DOCUMENT SCANNER

PROJECT REPORT

BY:

PARAS RAJ PAHARI

Table of Contents

1 INTRODUCTION

The Document Scanner tool can be applied to detect the paper on the clicked or captured photos and cut out the paper to give the only view of the paper with added black and white effect.

1.1 Project Description

The project is on document scanner. It involves uploading the required image from the disk or capturing it through the IP camera. Then, the document scanner will grab the document from the uploaded/captured image. Document Scanner will automatically detect the edges of the document in the image using the canny edge detection. Further, it finds the four corner points of the document by detecting the largest contour. Then it will apply perspective transformation to obtain a top-down view of the document. Then it apply threshold to the image to give the black and white effect to the document.

1.2 Assumptions and Constraints

Following assumptions will introduce the constraints on the source image.

- The captured or uploaded image should have paper on it, where the paper is the biggest object in the image.
- Paper should be rectangle.
- There should be proper contrast between image background and paper on the image.

- It’s better if the background is darker.
- Image should be of decent quality.

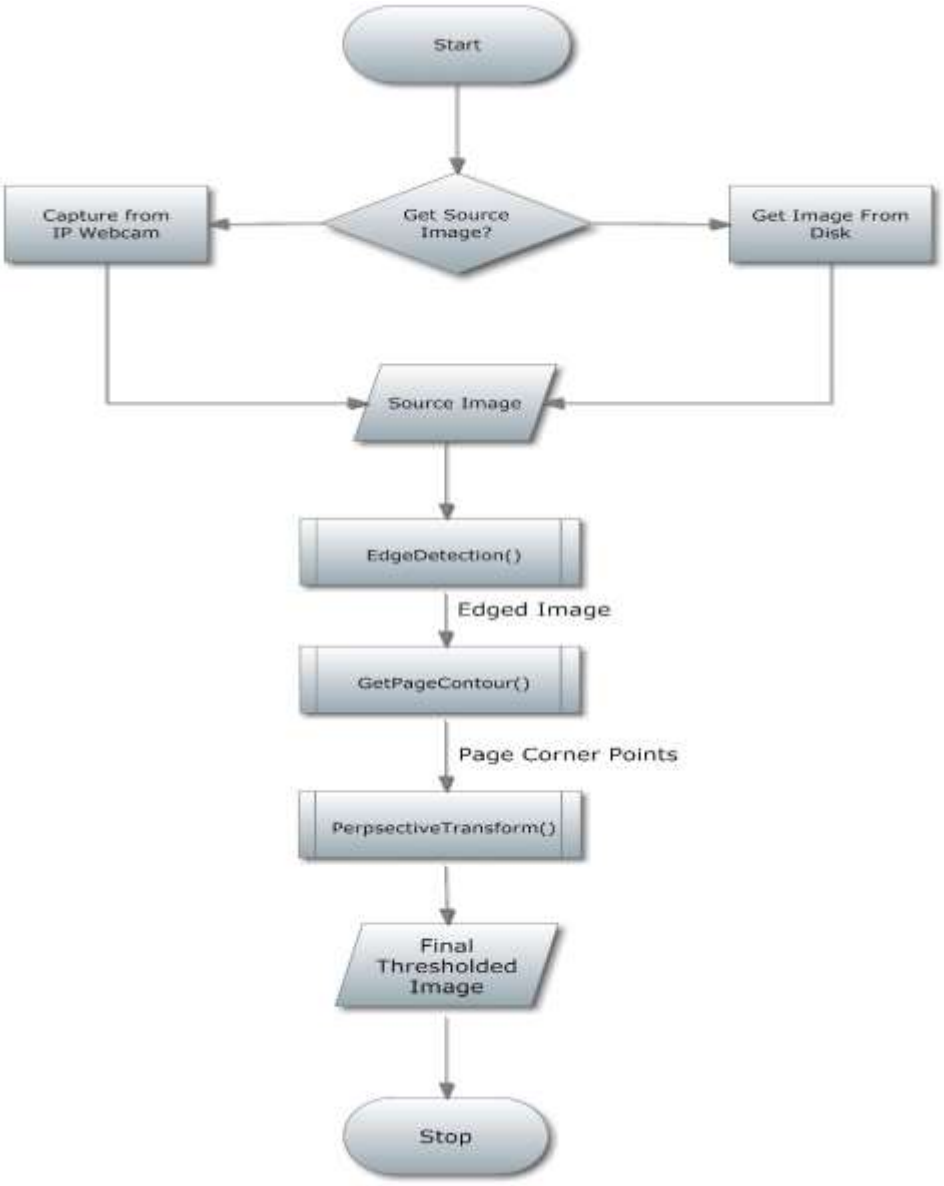
1.3 Required Python Libraries

- Numpy : Required for Image Array Operations
- Opencv : Opencv library as cv, to process the image
- Urllib2 : Required for opening the stream URL from IP address of webcam
- os.path : To check if the file path exists or not
- PIL : Pillow Library , it is used to read the image and save it as pdf

1.4 Steps

- Get Source Image from Disk or Capturing from IP Webcam
- Resize the Source Image and Introduce it to Bilateral Filter, and Apply Canny Edge Detection
- Then, Contour Detection is applied to the edged image to get the page contour which is the four points of the paper.
- Then, Perspective transform is carried out with the four points of paper to destination point.
- Finally, the transformed image is given the black and white effect with the Thresholding method.

1.5 Flow Diagram



1.6 Implementation

This image processing Project is implemented in Python 2.7 with Opencv 3.2 Library.

1.7 References

[1] OpenCV-Python Tutorials Documentation -Release 1: Alexander Mordvintsev & Abid K

[2] PyImage Search, Adrian Rosebrock https://www.pyimagesearch.com/

[3] Stack Overflow: https://stackoverflow.com/

[4] IP Webcam, Codacus
https://thecodacus.com/ip-webcam-opencv-wireless-camera/#.WiTL5VWWbIW

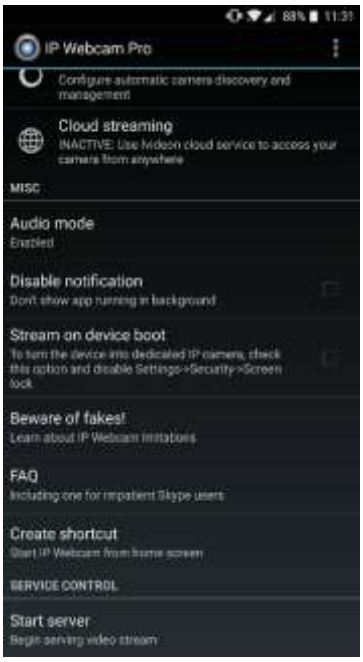
[5] OpenCV Tutorials, Robin David http://www.robindavid.fr/opencv-tutorial/

[6] Wikipedia: https://en.wikipedia.org/wiki/Main_Page

[7] Tutorials Point: https://www.tutorialspoint.com/

2 Capturing the Image from IP Webcam

This one will help to create our android phone as web camera. First we need to install IP Webcam in the android phone. It will give us the application interface to access the video streaming server. Then python script can grab the video stream and it can capture the frame as jpg image from the video stream.



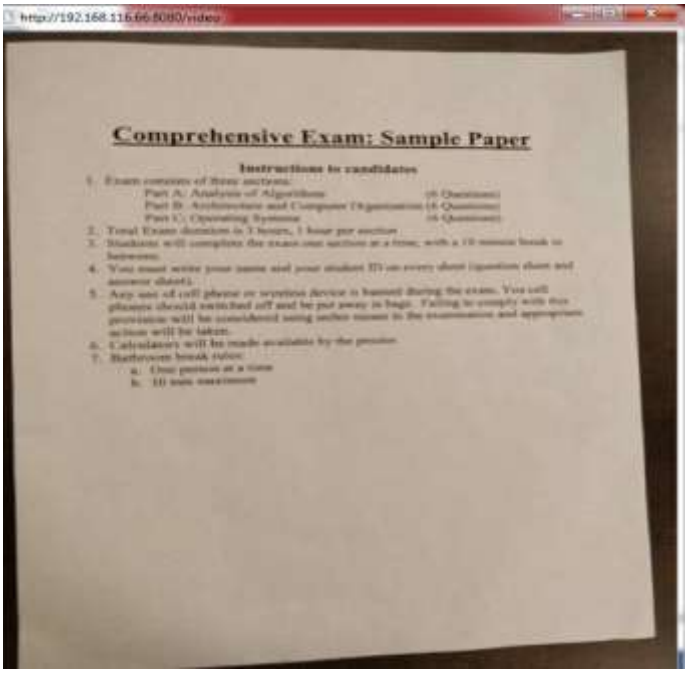


Figure: Streaming from IP Webcam

This application opens the webcam with the streaming address.

```
stream = urllib2.urlopen(streamurl)
```

Above code opens the http video stream from the url.

```
buffer += stream.read(1024)
```

The Stream reads the bytes of data. Here we treat bytes as buffer of size (1024), which acts as queue that keeps growing. Every jpeg frame starts with string 0xff and 0xd8, ends with string 0xff and 0xd8. This jpeg frame is parsed from the http stream in the loop.

```
x = buffer.find('\xff\xd8')  # Finds the beginning of jpg frame
y = buffer.find('\xff\xd9') # Finds the end of jpg frame
jpgframe = buffer[x:y+2] # Only Grab the jpg frame from http stream
```

Now we have to decode the jpg frame string into numpy array.

```
# Decodes the string image from string to NumPy Image array format
img = cv2.imdecode(np.fromstring(jpgframe, dtype=np.uint8), -1)
```

Then we can finally display the image frame by frame in the loop.

```
cv2.imshow(streamurl, img)
```

Then we, capture the specific image frame using the key ‘c’ and then return the captured image as source image.

```
if cv2.waitKey(1) == ord('c'):
    return img
```

3 Edge Detection

It's the preprocessing steps before we introduce the image for contour detection. Under the edge detection we follow the following steps.

First we convert the image from BGR channel to Gray scale image. Then we blur the image using the Bilateral Filter. Bilateral Filter is very effective in reducing the noise and keeping the edge sharp. But its slower than other filter. Then we apply otsu Thresholding to the image we applied the bilateral filter. Then we apply the canny edge detection which gives the edged detected in the image.

3.1 Process of Canny Edge Detection

It's popular edge detection multistage algorithm.

1. First step is the reduction of noise. It reduces the noise by using Gausian Filter of 5 x 5 kernel size.
2. Second step is to find the intensity gradients of the image. Here the Gaussian filtered image is then filtered with sobel kernel to get derivative in both horizontal and vertical direction. Then with these two image, edge gradient and direction of each pixel is found
3. Third Step is non-maximum suppression to remove unwanted pixel.
4. Final step is hysteresis Thresholding which determines which edges are and which are not. Edges with intensity gradient more than maxVal are edges and those below minVal are eliminated. Those in between are decided based on their connectivity.

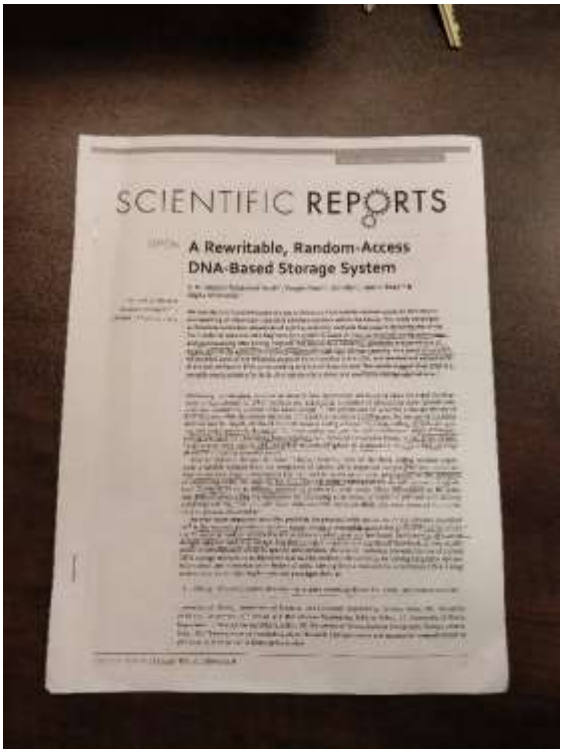


Figure: Source Image

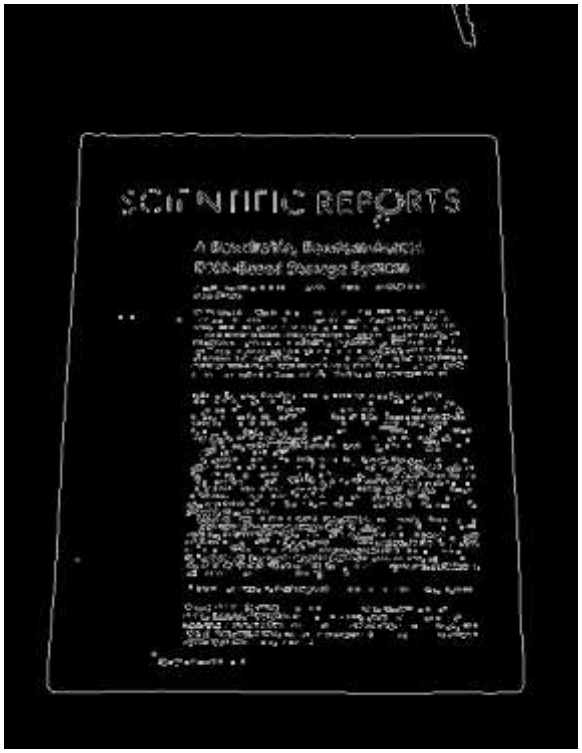


Figure: Edged Image

4 Contour Detection

What are Contours?

Contours are simply an outline representing the bounding object. Technically, it can be explained as the curves joining all the points along the boundary with similar intensity. To detect contour while doing image processing, for better accuracy, its better if image is thresholded or canny edge detection is applied. While detecting contour, it's important to note that the object is to be white and background should be black.

The aim is to find the largest contour with the shape of rectangle with 4 corner points. For this we follow the steps:

Contour in opencv is detected using findContours function:

```
cv2.findContours(image, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

Then the detected contour is subjected to function 'contourArea()'

The key is to find the contour with large area and then we can perform it to contour approximation to find the 4 corner points.

```
perimeter = cv2.arcLength(cnt,True)
# Epsilon is the maximum distance from contour to approximated contour
epsilon = 0.05 * perimeter
approx = cv2.approxPolyDP(cnt,epsilon,True)
```

Then the found contour is estimated to be the corner points of page in the image.

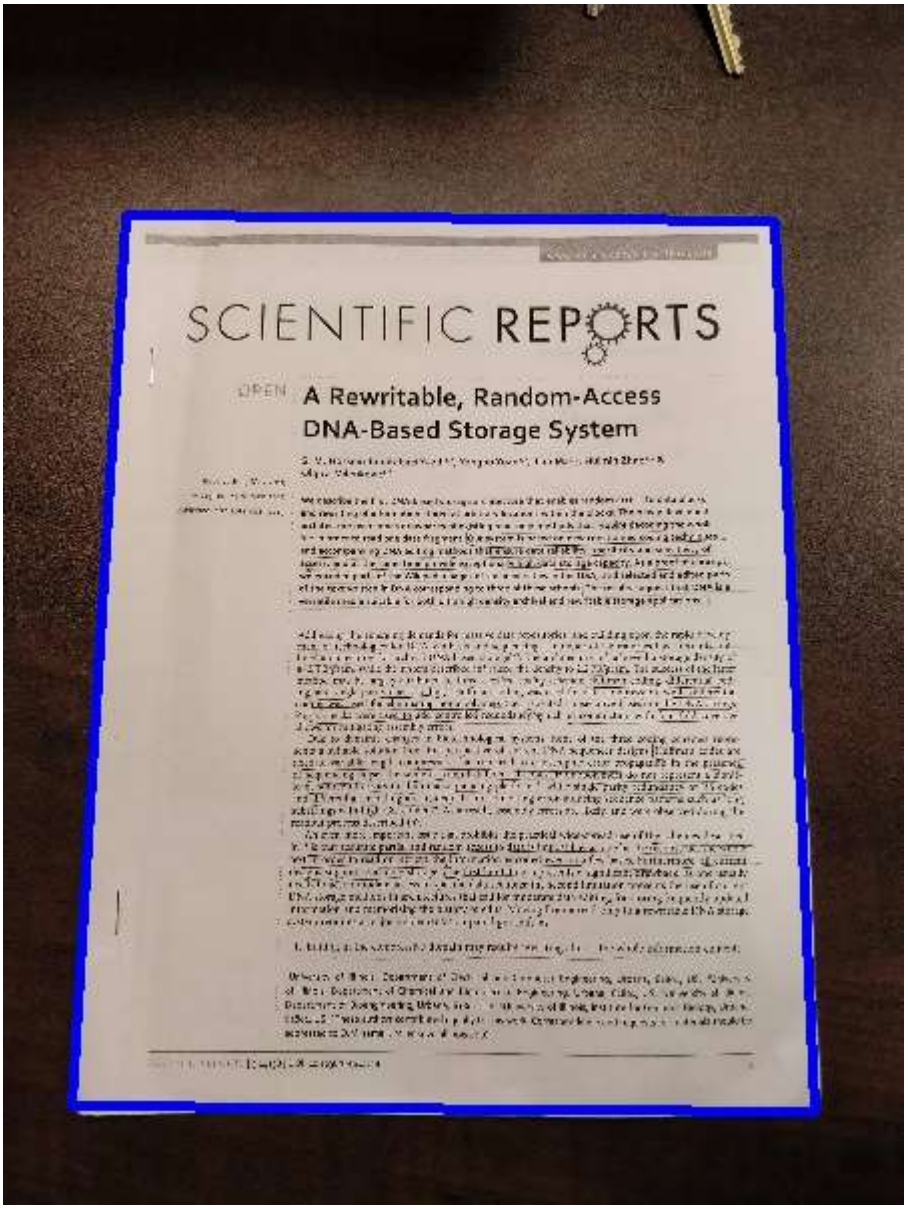


Figure: Contour Outline

5 Understanding Perspective Transformation

A Perspective transformation can be specified with 4 corresponding points. It can be carried out by dragging any four corner points of the image and place the image to any destination point we prefer.

To perform perspective transformation we need 3 x 3 transformation matrix. Transformation matrix can be found by OpenCV function: ***getPerspectiveTransform*** function. Then, ***warpPerspective()*** function is applied with 3 x 3 transformation matrix.

To understand perspective transformation, let's consider an example of 4 corner points of page which was detected as contour points.

4 Corner points of paper in an image are: [[316,104] [19,116] [9,506] [330,517]]

Now, we have to figure out which is the top left, top right, bottom left, and bottom right corners.

So, for this it is necessary to, find the sum between x and y and difference between x and y.

The point with max difference obtained from y-x is Top Left Corner (TL)

The point with max sum of x and y is Top Right Corner (TR).

The point with min sum of x and y is Bottom Left Corner (BL).

The Point with max difference obtained from y-x is Top Left Corner (BR).

X Point	Y Point	Sum (Y+X)	Difference(Y-X)	Addressed Corner
316	104	420	-212 (Min Diff)	Bottom Right (BR)
19	116	135 (Min Sum)	97	Bottom Left (BL)
9	506	515	497 (Max Diff)	Top Left (TL)
330	517	847 (Max Sum)	187	Top Right(TR)

Now, the exact corner point to be addressed are found. These points can be considered as 4 corner source point to be transformed into destination point.

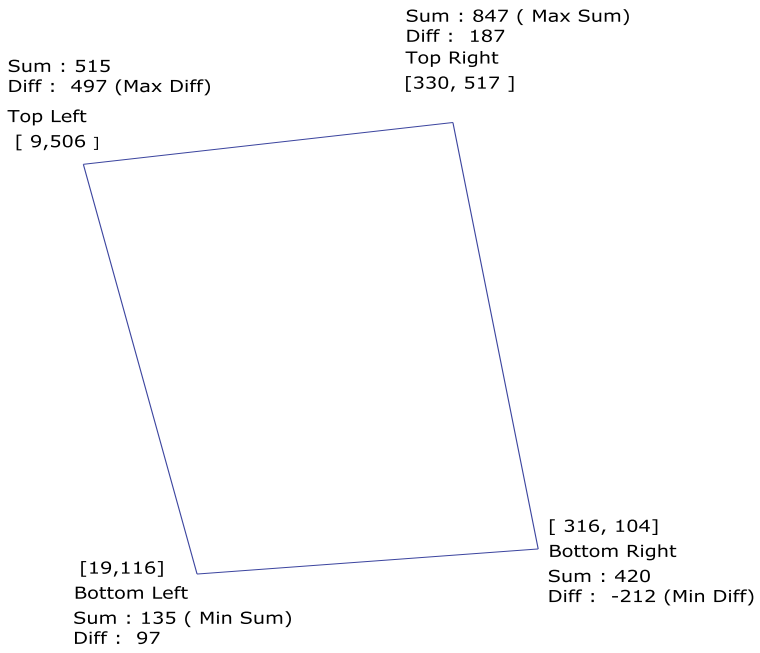


Figure: 4 Corner Points

Next step is to find the destination point where these source points are going to be dragged.

For that, we need to find the height and width of the source object using this corner points. To find the height and width, the distance between two (x,y) points need to be considered. Distance between any two points can be calculated using Euclidean distance measure:

Euclidean Distance (Point 1, Point 2) = $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

Height 1: Euclidean Distance (TL, BL)

Height 2: Euclidean Distance (TR, BR)

Width 1: Euclidean Distance (TL, TR)

Width 2: Euclidean Distance (BL, BR)

Destination Height (H): Max (Height 1, Height 2)

Destination Width (W): Max (Height1, Height 2)

Then the Destination point can be calculated as desired:

New Top Left: (0, H)

New Top Right: (W, H)

New Bottom Left: (0, 0)

New Bottom Right: (W, 0)

This way the respective corner points from source point is transformed to new destination points.

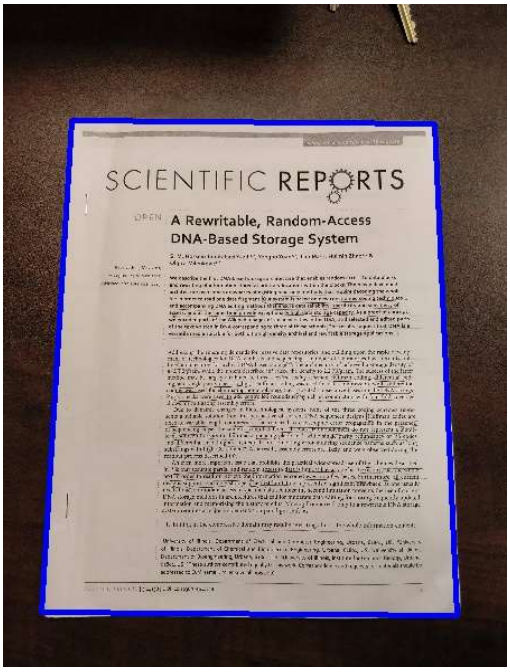


Figure: Contoured Image

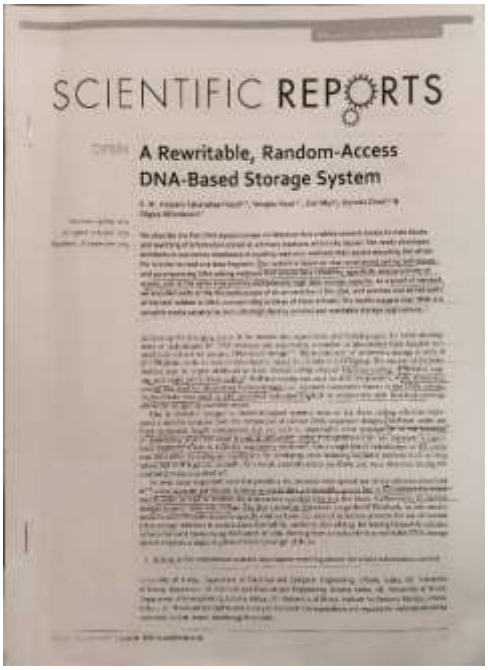


Figure: Transformed

6 Thresholding

Thresholding is the process of image segmentation. It converts the gray scale image into binary image. That is only black and white image. With Thresholding, there is a threshold value of intensity. If the image intensity is less than specified threshold intensity, then it is considered as black pixel, else if image intensity is greater than specified threshold than it is considered as white pixel.

Here in the project, Otsu Thresholding Technique is applied to give the black and white effect to the final transformed image. Otsu Binarization calculates the optimal threshold value.

SCIENTIFIC REPORTS

A Rewritable, Random-Access DNA-Based Storage System

Figure: Binarized Image