CamScanner

implemented in python using openCV

Mahendra Nandi

Assignment 3
Semester 2

Big Data Analytics
RKMVERI
India
14 - 03 - 2021

Almost all of us have used CamScanner in our lives. It is a very effective app which allows users to scan documents from your mobile and share it as an image. The biggest advantage of the application is that it 'cleans' (denoising, rotation, sharpened, etc) up a camera-clicked image into a very refined output. In this project we are going to create our own CamScanner using the basics of OpenCV in Python

1 Preprocessing:

1.1 BLURRING:

The goal of blurring is to reduce the noise in the image. It removes high frequency contentfrom the image — resulting in blurred edges. I have used *cv2.GaussianBlur* which convolutes the gaussian kernel to blur the image.

2 Canny Edge detection & Extraction of biggest contour:

After image blurring thresholding, the next step is to find the biggest bounding box and crop out the image. This is done by using Canny Edge Detection followed by extraction of biggest contour using four-point transformation.

2.1 CANNY EDGE:

Canny edge detection is a multi-step algorithm that can detect edges. We should send a de-noised image to this algorithm so that it is able to detect relevant edges only.

2.2 FIND CONTOURS:

After finding the edges, pass the image through **cv2.findcontours()**. It joins all the continuous points (along the edges), having same colour or intensity. After this we will get all contours — rectangles, spheres, etc

2.3 EXTRACTING THE BIGGEST CONTOUR:

Although we have found the biggest contour which looks like a rectangle, we still need to find the corners so as to find the exact co-ordinates to crop the image. For this first you pass the co-ordinates of the approx rectangle(biggest contour) and apply an order points transformation on the same. The resultant is an exact (x,y) coordinates of the biggest contour.

while doing this whole extracting the biggest contour process we have used three functions - *cv2.contourArea, cv2.arcLength, cv2.approxPolyDP*. The first

function returns the contour area on based on which we have sorted the list of contours. then we need to extract the top contour with biggest perimeter which is approximated by the last function.

2.4 Four Point Transformation :

Using the above (x,y) coordinates, calculate the width and height of the contour. Pass it through the **cv2.warpPerspective()**to crop the contour. warpperspective uses the homography matrix which transforms the original image space to the final one. To get the homography matrix we have used the function *cv2.findHomography*

3 RESULTS:

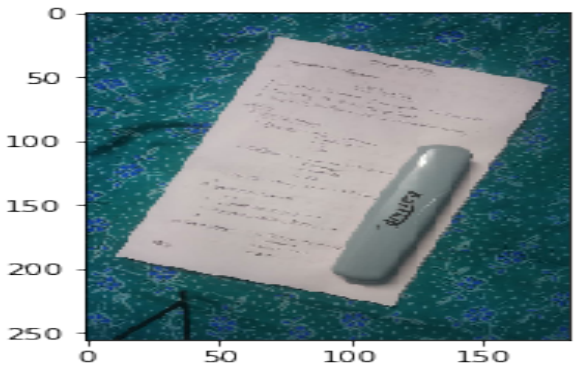

Figure 1: Original Image

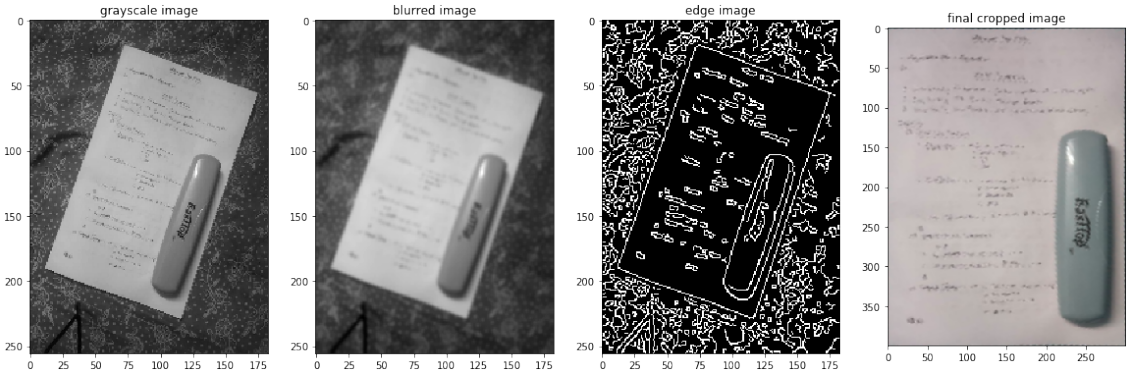

Figure 2: multiple steps related to scanning the actual image