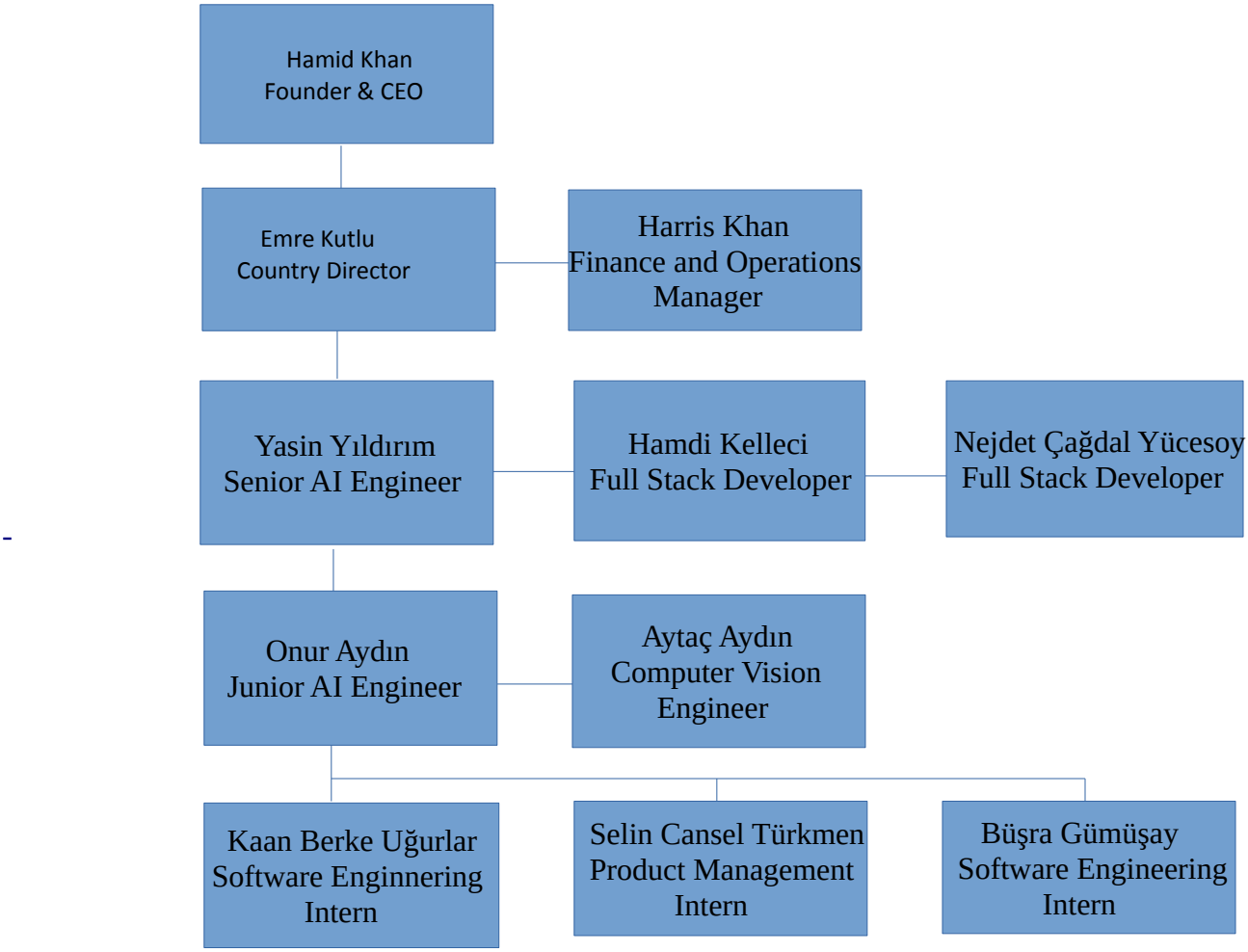
2.1 Company Name

Amani Technology Limited

2.2 Company Location

Esentepe, Kolektif House, Harman Street Entrance, Talatpaşa Cd. No: 5, 34330 Şişli/Istanbul

2.3 Organizational Structure of the Company



2.4 Main Area of Business

Amani offers a unique and fully automated KYC, rectification and on-boarding tool built on highly scalable proprietary technology. It will fundamentally transform the UAE on-boarding processes, eliminate non compliance and create a KYC audit and reporting trail, all whilst materially reducing costs and improving customer experience. Amani has developed an AI Powered Identity Verification platform for the Financial Services, Telecoms, Hospitality, Travel, HR, Transportation and Security sectors.

2.5 Brief History of Company

Amani Technologies Ltd is a Reg-tech start up based in the DIFC, Dubai. Amani was a finalist in the 2018 DIFC Fin tech Hive Accelerator program and now has a tangible and validated pipeline of customers in the Middle East.

3. Project Description

The project that I involved was designing and implementing end to end system that deals with image input that provided by costumer and from that image classifying documentation type, extracting necessary information, comparing it with previously confirmed customer database and return all these information and with confidence level back to consumer. Under the supervision of (BS. Bilkent Electrical and Electronics Engineer, MS. Computer Science) Onur Aydın I delivered minimum viable product and further enriched and scale the system in 45 days of first compulsory second voluntary internship.

The documentation that I dealt with was identity documentation. Identity documents is government or institution issued any documentation that proves person identity. This information and validity of is very important in the functioning of government and corporate life. This documentations is varied. You can use driver license,passports,id cards,residence cards etc. as your identity document and using these things you can cross the borders, open bank accounts, benefit government aids etc. Since these documents are very powerful we come across many attempts to forge the fake of it. But these attempts is nearly always imperfect. First of all generally in the documentation there can be inconsistency in the parts of information also even passport information and texture is correct, the database and document information can be inconsistent as a result. These forgery cases can be detected and identity security can be established. First of all our customer for this image processing job was First Abu Dhabi Bank. Since we had restricted time frame I had to customize the module for them so I need to make this system for majority of the United Arab Emirates population.

3.1. Minimum Viable Product

First I had to design the overall system

I divided system to 3 sub parts. Because to accomplish every part individually we need to use different technologies and also I thought if we design system as modular we can use it in other projects. Since it is minimum viable product, I didn't think about scalability. We need a system which accomplish the process for limited number of document type.

3. 1. 1. Classification Of Document Types

The first question was how do we classify the identity documents according to its national origin and what type of document it is from its image. We need this classification because different kinds of documents of various nations include different kinds of information. To know what we look for or where should we looking for and extract necessary information for particular document type accordingly we need to classify it. We can get this classification information from document.

Figure 1: The information inside of the rectangle sufficient to understand what type of document it is and which country it is belong to.

Even document structures may differ for each nation, document types or document generation(Figure 2), we know that for particular type of document from particular nation for particular time frame it has generic structure(such as British passports after 2015 have generic structure that allow us to know which part of document posses which information.) Considering this information we strictly limited our image input of our platform which provided by consumer as to get full picture of document no bigger or smaller than its frame. This allow us to know that image of the document have same proportion as original document.

Figure 2: These two different type of documents from two different nations have different information about people who possess it.

Since we know that for ID documentation document type is written top of the document(Figure 1) we should look at that area. I decided to use template matching method to find it. Template matching is a technique in digital image processing for finding small parts of an image which match a template image. It can be used in manufacturing as a part of quality control,a way to navigate a mobile robot,or as a way to detect edges in images. The main challenges in the template matching task are: occlusion, detection of non-rigid transformations, illumination and background changes, background clutter and scale changes.[1]

First I should have found images of documents, but the images that I found in internet, even if they are complete, they have some background images and document stands in different perspective. To fix that I wrote (Appendix 1.1) code, using this code user can manually select the edge points of the document that will be extracted and using “cv2.getPerspectiveTransform” and “cv2.warpPerspective” can fix it.

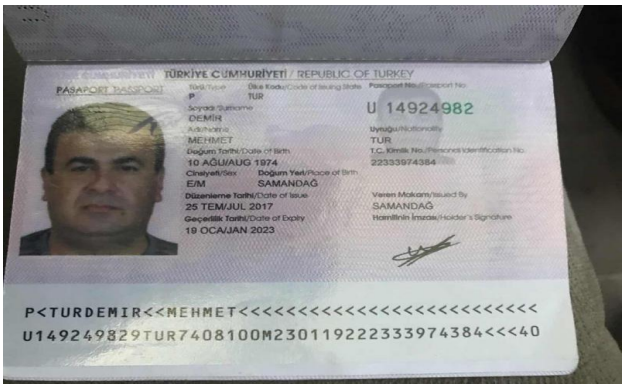


Figure 2: Figure 3: Original image which is found in internet.

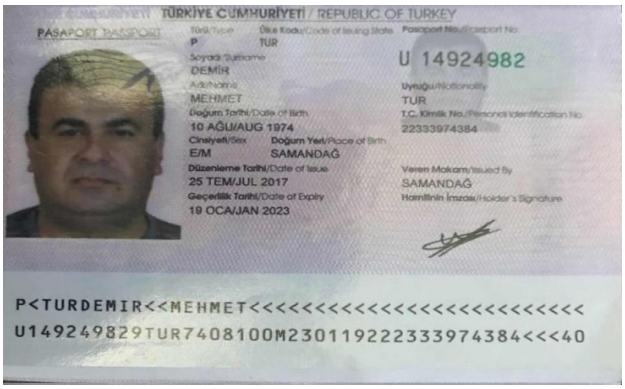


Figure 4: After cropped and perspective fixer.

After I get ID document photo with fixed perspective and border that our customer would provide to apply template matching I need to get samples of the template. Since there exist hundreds of nations and id document types and generation to ease the process I wrote (Appendix 1.2). Using these code I can crop templates by taking rectangle frame around that one such template can be seen below.(Figure 5)

Figure 5: Extracted template sample

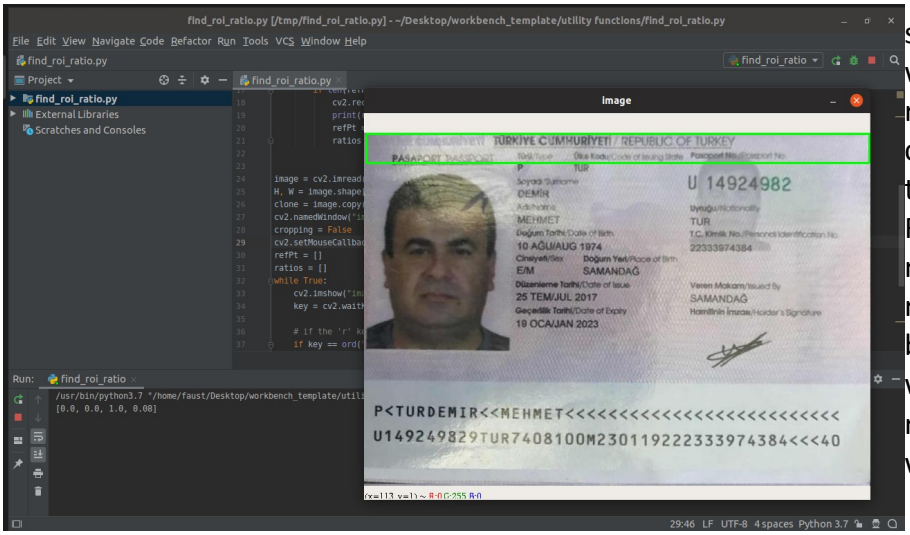


Figure 6: Using green rectangles we specify ROI and as a result we get ratios as seen in terminal.

Now since we have templates, we search on image to find matches, but as we know looking entire image for regarding template would be time consuming and error prone. So, to fix that issue I decided to restrict search ROI(Region of interest) but since even we restrict our customer to get image for restricted frame there could be shifts because of that we cant restrict the ROI with pixel values so I decided to find ROI ratios to picture height and width and wrote (Appendix 1.3).

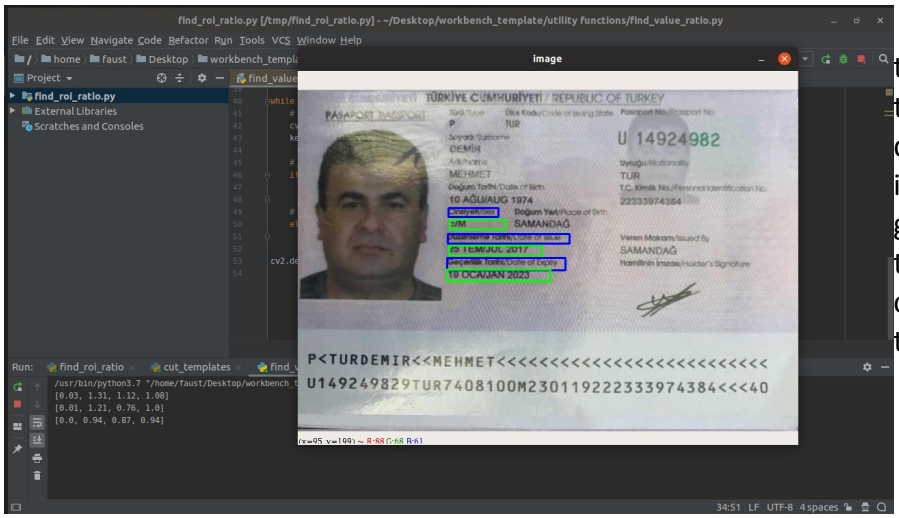
So now we know have what type of template we are going to search and where we are going to search with respect to image height and width. Using these information when we match the template with image we classify what kind of document it is.

3.1.2.Extracting Necessary Informations

After classifying the document, I started to work on to get the rest of the information that document posses. To do that first I cut out the descriptor templates of the information then I find the ROI ratios of where I should look for them.

Figure 7: The descriptors inside of the rectangles help us to find where the information is located.

Then after finding the place where these descriptive words are positioned. I wrote a code (Appendix 1.4) to get the information that is described, using the proportion of the value of the descriptive words position to value position.(Figure 8)



Write now I automated the system to get particular type of document take the necessary templates finding document type and get information image. Though the important thing is not get these values as image format but then change it to string values to compare it with database values. To do that I use OCR technology.

Figure 8: By choosing blue rectangle we predict the template matching zone and regarding it we choose value zone by green rectangle.

3.1.3 Returning Information

Also I had to compare it with database JSON CSV format to do that I wrote the following codes (Appendix 1.6) with this code I compare OCR result with existing database values since OCR can give deviated results I used
Levenshtein distance [6] algorithm to decide matches gives feedback wit confidence level.

3.2 Improvement of The System

At this point even I provided minimum necessities for the system to recognize document type, take the necessary information and to compare it with existing database by iterating the workflow for each nation, document type and generation to cover target nations. For every document type we had to go through template cutting, ROI ratio finding process again and again this would slow down to scaling module into other document type and even there could be future use for the module to classify document that does not posses kind of information that what kind of document it is.

To fix this problem I decided to use multi class classification using supervised learning. Supervised learning basically maps the input to output based on example inputs-outputs. We use labeled training data that consists input object and output value. This input object can be images and output value can be what type of document it is. To classify image we use feature vector. This features can be extracted by using sophisticated algorithms and by adding labels and by feeding this labeled feature vector to selected learning model then computer extrapolate this knowledge to label unseen images. First I had to decide what type of learning model I should use for document classification I decided to use SVM(Support Vector Machine).

From the beginning of my internship I was studying machine learning from MOOC(massive open online course) site Coursera. I wanted to put into practice what I have learned.

3.2.1 Data Preparation

For every data analytics project data preparation is the first step from definition we see that data preparation can include many discrete tasks such as loading data or data ingestion, data fusion ,data cleaning,data augmentation and data delivery.[1] To train my learning model I had to gather data but the problem was there was no prepared data set for identity documentations due to privacy issues so I had to scrape internet by myself for every document types. I had some data specification. First of all image should not be smaller than height of 480 px and width of 768 px as I uniform and augment my data set I decided these size specification so anything smaller than this size would result loss of resolution which result loss of features.

After using various search engines (Google,Yandex,Bing etc.) I gather 15 unique document image for each document type. The number had to be equal to prevent imbalanced data set. A data set is imbalanced if it contains many more samples from one class than from the rest of the classes. Data sets are unbalanced when at least one class is represented by only a small number of training examples (called the minority class) while other classes make up the majority. In this scenario, classifiers can have good accuracy on the majority class but very poor accuracy on the minority class(es) due to the influence that the larger majority class[2] I used different search engines because every search engines breath to index image data was different and using only one was not sufficient.

But as I will explain at 3.2.2 my feature size is quite large(16000 features) because of that I need to expand my data set for my learning model to learn feature label relationship accurately. To do that I decided to use data augmentation. Data augmentation is a strategy that enables practitioners to significantly increase the

diversity of data available for training models, without actually collecting new data. Data augmentation techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks.[3] So using this techniques I expand my number of data for each class to 90 images. The code that I wrote for this task can be seen in Appendices 6.7 Image Augmenter.

Lets analyze the code. For handling images I used PIL and OpenCV library and to augment them imgaug package. To automate process I differentiate the files where my original images are and my augmented images so to access these files since I use Linux machine I used glob and os libraries. First I get user defined number “k” to determine how many times each image will be augmented. Then I defined “rotate_image” function which rotates image such that the whole image is preserved and fitted to frame. After that I reach where every document folder is held “Documents” than each sub folder for each documents and every photo in these sub folders. After I get these images I open these images as pillow image but since for pillow image channel configuration is not “RGB” I convert it to “RGB” image. After that I randomly configure my augmentation specifications GaussianBlur,brightness,contrast and resize my image to my uniform size. After that I randomly rotate my image using my user defined function “rotate_image” and save them to their regarding augmented document folder.

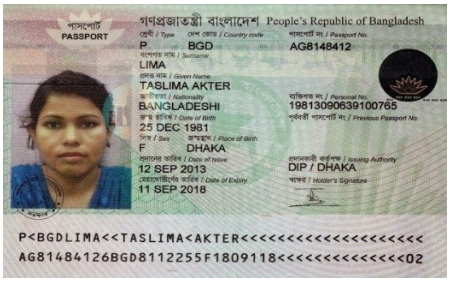


Figure 9: Original image without modification

Figure 10: Modified image brightness of it change,blur applied and rotated.

3.2.2 Feature Extraction and Labeling

After I augment my data I need to extract image features that I will use to determine what kind of document I am dealing with because SVM use feature vector to train its model. There exist many alternative algortihms such as SURF, ORB, SIFT, BRIEF. But ORB is the fastest and nearest algorithm that performs as good as licensed alternative SIFT [4] As a result I decided to use ‘ORB’ algorithm to extract my image features. It is based on the FAST keypoint detector and the visual descriptor BRIEF (Binary Robust Independent Elementary Features). Its aim is to provide a fast and efficient alternative to SIFT.

In my code using “import SVM_Aug_Tester” first I call data augmentation script it asks how many augmentation will be applied per image. After I enter the number program goes to directory that holds folders for every nationality and then it goes every sub directory and finally pick every image inside of it one by one. We open image as pillow object because of that RGB schema changes so we first convert our image channel to RGB and convert it into array.

Then we initialize our orb object and using “orb.detect” we find key points and then we use “orb.detect” to find descriptors for our key points as a result we get matrice of(32,500) 32 key points and 500 descriptors per key point. But while augment our image if we distort it (such as too much blurring) there can be loss of descriptors to prevent that we standard our descriptor number to 500 then we flatten this (32,500) matrice into (1,16000) matrice and add it into our array “c” which initialized as (1,16000) zero matrice we iterate this process every image at the sub folder while filling our length array the number of image that passes

500 descriptor case which will be useful when label our data according to class type and we continue to do this process until all sub folders finish. Then we delete first row of “c” which is “0” due to initialization and below that in the for loop we label our data using length array data as a result we got (300,160001) matrice which includes 6 sub folder and 50 augmented image per folder,16000 descriptor for 32 key point per image and final column that indicates which class the image belongs to. Then we shuffle our rows because in the next step we are going to partition data. After that we save it as pickle data.

3.2.3 Model Training and Testing

To make this data readable for my SVM I convert this to pickle format. Then we take resulting matrice data and label part separately and save them as“d” and “l” arrays. Then we partition our data to 70% train 30% test data using “train_test_split”. Then we start to arrange our learning method we use linear, sigmoid, rbf and poly kernel beside C parameters ranging from 1,15.Then we define our SVM model with gamma=0.00000000001 we reduce gamma to this range due to increase accuracy,preventing overshoot and we don’t have time restriction.

Then by merging our range of parameters with our learning model and cv= 4…. we create our classifier and into our classifier we put our train data and train label for it to learn the correlation. We save resulting classifier as pickle. And we retrieve it to predict our test data’s label. And using “classification_report(l_test, prediction_array)” we get resulting precision,recall and f1-score even our macro accuracy 0.93 percent for some class 0 and 3 we see numbers as high as 1 this happens due to over training.

Since we use augmentation instead of distinct examples due to intrinsic hardship of the finding document examples our model learn too god our few modified few cases. So using SVM for documentation classification with limited data is not feasible.

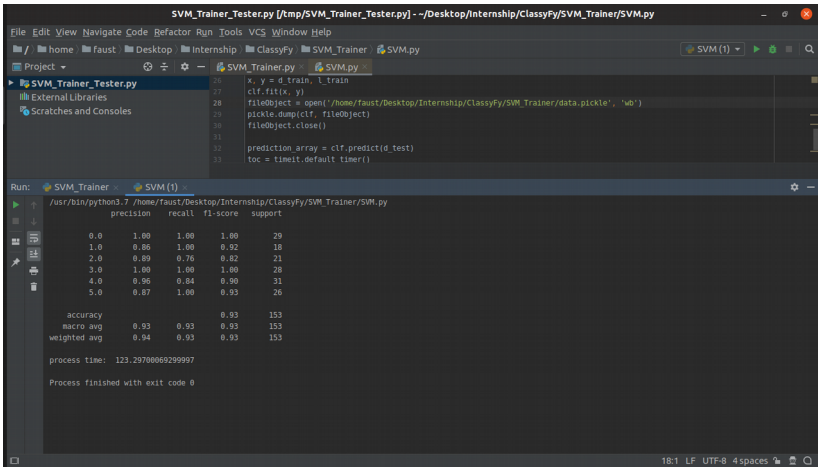


Figure 11: Resulting precision,recall and f1-score for each class shows that there exist overfitting.

4.Conclusion

In my summer internship I delivered commercial product module that can be used by customer. The module differentiate document type by template matching technique. Moreover, using same technique I extracted important value's image from documents and using OCR I convert them to string value. I wrote program to compare these values with existing databases. Furthermore, I used supervised learning to make classification independent of template matching but it failed due to lack of original data.

Throughout my internship both by myself and from my supervisor I learned a lot about image processing how to design product from scratch implement it under rigid time frame, how to develop modular program and harmonic teamwork. Beside technical skills I learned a lot about project management and agile technique.

5.Reference

[1] https://www.wikiwand.com/en/Template_matching
[2] https://www.wikiwand.com/en/Data_preparation
[3] https://towardsdatascience.com/comparing-different-classification-machine-learning-models-for-an-imbalanced-dataset-fdae1af3677f
[4] https://bair.berkeley.edu/blog/2019/06/07/data_aug/Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images
[5]Document Classification with Support Vector Machines,Konstantin Mertsalov,Michael McCreary
[6]https://dzone.com/articles/the-levenshtein-algorithm-1

6. Appedice

6.1 Perspective Fixer

```
import cv2
import math
import imutils
import numpy as np
import os

def perspective(event, x, y, flags, param):
    global coors, resized

    if event == cv2.EVENT_LBUTTONDOWN:
        coors.append((x, y))
        cv2.circle(resized, (x, y), 5, (0, 255, 0), -1)

        if len(coors) == 4:
            p1 = tuple(map(lambda x: x * ratio, coors[0]))
            p2 = tuple(map(lambda x: x * ratio, coors[1]))
            p3 = tuple(map(lambda x: x * ratio, coors[2]))
            p4 = tuple(map(lambda x: x * ratio, coors[3]))

            w = int(math.sqrt((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2))
            h = int(math.sqrt((p1[0] - p3[0])**2 + (p1[1] - p3[1])**2))

            pts1 = np.float32([p1, p2, p3, p4])
            pts2 = np.float32([[0, 0], [w, 0], [0, h], [w, h]])
            M = cv2.getPerspectiveTransform(pts1, pts2)
            dst = cv2.warpPerspective(image, M, (w, h))
            cv2.imwrite(os.path.join(SAVE_PATH, "warped.jpg"), dst)

            coors = []
            resized = clone.copy()

INPUT_PATH = "/home/faust/Desktop/transform.jpg"
image = cv2.imread(INPUT_PATH)  # CHANGE THE PATH
H, W = image.shape[:2]
resized = imutils.resize(image, width=600)
ratio = W / 600
clone = resized.copy()
cv2.namedWindow("image")
cv2.setMouseCallback("image", perspective)
coors = []
SAVE_PATH = "/home/faust/"

while True:
    # display the image and wait for a keypress
    cv2.imshow("image", resized)
    key = cv2.waitKey(1) & 0xFF
   # if the 'r' key is pressed, reset everthing
    if key == ord("r"):
        coors = []
        resized = clone.copy()
    # if the 'q' key is pressed, break from the loop
    elif chr(key) in ("q", "Q"):
        break

cv2.destroyAllWindows()
```

6.2 Template Cutter

```
import cv2
import os

def cut(event, x, y, flags, param):
    global coors

    if event == cv2.EVENT_LBUTTONDOWN:
        coors.append((x, y))
        if len(coors) == 2:
            cv2.rectangle(image, coors[0], (x, y), (255, 0, 0), 1)
            image_name = input("Enter the name of the template: ")
            while not image_name.endswith(".jpg"):
                image_name = input("Type .jpg at the end of the name!", '\n')

            t_region = clone[coors[0][1]:y, coors[0][0]:x]
            t_region = cv2.resize(
                t_region, (0, 0), fx=RESIZE_RATIO, fy=RESIZE_RATIO)
            image_save_path = os.path.join(SAVE_PATH, image_name)
            cv2.imwrite(image_save_path, t_region)
            coors = []


image = cv2.imread('samples/ukpassport_1.jpg')  # CHANGE THE PATH
clone = image.copy()
H, W = image.shape[:2]
SAVE_PATH = "cuted_templates"
RESIZE_RATIO = 600/W

cv2.namedWindow("image")
cv2.setMouseCallback("image", cut)

coors = []

if not os.path.exists(SAVE_PATH):
    os.makedirs(SAVE_PATH)

while True:
    # display the image and wait for a keypress
    cv2.imshow("image", image)
    key = cv2.waitKey(1) & 0xFF

    # if the 'r' key is pressed, reset everthing
    if key == ord("r"):
        coors = []
        image = clone.copy()
    # if the 'q' key is pressed, break from the loop
    elif key == ord("q"):
        break

cv2.destroyAllWindows()
```

6.3 ROI(Region of Interest) Ratio Finder

```
import cv2

def click_and_crop(event, x, y, flags, param):
    global refPt, ratios

    # if the left mouse button was clicked, record the starting
    # (x, y) coordinates and indicate that cropping is being
    # performed
    if event == cv2.EVENT_LBUTTONDOWN:
        refPt.append((x, y))
        tlx_r = round(x / W, 2)
        tly_r = round(y / H, 2)
        ratios.append(tlx_r)
        ratios.append(tly_r)

        if len(refPt) == 2:  # draw a rectangle around the region of interest
            cv2.rectangle(image, refPt[0], refPt[1], (0, 255, 0), 2)
            print(ratios)
            refPt = []
            ratios = []


image = cv2.imread('samples/ukpassport_1.jpg')
H, W = image.shape[:2]
clone = image.copy()
cv2.namedWindow("image")
cropping = False
cv2.setMouseCallback("image", click_and_crop)
refPt = []
ratios = []
while True:
    cv2.imshow("image", image)
    key = cv2.waitKey(1) & 0xFF

    # if the 'r' key is pressed, reset the cropping region
    if key == ord("r"):
        image = clone.copy()
        refPt = []
        ratios = []
    # if the 'q' key is pressed, break from the loop
    elif key == ord("q"):
        break

cv2.destroyAllWindows()
```

6.4 Value Ratio Finder

```
import cv2

def find_ratio(event, x, y, flags, param):
    global coors, dim

    if event == cv2.EVENT_LBUTTONDOWN:
        coors.append((x, y))
        if len(coors) == 2:
            cv2.rectangle(image, coors[0], (x, y), (255, 0, 0), 2)

        if len(coors) == 4:
            temp_tx = coors[0][0]
            temp_ty = coors[0][1]
            temp_w = abs(coors[1][0] - coors[0][0])
            temp_h = abs(coors[1][1] - coors[0][1])

            value_tx = coors[2][0]
            value_ty = coors[2][1]
            value_w = abs(coors[3][0] - coors[2][0])
            value_h = abs(coors[3][1] - coors[2][1])

            tx_r = round((value_tx - temp_tx) / temp_w, 2)
            ty_r = round((value_ty - temp_ty) / temp_h, 2)
            bx_r = round(value_w / temp_w, 2)
            by_r = round(value_h / temp_h, 2)

            print([tx_r, ty_r, bx_r, by_r])
            cv2.rectangle(image, coors[2], (x, y), (0, 255, 0), 2)

            coors = []


image = cv2.imread('samples/ukpassport_1.jpg')  # CHANGE THE PATH
clone = image.copy()
cv2.namedWindow("image")
cv2.setMouseCallback("image", find_ratio)
coors = []

while True:
    # display the image and wait for a keypress
    cv2.imshow("image", image)
    key = cv2.waitKey(1) & 0xFF

    # if the 'r' key is pressed, reset everthing
    if key == ord("r"):
        coors = []
        image = clone.copy()
    # if the 'q' key is pressed, break from the loop
    elif key == ord("q"):
        break

cv2.destroyAllWindows()
```

6.5 CSV-JSON Matcher

```
import collections
import json
import csv
import stringdist
import json
from itertools import product
import re
import months

class Matcher:

    def map_string(dictionary, input_str):
        min_dist = round(len(input_str) * 0.6)
        min_dist_value = input_str
        for key, value in dictionary.items():
            dist = stringdist.levenshtein(key, input_str)
        if dist < min_dist:
            if dist == 0:
                return value
            min_dist = dist
            min_dist_value = value
        return min_dist_value

def apply_regex(string, str_type="mixed"):
    if str_type == "digit":
        return ''.join((re.findall("[0-9]+", string)))
    elif str_type == "letter":
        return ''.join(re.findall("[A-Za-z]+", string))
    elif str_type == "mixed":
        return ''.join(re.findall("[0-9A-Za-z]+", string))
    else:
        return string

def unify_date(date_str):
    letters = apply_regex(date_str.upper(), "letter")
    date_str = apply_regex(date_str, "digit")
    if len(letters) < 2:
    # Possible formats are:
    # 21.05.1993, 21-05-1993, 210593 etc..
        unified = date_str[-2:] + date_str[2:4] + date_str[:2]
    else:
    # 21MAY/MAY1993, 21MAY/MAY93 etc...
        month = map_string(months, letters)
    unified = date_str[-2:] + month + date_str[:2]
    return unified

def process(file_csv,file_json):
    mapper = {"id_number": "id_number", "name": "name","birth_date": "birth_date"}
    dummy = 0
    k = 0
    pre_confidence = 0

    with open(file_json +'.json') as fake_data_json:
        json_check = json.load(fake_data_json)
        if all(k in json_check for k in ("first_name", "last_name")):
```

```
            new_val = json_check['first_name'] + ' ' + json_check['last_name']
            json_check['name'] = new_val
        del json_check['first_name']
        del json_check['last_name']

with open(file_json+'.json', 'w') as fake_data_json:
    json.dump(json_check, fake_data_json)

fake_data_json = open(file_json+'.json')
with open(file_csv+'.csv') as fake_data_csv:
    headers = next(fake_data_csv)
    number_of_header = headers.count(',')
    Data = collections.namedtuple('data', headers)
    fake_data_csv = csv.reader(fake_data_csv, delimiter=',')
    for line_csv, line_json in product(fake_data_csv, fake_data_json):
        csv_obj = Data(*line_csv)
        line_json = json.loads(line_json)
    for i, m in mapper.items():
        json_string = line_json[m]
        if m == 'id_number':
            line_json[m] = re.sub("[^0-9]", "", json_string)
        if i == 'birth_date':
            standard_csv_date = unify_date(getattr(csv_obj, i))
            standard_json_date = unify_date(line_json[m])
            dist = stringdist.levenshtein(standard_csv_date,standard_json_date)
            value_csv = str(standard_csv_date)
            value_json = str(standard_json_date)
        else:
            dist = stringdist.levenshtein(getattr(csv_obj, i),  line_json[m])
            value_csv = str(getattr(csv_obj, i))
            value_json = str(line_json[m])
            upper_lev = max(len(value_csv), len(value_json))
            confidence = (upper_lev - dist)/upper_lev
        if confidence >= 0.75:
            dummy += 1
        pre_confidence += confidence
    overall_confidence = pre_confidence / number_of_header
    if overall_confidence >= 0.75:
        return {'Customer ID': str(line_csv[0]), 'Confidence': overall_confidence}
    else:
        pre_confidence = 0
        overall_confidence = 0
        dummy = 0

return {'Customer ID': None, 'Confidence':overall_confidence}
```

6.6 Image Augmenter

```
import cv2
import random
from imgaug import augmenters as iaa
import glob
import numpy
import os

k = int(input(print("Enter Sample Number:")))


def get_immediate_subdirectories(a_dir):
    return [name for name in os.listdir(a_dir)
            if os.path.isdir(os.path.join(a_dir, name))]


def rotate_image(mat, angle):
    height, width = mat.shape[:2]
    image_center = (width/2, height/2)
    rotation_mat = cv2.getRotationMatrix2D(image_center, angle, 1.)
    abs_cos = abs(rotation_mat[0,0])
    abs_sin = abs(rotation_mat[0,1])
    bound_w = int(height * abs_sin + width * abs_cos)
    bound_h = int(height * abs_cos + width * abs_sin)
    rotation_mat[0, 2] += bound_w/2 - image_center[0]
    rotation_mat[1, 2] += bound_h/2 - image_center[1]
    rotated_mat = cv2.warpAffine(mat, rotation_mat, (bound_w, bound_h))
    return rotated_mat


for folder_name in get_immediate_subdirectories('/home/faust/Desktop/Internship/ClassyFy/SVM_Trainer/
Data_Augmentation_Trainer/'):
    a=0
    z = 1
    os.mkdir('/home/faust/Desktop/Internship/ClassyFy/SVM_Trainer/Data_Extraction_Trainer/'+folder_name)
    output_path = '/home/faust/Desktop/Internship/ClassyFy/SVM_Trainer/Data_Extraction_Trainer/'+folder_name
    for images in glob.glob('/home/faust/Desktop/Internship/ClassyFy/SVM_Trainer/Data_Augmentation_Trainer/'+folder_name+'/
*.jpg'):
        if z <=15:
            z+=1
            a +=1
            open_cv = cv2.imread(images)
            open_cv_image = numpy.array(open_cv)
            for i in range(1, k+1):
                rnd_int = random.randint(-15, 15)
                seq = iaa.Sequential([
                    iaa.Resize({"height":480, "width":768}),
                    iaa.GaussianBlur(sigma=(0, .0)), # blur images with a sigma of 0 to 3.0
                    iaa.Add((-10, 10), per_channel=0.5), # change brightness of images (by -10 to 10 of original value)
                    iaa.ContrastNormalization((0.5, 1.0), per_channel=0.5)], random_order=True)
                image_aug = seq.augment_image(open_cv_image)
                result = rotate_image(image_aug,rnd_int)
                cv2.imwrite(os.path.join(output_path,'%s_%s.jpg'%(i,a)), result)
        else:
            break
```

6.7 Feature Extractor

```
import numpy as np
import cv2
import pickle
import os
import glob
from PIL import Image
import numpy
import SVM_Aug_Tester

def get_immediate_subdirectories(a_dir):
    return [name for name in os.listdir(a_dir)
            if os.path.isdir(os.path.join(a_dir, name))]

sub_dir = get_immediate_subdirectories('/home/faust/Desktop/Internship/ClassyFy/SVM_Tester/Data_Extraction_Tester/')

c =np.zeros((1,16000))
a=0
length = []
f=0
size=0
for folder_name in sub_dir:
    print(size)
    length.append(size)
    f += 1
    size = 0
    print(f,size)
    for images in glob.glob('/home/faust/Desktop/Internship/ClassyFy/SVM_Tester/Data_Extraction_Tester/'+folder_name+'/*'):
        print(folder_name)
        a+=1
        imge = Image.open(images)
        pil_image = imge.convert('RGB')
        open_cv_image = numpy.array(pil_image)
        open_cv_image = open_cv_image[:, :, ::-1].copy()
        orb = cv2.ORB_create()
        kp = orb.detect(open_cv_image,None)
        kp, des = orb.compute(open_cv_image, kp)
        print(des.shape)
        if des.shape[0] == 500:
            c = np.r_[c, np.ndarray.flatten(des).reshape(1, 16000)]
            size += 1
            print(size,'kj')

length.append(size)
length.pop(0)
c = numpy.delete(c, (0), axis=0)
lab = np.zeros((length[0], 1))
arr_size = len(sub_dir)
for y in range(1,arr_size):
    n = np.full((length[y], 1), y)
    lab = np.r_[lab,n]

data = np.c_[c,lab]
np.random.shuffle(data)
print(data.shape)
fileObject = open('/home/faust/Desktop/Internship/ClassyFy/SVM_Tester/test_data.pickle', 'wb')
pickle.dump(data, fileObject)
fileObject.close()
```

6.8 Model Trainer and Tester

```
import cv2
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import svm
import pickle
from sklearn.metrics import classification_report
import timeit
from sklearn.model_selection import GridSearchCV

f = open('data.pickle', 'rb')
array = pickle.load(f)

d = array[:, :-1]
l = array[:, -1]


tick = timeit.default_timer()
d_train, d_test, l_train, l_test = train_test_split(d, l, test_size=0.30)
parameters = {'kernel':('linear', 'rbf','poly','sigmoid'), 'C':[1, 15]}
svc = svm.SVC(gamma=0.00000000001)
clf = GridSearchCV(svc, parameters, cv=4)


x, y = d_train, l_train
clf.fit(x, y)
fileObject = open('/home/faust/Desktop/ClassyFy/SVM_Trainer/svm_data.pickle', 'wb')
pickle.dump(clf, fileObject)
fileObject.close()

prediction_array = clf.predict(d_test)
toc = timeit.default_timer()
print(classification_report(l_test, prediction_array))
print('process time: ', toc-tick)
```