An algorithm to improve quality of image perception

[1]Deepa Narayanan
[1]Student
[1] Department of Computer Science,
[1]Vivekanand Education Society's Institute of Technology, Mumbai, India

Abstract : OpenCV provides an inbuilt functionality which improves the perception of an input image. However, if the redundancy in the background is greater than the ROI (Region of Interest), the perception algorithm in OpenCV may give loss of accuracy in the image as the output. For example, while scanning the image of a passbook. Main purpose of this research is to improve the quality of image perception by extracting the major coordinates that surround the ROI and supplying the same to the perception algorithm given by OpenCV. Various image processing techniques have been applied in this paper on the sample test image and a combination of them provides dramatic results. The different algorithms used are also compared and outcomes of the same are tabulated to provide optimum results.

IndexTerms **- ApproxPolyDP, contours, Image Perception, masking, OpenCV, threshold**

I. INTRODUCTION

Digitalization is progressing at a rapid rate. It has become mandatory to convert the manual documents in digital form. For simplifying the same, OCRs have been used, which help people to convert text from images to type written format. However, this has to be done by scanning the documents properly and then supplying it to the OCR tools. As humans, it is not expected to see much of perfection from the user who is scanning the document with his camera lens and supplying to the OCR for textual conversion. Perception is an important aspect when it comes to image correction and it captures a 3D image in 2D frame. While correcting the perception of an image it is important to note that the ROI is not lost and at the same time, get rid of redundant background. The research begins with exploring the primary requirements of what the perspective transformation requires to correct the input image. Drawing contours around our ROI helps to determine which portion of image should be retained and which should be shun off. By default OpenCV demands a 3x3 transformation matrix, where straight lines are left uncorrected and slanted lines are tried to get in alignment with the 2 dimensional axis. It is mandatory to supply the 4 coordinates to the algorithm out of which 3 ought to be non linear. This ensures that the coordinates are the 4 end points of our ROI. Manually entering the coordinates to correct the image may seem a bit tedious as it deals with pixel level coordinates. This research aims at defining a process by which, we are able to locate the contour of our ROI and thereby self-generate the four coordinates for supplying it to the perception algorithm. Proposed in this paper is the methodology employed to extract the ROI..

II. PROPOSED METHODOLOGY

A series of image processing steps were applied to build this algorithm. This proposed system is compared with other alternatives for comparison purposes.

The methodology for the proposed system is as follows

1) Grayscaling

Gray scaling of images are done to reduce the complexity. In this method, we convert the colored images to grayscale form to define the edges prominently for applying algorithms like canny edge detection and masking. In order to reduce computational complexity the grayscale representations are often in place of operating on color images directly [2]. In image processing, gray scaling helps us to improve the efficiency by providing focus on our real time application than dealing with complex colors in the RGB. All interfaces like Matlab and Python provide simplified development in grayscale mode. [1] [6]
The simple in built function provided by OpenCV,

imgray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY) …eq (1)

where im is the name of input image and the imgray is the variable
This equation will convert original image to gray scale

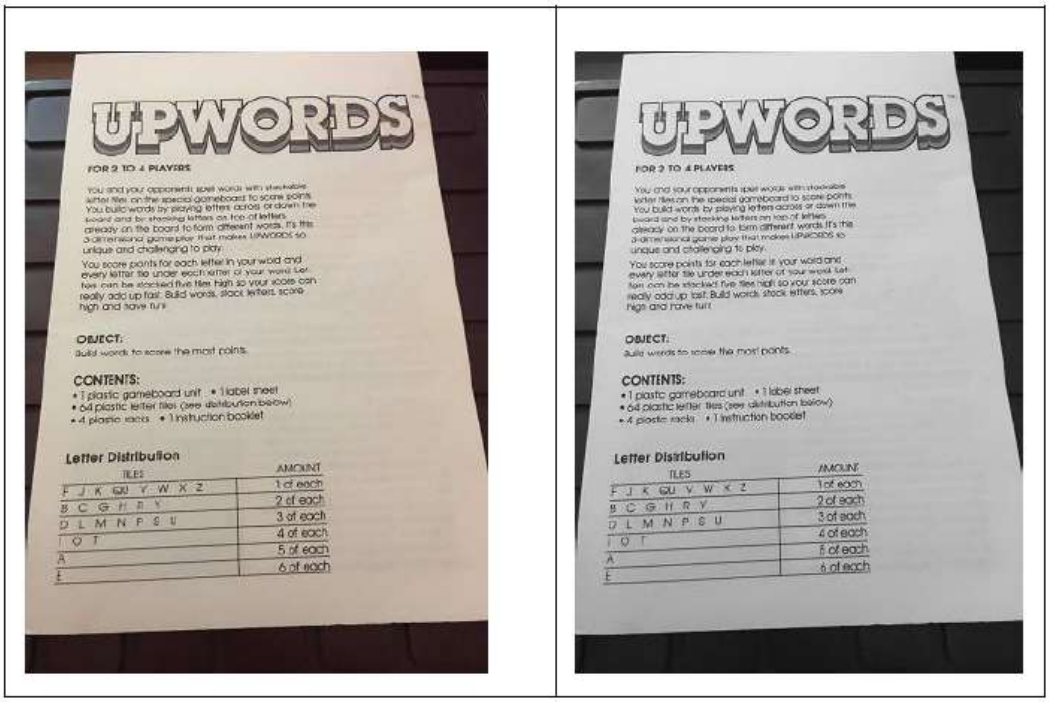


Fig.1: Grayscale image

2) Adaptive threshold

Adaptive Thresholding results in binarization of images by depicting variations in different threshold values.[6] A threshold value is defined beforehand and those which exceed the set value are described by black and the rest by white. [2] This method is useful in performing segmentation by fixing the pixels of images depending on the intensity values and threshold value and is applied on the gray scale of original image. Here, the spatial image pixels can be changed over dynamically and it can be used to calculate the threshold for smaller regions in the images. [3]

The adaptive threshold value to applied using the following function,

thresh=cv2.adaptiveThreshold(imgray,255,cv2.ADAPTIVE_T HRESH_MEAN_C,cv2.THRESH_BINARY,51,2) …eq (2)

where, imgray is the gray scaled image as mentioned in equation (1), ADAPTIVE_THRESH_MEAN_C is the threshold value from the mean of neighborhood area, the maximum value of which is 255. The block size decides the size of neighborhood area. In this case 51 is the block size calculated from it and 2 is the constant value subtracted from the threshold.

The image from equation (1) is converted to binarized image with the help of Adaptive Thresholding as in,

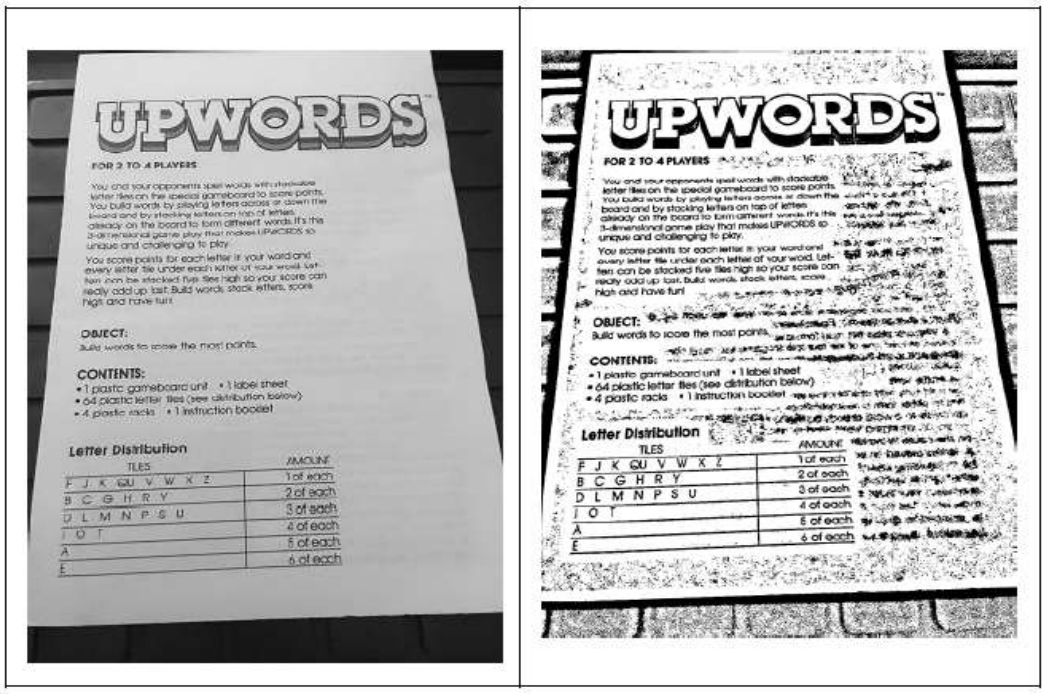


Fig.2: Binarization

3) Contour generation

3.1 Contours

Contours are defined as a curve which traces the boundary of objects having the same color intensity. After successfully binarizing the image, now it is time to trace out the edges of our ROI. An image may contain redundant data in the background as well. It is majorly used to differentiate all the distinct objects in the given image. Better results are generated on binary images, hence the above step has been recommended. During this process contours are generated around every small complete object that may not be required by us. However, we just need to concentrate on the largest contour generated that will by default be our ROI. Contour generation can be improvised by removing the noise from the image. For this some filter algorithms can be used. [4]
The following function has been used to get optimal outline along the ROI

contours,hierarchy=cv2.findContours(thresh,cv2.RETR_EXT ERNAL,cv2.CHAIN_APPROX_NONE) …eq (3)

Every contour generated is stored as a vector of points after detecting them, Numpy array of (x, y) coordinates. [6] The information topology is stored in the hierarchy parameter which decides if a generated contour has any parent or child. The input image is taken from the previous step, thresh. RETR_EXTERNAL flag returns the extreme outer flags only. This may help us to get rid of some minor redundant portions in the image which do not belong to our ROI or form a separate region inside the ROI. CHAIN_APPROX_NONE has been used to extract all the boundary points of the contours. Instead we could also use CHAIN_APPROX_SIMPLE which could directly give us 4 coordinates of the boundary points, but as mentioned earlier, considering our image has much redundant data in the background, the coordinates so generated were very far or within the ROI thereby generating data loss.

3.1 Extracting the largest contour

The largest contour is recovered in the following format,

largest_areas = sorted(contours, key=cv2.contourArea)
cnt = largest_areas[-1] …eq (4)

Out of all the generated contours, we will concentrate on the largest one as that will be the ROI or the scanned document. largest_areas will contain all the regions arranged in increasing order of their areas, out of them we extract the largest one that is, the last in sorted list. cnt contains a collection of points in the form of numpy array. [6]

4) Masking and plotting

In order to remove the actual contouring boundary and get the ROI as a continuous object, we apply mask. This has been used to recalculate each pixels value in an image so that the inner contours that may still exist as a separate entity may merge with the major ROI, thereby leaving a masked ROI according to a mask matrix. This mask holds values that will adjust how much influence neighboring pixels (and the current pixel) have on the new pixel value. The image is then cropped out of the mask and stored in black and white format. This is the masked portion for sample image, Figure (3).

Fig.3: Masking and plotting

4.1 ApproxPolyDp

ApproxPolyDp is used to improve the coordinate plotting precision and mark out circles along the edges. In order to print more points along the contours, ApproxPolyDp and epsilon (0.01 gives better results than 0.1) is used. ApproxPolyDp, helps to approximately points forming a near-to-perfect polygon. It is used for contour approximation. Depending upon the precision specified, it approximates a contour shape to another shape with less number of vertices. The maximum distance from contour to approximated contour is denoted by an epsilon which is supplied as the second parameter. It is an accuracy parameter.[6]

```
epsilon = 0.01*cv2.arcLength(cnt,True)
approx = cv2.approxPolyDP(cnt,epsilon,True)
```
…eq (5)

4.2 Four Point coordinate plotting

The requirement is only for 4 points from the above plotted points to give it to the perspective algorithm. Hence it is a must to filter out them. 2 approaches were tried :

4.2.1 To find the min max from x and y coordinates and generating 4 points.

The coordinates extracted in approx is stored in the form of numpy array. The maximum and minimum of X and Y coordinates are noted from this array to give the four extreme coordinates.

```
cv2.circle(im,(xmin,ymin),10,[255,0,102],-1)
cv2.circle(im,(xmin,ymax),10,[255,0,102],-1)
cv2.circle(im,(xmax,ymin),10,[255,0,102],-1)
cv2.circle(im,(xmax,ymax),10,[255,0,102],-1)
```

4.2.2 Find the point that either lies inside or on the contours to get more precision.

```
for points in approx:
        x1 = points[0][0]
        y1 = points[0][1]
        dist = cv2.pointPolygonTest(cnt,(x1,y1),False)
if dist == -1 or dist == 0:
        cv2.circle(im,(x1,y1),15,[43,255,0],-1)
        print("Green Coordinates")
        print(x1,y1)
```

III. CANNY EDGE DETECTION AND HULL DEFECT CONVEXITY

In order to test with various other algorithms, two more image processing steps were tried, namely Canny Edge Detection and Hull Defect Convexity. The Canny edge detection algorithm can be initiated by smoothening the image with Gaussian filter followed by computing gradient intensity representation. It is a multi-stage algorithm and we will go through each stages such as noise reduction, finding intensity gradient of the Image…etc. Tracking of edges by suppressing the weak one and highlighting the strong ones is done with the help of Canny Edge Detection. Hull Convexity is used to trace all points that may give a complete polygon. This helps in determining the skeleton of the ROI. The algorithm checks for any defects along the convex space and corrects by reforming the bulges if any. [6]

```
hull = cv2.convexHull(cnt,returnPoints = False)
defects = cv2.convexityDefects(cnt,hull)
```
…eq (6)

IV. COMPARISON

Other methods were tried as well, however, the masking of the above was optimum. Testing of the same was done on 14 images. Various methods included applying Canny Edge Detection with 0.01 epsilon value, Hull Defect Convexity, Hull Defect Convexity with 0.01 epsilon value and lastly Adaptive Threshold with masking on the gray scale image as shown,

Image no.	Canny Edge + Epsilon	Hull Defect Convexity	Hull Defect Convexity + Epsilon	Adaptive Thresholding (Optimum)
1	No edges detected	*(1 line detected)	*	*
2	*	*	*(Hull better)	*
3	*	*	*	*
3-crop	*	*	*	*
4	*	*	*	*
5	OK	*	*	*
6	*	*	*	*
7	*	-	-	*
7-1	*	-	OK	*
7-bottom	*	-	OK	*
7-top	*	OK	-	*
8	OK	Line Detected*	-	*
9	Hull better	*	Hull better	*
10	*	*	*	*

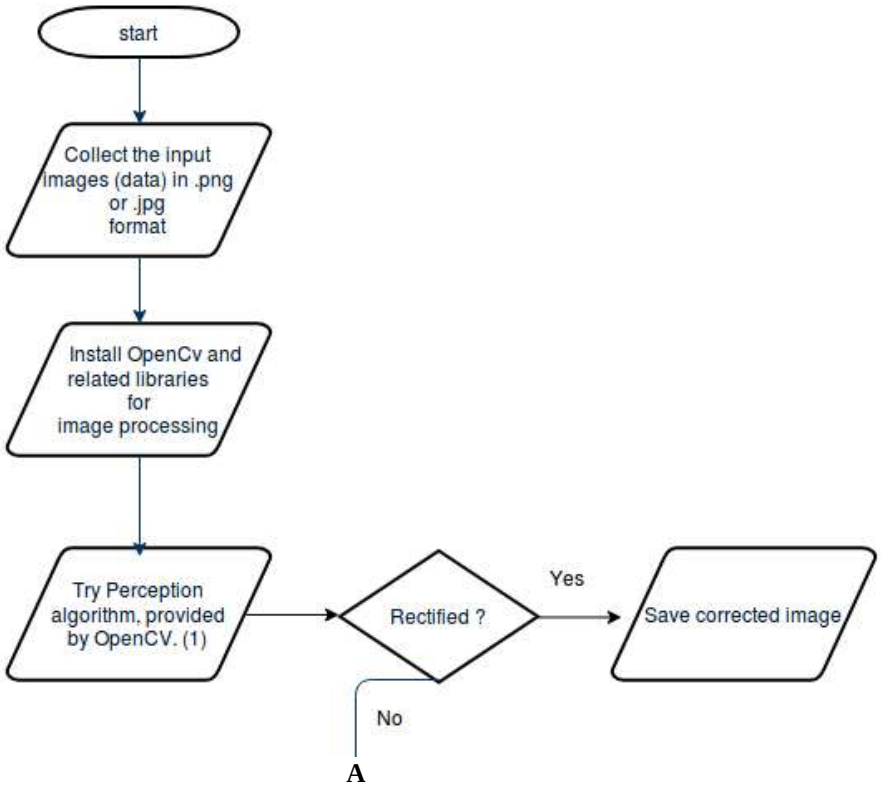
* Mask can be accepted.

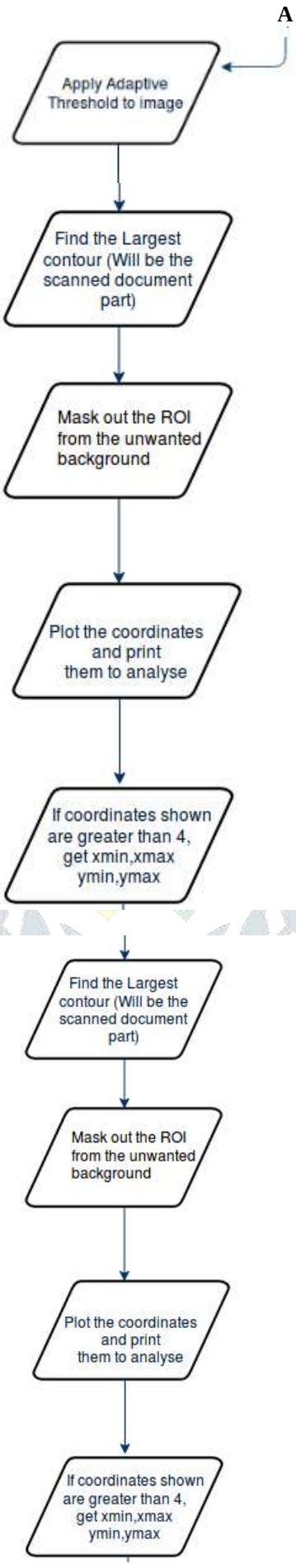
- Very poor masking

OK Can be Improved

Table IV: Comparison with test images

V. FLOWCHART




A
Apply Adaptive Threshold to image
Find the Largest contour (Will be the scanned document part)
Mask out the ROI from the unwanted background
Plot the coordinates and print them to analyse
If coordinates shown are greater than 4, get xmin,xmax ymin,ymax
Find the Largest contour (Will be the scanned document part)
Mask out the ROI from the unwanted background
Plot the coordinates and print them to analyse
If coordinates shown are greater than 4, get xmin,xmax ymin,ymax

B

B

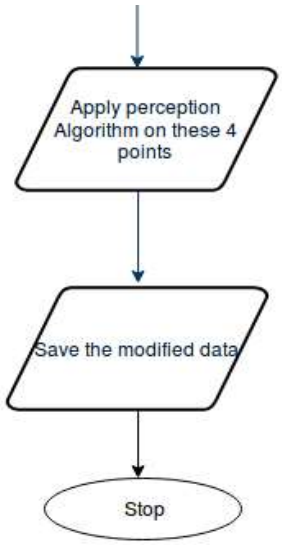


Fig.3: Flowchart

VI. PERCEPTION ALGORITHM

The 4 coordinate points generated with the help of above algorithm needs to be supplied to the perception algorithm as shown below along with the dimensions of output image. Applications of this feature can be extended to supplying the corrected image to the OCR so that there is minimum data loss and more accuracy in detection of text.[6]

points1 = np.float32([[x1,y1],[x2,y2],[x3,y3],[x4,y4]])
points2 = np.float32([[x1',y1'],[x2',y2'],[x3',y3'],[x4',y4']])
final_img = cv2.getPerspectiveTransform(points1,points2) …eq (6)

VII. CONCLUSION

Thus, using image processing this research has helped us to successfully generate an algorithm which will accept the image and inputs and mark our 4 corner points from the ROI which can be further given to the perception algorithm. Various steps mentioned in the above procedure include a series of image processing with was applied on 14 different images to study the influence of combination of different techniques as mentioned in analysis chart. The block size in adaptive threshold needs to be changed as per the requirement of our document and the noise level in the image and neighborhood pixels can affect the binarization of image.

REFERENCES

[1] Khobragade, Kavita. (2012). A Comparative study of Converting Coloured Image to Gray-scale Image using Different Technologies. (references)
[2] Kanan C, Cottrell GW (2012) Color-to-Grayscale: Does the Method Matter in Image Recognition? PLoS ONE 7(1): e29740. 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)
[3] P. Roy, S. Dutta, N. Dey, G. Dey, S. Chakraborty and R. Ray, "Adaptive thresholding: A comparative study," 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kanyakumari, 2014, pp. 1182-1186.
[4] C. Huang, D. Chen and X. Tang, "Implementation of Workpiece Recognition and Location Based on Opencv," 2015 8th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, 2015, pp. 228-232.
[5] Suzuki S. and Abe K., Topological Structural Analysis of Digitized Binary Images by Border Following.Computer Vision Graphics and Images Processing,1985,vol.30, pp.32-46.
[6] "The OpenCV Reference Manual Release 2.4.9.0",2014.