PAN Card Tampering Detection

The purpose of this project is to detect tampering of PAN card using computer vision. This project will help different organization in detecting whether the Id i.e. the PAN card provided to them by thier employees or customers or anyone is original or not.

For this project we will calculate structural similarity of original PAN card and the PAN card uploaded by user.

```
# import the necessary packages
from skimage.metrics import structural_similarity
import imutils
import cv2
from PIL import Image
import requests


!mkdir pan_card_tampering
!mkdir pan_card_tampering/image


# Open image and display
original = Image.open(requests.get('https://www.thestatesman.com/wp-content/uploads/2019/07/p
tampered = Image.open(requests.get('https://assets1.cleartax-cdn.com/s/img/20170526124335/Pan
```

Loading original and user provided images.

```
# The file format of the source file.
print("Original image format : ",original.format)
print("Tampered image format : ",tampered.format)

# Image size, in pixels. The size is given as a 2-tuple (width, height).
print("Original image size : ",original.size)
print("Tampered image size : ",tampered.size)
```

```
Original image format :  JPEG
Tampered image format :  PNG
```

```
Original image size :  (1200, 800)
Tampered image size :  (282, 179)
```

- Converting the format of tampered image similar to original image.

```
# Resize Image
original = original.resize((250, 160))
print(original.size)
original.save('pan_card_tampering/image/original.png')#Save image
tampered = tampered.resize((250,160))
print(tampered.size)
tampered.save('pan_card_tampering/image/tampered.png')#Saves image
```

```
(250, 160)
(250, 160)
```

- Here, we checked the format and size of the original and tampered image.

```
# Change image type if required from png to jpg
tampered = Image.open('pan_card_tampering/image/tampered.png')
tampered.save('pan_card_tampering/image/tampered.png')#can do png to jpg
```

- Converting the size of tampered and original image.

```
# Display original image
original
```

- Orginial PAN card image used for comparision.

```
# Display user given image
tampered
```

- User provided image which will be compared with PAN card.

```
# load the two input images
original = cv2.imread('pan_card_tampering/image/original.png')
tampered = cv2.imread('pan_card_tampering/image/tampered.png')
```

- Reading images using opencv.

```
# Convert the images to grayscale
original_gray = cv2.cvtColor(original, cv2.COLOR_BGR2GRAY)
tampered_gray = cv2.cvtColor(tampered, cv2.COLOR_BGR2GRAY)
```

- Converting images into grayscale using opencv. Because in image processing many applications doesn't help us in identifying the important, edges of the coloured images also coloured images are bit complex to understand by machine beacuse they have 3 channel while grayscale has only 1 channel.

```
# Compute the Structural Similarity Index (SSIM) between the two images, ensuring that the di
(score, diff) = structural_similarity(original_gray, tampered_gray, full=True)
diff = (diff * 255).astype("uint8")
print("SSIM: {}".format(score))
```

```
SSIM: 0.31678790332739426
```

- Structural similarity index helps us to determine exactly where in terms of x,y coordinates location, the image differences are. Here, we are trying to find similarities between the original and tampered image. The lower the SSIM score lower is the similarity.

```
# Calculating threshold and contours
thresh = cv2.threshold(diff, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
```

Here we are using the threshold function of computer vision which applies an adaptive threshold to the image which is stored in the form array. This function transforms the grayscale image into a binary image using a mathematical formula.

Find contours works on binary image and retrive the contours. This contours are a useful tool for shape analysis and recoginition. Grab contours grabs the appropriate value of the contours.

```
# loop over the contours
for c in cnts:
    # applying contours on image
    (x, y, w, h) = cv2.boundingRect(c)
    cv2.rectangle(original, (x, y), (x + w, y + h), (0, 0, 255), 2)
    cv2.rectangle(tampered, (x, y), (x + w, y + h), (0, 0, 255), 2)
```

Bounding rectangle helps in finding the ratio of width to height of bounding rectangle of the object. We compute the bounding box of the contour and then draw the bounding box on both input images to represent where the two images are different or not.

```
#Diplay original image with contour
print('Original Format Image')
Image.fromarray(original)
```

```
Original Format Image
```


```
#Diplay tampered image with contour
print('Tampered Image')
Image.fromarray(tampered)
```

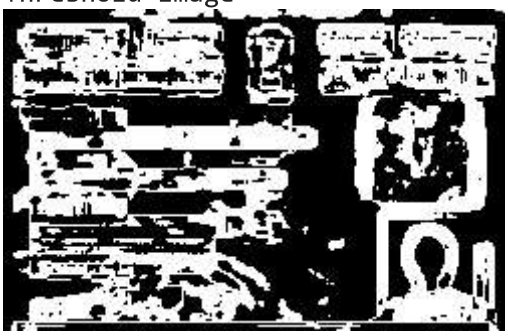
Tampered Image


```
#Diplay difference image with black
print('Different Image')
Image.fromarray(diff)
```

Different Image

```
#Display threshold image with white
print('Threshold Image')
Image.fromarray(thresh)
```

Threshold Image



Summary

Finding out structural similarity of the images helped us in finding the difference or similarity in the shape of the images. Similarly, finding out the threshold and contours based on those threshold for the images converted into grayscale binary also helped us in shape analysis and recognition.

As, our SSIM is ~31.2% we can say that the image user provided is fake or tampered.

Finally we visualized the differences and similarities between the images using by displaying the images with contours, difference and threshold.

Scope

This project can be used in different organizations where customers or users need to provide any kind of id in order to get themselves verified. The organization can use this project to find out whether the ID is original or fake. Similarly this can be used for any type of ID like adhar, voter id, etc.