The Realization of Document Scanner

Boya Zhang
Vrije Universiteit Brussel EE

Divyanshi Awasthi
Universite libre de Bruxelles EE

Xiaotian Liu
Vrije Universiteit Brussel EE

Xiaojian Zheng
Vrije Universiteit Brussel EE

***Abstract*—In this article we first discussed about the methods to achieve a document scanner in a static environment. Then we applied the methods from static environment to a real-time webcam. We used methods like Canny Edge Detection and Geometric Image Transformations. The results show that we achieve the basic and advanced requirements of the project.**

***Index Terms*—Document Scanner, Real-time Scanner, Edge Detection, Document Transforming**

I. Introduction

The objective of this project is to convert an image of a document (that is placed on a flat surface) into a high quality scanned version. We used python together with many modules as our coding tool.

II. Basic Algorithm

This part of report includes the description of all the algorithms which play important roles in the implementation of this scanner project.

A. Smoothing Images

Gaussian filters have the properties of having no overshoot to a step function input while minimizing the rise and fall time. In terms of image processing, any sharp edges in images are smoothed while minimizing too much blurring.
The Gaussian blurring algorithm [1] will scan over each pixel of the image, and recalculate the pixel value based on the pixel values that surround it. The area that is scanned around each pixel is called the kernel. A larger kernel scans a larger amount of pixels that surround the center pixel. Gaussian blurring doesn't weigh each pixel equally.
However, the closer a pixel is to the center, the greater it affects the weighted average used to calculate the new center pixel value. This method assumes pixels closest to the center pixel will be closest to the true value of that pixel, so they will influence the averaged value of the center pixel greater than pixels further away.

B. Canny Edge Detection

The Canny edge detector [2] is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images.
Canny edge detection algorithm has five main stages within it:

1. First Gaussian filter is applied to remove the noise by smoothing the image.
2. Then the intensity gradients of the image are determined An edge in an image may point in a variety of directions, so the Canny algorithm uses four filters to detect horizontal, vertical and diagonal edges in the blurred image.

$$Edge\_Gradient(G) = \sqrt{G_x^2 + G_y^2} \tag{1}$$

$$Angle(\theta) = tan^{-1}(\frac{G_y}{G_x}) \tag{2}$$

3. Non-maximum suppression is applied to remove deceptive response to edge detection. It is an edge thinning technique which is applied to find "the largest" edge.
4. Double threshold is applied to determine potential edges by filtering out edge pixels with a weak gradient value and preserving edge pixels with a high gradient value.
5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges. The edge A is above the maxVal, so considered as "sure-edge". Although edge C is below maxVal, it is connected to edge A, so that also considered as valid edge and we get that full curve. But edge B, although it is above minVal and is in same region as that of edge C, it is not connected to any "sure-edge", so that is discarded.

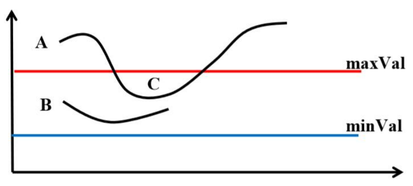

Fig. 1. Canny edge detection

C. Contour Features

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity [3]. The contours are a useful tool for shape analysis and object detection and recognition.

It outputs the contours and hierarchy. contours is a Python list of all the contours in the image. Each individual contour is a Numpy array of (x,y) coordinates of boundary points of the object.The function retrieves contours from the binary image using the algorithm $[Suzuki85]$ [4].

D. Image type detection

The color value of image is detected to distinguish the image is colorful one or black-white one. The *ImageStat* module is employed to calculate the variance for each RGB tone in the image. After getting the variance, the following process is executed:

$$Maxmin = abs(max(v) - min(v)) \tag{3}$$

$$dst(x,y) = \begin{cases} color\ one & \textit{if Maxmin} > \textit{COLOR} \\ black\ and\ white & \textit{otherwise} \end{cases} \tag{4}$$

If the variance over three tones stay close, i.e.smaller than the figure Color which we set it to 100 as it is a empirical number, we can say the figure is white and black image.

E. Enhancement

Once the image is black and white, a threshold is applied: if pixel value is in a certain threshold, it is assigned to 1, otherwise it is assigned 0

$$image = \begin{cases} 1 & \textit{if scr(x,y)} > \textit{166} \\ 0 & \textit{otherwise} \end{cases} \tag{5}$$

When the image is the color one, two important properties are adjusted by function $ContrastandBrightness$. Brightness decides the absolute value of colors (tones) lightness/darkness while Contrast aims to make objects more obvious within an image.
By increasing brightness of an image, all colors of original pixel will be lighted out. Reversely, all colors gets darker by decreasing brightness.
Similarly, bigger contrast increases the difference between light and dark areas and smaller contrast makes global areas stay approximately and becomes more flat.

F. Median Filter

The Median Filter is employed to remove noise from an image while keeping the most information of edge. As median filter has the properties to preserve the edge, such noise reduction will improve the results of later processing.
Considering the processing such as filtering will arise some information lost problems, an enhancement operation is desired to argument the image.

G. Geometric Image Transformations

It is the most important function on which our algorithm depends. The purpose of writing this function is to convert any irregular shape of image into the shape of a document. The main logic behind it to compare the two different lengths and two different widths of the image and find the maximum length and width. Then we take these maximum values and fix them as reference size for the scanned document.

$$maxWidth = max(int(widthA), int(widthB)) \tag{6}$$

where

$$\begin{aligned} widthA &= np \cdot \sqrt{(tr[0] - tl[0])^2 + (tr[1] - tl[1])^2} \\ widthB &= np \cdot \sqrt{(br[0] - bl[0])^2 + (br[1] - bl[1])^2} \end{aligned} \tag{7}$$

$$maxHeight = max(int(heightA), int(heightB)) \tag{8}$$

where

$$\begin{aligned} heightA &= np \cdot \sqrt{(tr[0] - br[0])^2 + (tr[1] - br[1])^2} \\ heightB &= np \cdot \sqrt{(tl[0] - bl[0])^2 + (tl[1] - bl[1])^2} \end{aligned} \tag{9}$$

And new points are as follows:

- $[0, 0]$
- $[maxWidth - 1, 0]$
- $[maxWidth - 1, maxHeight - 1]$
- $[0, maxHeight - 1]$

For perspective transformation, you need a 3×3 transformation matrix. Straight lines will remain straight even after the transformation. To find this transformation matrix, you need 4 points on the input image and corresponding points on the output image. Among these 4 points, 3 of them should not be col-linear. Then transformation matrix can be found by the function $cv2.getPerspectiveTransform$.

$$\begin{bmatrix} t_i x_i^{'} \\ t_i y_i^{'} \\ t_i \end{bmatrix} = map\_matrix \cdot \begin{bmatrix} x_i \\ y_i \\ l \end{bmatrix} \tag{10}$$

where $dst(i) = (x_i^{'}, y_i^{'})$, $src(i) = (x_i, y_i)$, $i = 0, 1, 2, 3$

Then apply cv2.warpPerspective with this 3×3 transformation matrix as follows.

$$dst(x,y) = src\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}\right) \tag{11}$$

III. IMPLEMENTATION

In this section, the implementation will be explained into details. The project is implemented in Python language [5]. With the powerful help of module and package, the purpose of edge detection and other requirements could be achieved. Two flow charts are introduced followed by expositions of Python code.
Our code can be divided into three main parts: Main, Scan and Webcam. Main file basically operates following the flow chart while we define all the functions in the scan file. At last to realize the bonus, a Webcam is designed.

A. Document Scanner

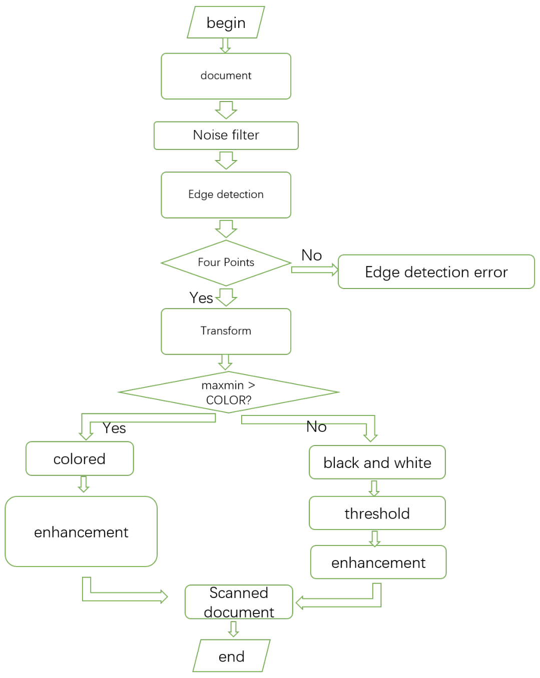

Fig. 2. The document scanner

The process of main code is displayed in the Figure2, we first import the scan. The image is read and resized in order to apply the display window.
After the image is read, the noise filter is used in order to remove the noise so that we can have a better edge detection [6]. As stated before, the kernel size of Gaussian filter here is 5×5, where the center pixel is the pixel that will be changed with respect to the surrounding 24. What's more, there is another argument sigma which dictates the width of the curve in the X and Y directions. We set sigma to zero here so that it will be auto-calculated from the kernel size.
The next step is edge detection. It is implemented by applying the Canny algorithm as explained previously. The lower and upper limit of threshold values are defined as 75 and 200, respectively [7].
After results of edge detection remains all the possible edges of our images, it is a good idea to retrieve all the outlines from the binary image and sorted them by the importance. Firstly, the $cv2.findContours()$ function is used to generate a list of all the contours, each individual contour is a Numpy array of (x,y) coordinates of boundary points of the object [8]. The third argument of this function is chosen as contour approximation method because it only retain end points of the line instead of number of useless points, thus saving a lot of memory.
To compare the importance among all the contours, we reorder them on the basis of area(size) of contour from largest to smallest. This step is implemented by $contourArea()$ function.
As all the contours are found and sorted, we need to check over the contours to see whether they can form a closure, or just a curve. This is done by Calculating contour circumference using $arcLength()$ function. After approximating the shape of contour by fixing the relevant parameter at 10 percent, the edges of the document are obtained. At the end if the edge is found correctly then it returns the approximated value otherwise zero.
After that, we employ the transform function to reshape the figure and project it to a new viewing plane. As explained in the Algorithm section, the maximal width and maximal height are checked first and applied as the arguments of $cv2.getPerspectiveTransform$. The obtained image now is in a rectangular fashion, however, the quality of the image decreases somehow due to the series of processing. Thus the enhancement is desired.
If the document is a black and white one, we threshold it firstly then enhance it. If the document is a colored document, the enhancement is employed by adjusting the contrast and brightness properties to reach a better intelligibility. In our implementation, the enhancement contrast parameter is 1.1 for both of the black & white and colored document. For brightness parameter, we set 3 to Black & white case and colored case. Finally we output the results and compare it with the original one.
Here we illustrate some details about the functions we use in the program.

- PreProcess:
  The pre-process function is employed on the image read by main function.
  Pre-process is composed of scaling, color convert, edge detection, contours detection and approximation. Under this function we scale the image to a good extent then colour conversion is applied on image to convert it to gray scale image.

- Reshape The reshape() function on any array object can be used to reshape the data [9]. In the case of reshaping a one-dimensional array into a two-dimensional array with one column, the tuple would be the shape of the array as the first dimension (data.shape[0]) and 1 for the second dimension.

B. Webcam

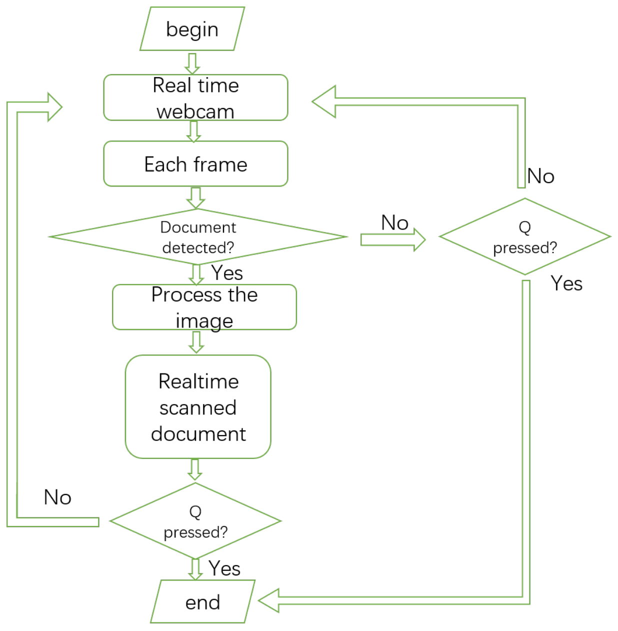

Fig. 3. The webcam

In this section we perform the real-time webcam with the use of cap = cv2.VideoCapture(0) [5]. As shown in Figure4We process the video frame by frame.For each frame through preprocess we know whether the document is detected or not. If we detected the document then we use what we've written in main file to process the image. After that we display the real-time scanned document on the screen. By pressing the button Q, we can exit the webcam.

IV. Results

In this section, the results will be given. We'll analyze one middle process result of edge detection and 3 different figures with typical features.

A. Edge detected

We can see that the edges are clearly detected. When changing the canny threshold it shows slightly difference between the left and right figures but as we use the contour that takes the largest square, it won't influence the results much.

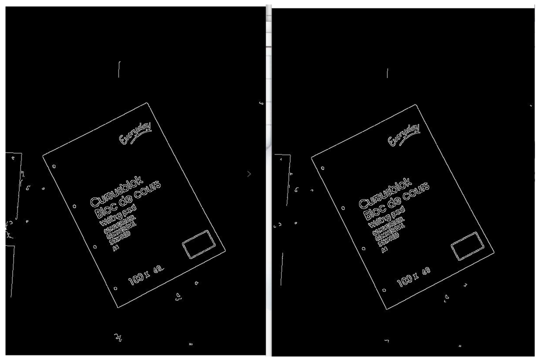

Fig. 4. The edge detected

B. White and Black document

From fig.5 we can see that we have a good result for the black and white document with a high density of characters. The threshold and enhancement has been applied.

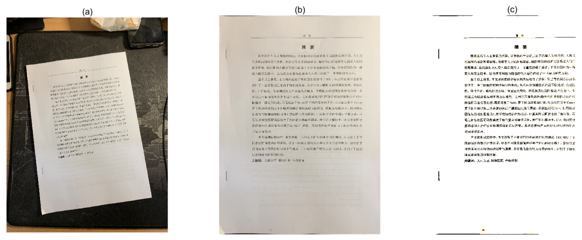

Fig. 5. **Original Image of a black and white file (a), its perspective corrected version (b) and after threshold (c)**

C. Color document

From fig.6 we can see that after enhancement the result for color document is very good. It really looks like a scanned version.

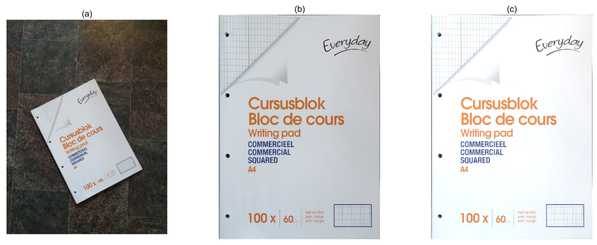

Fig. 6. **Original Image of a colored file (a), its perspective corrected version (b) and after (c)enhancement**

D. *Color document with a different background*

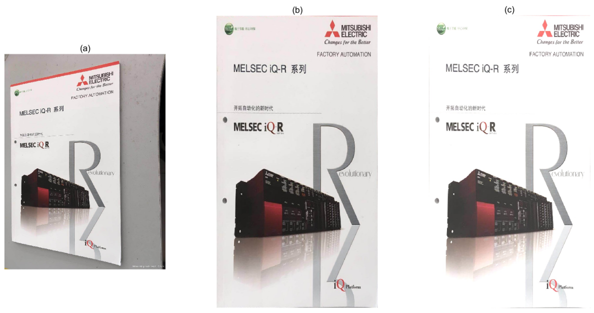

Fig. 7. **Original Image of a colored file (a), its perspective corrected version (b) and after (c)enhancement**

E. *Further discussion*

As all basic requirements are met, we take a further discussion with a corner case of our implementation and conclude the weakness of our work.

- **Case without noise filter**
  As we know, applying a Gaussian filter is required if Canny edge detection is employed. However, filtering the image before edge detection also causes the information lost to some extent. Redoing the color document case without Gaussian filter is an interesting way to implement. The results are shown in the Figure8 As we can

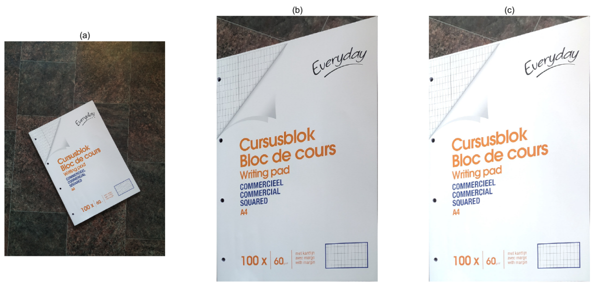

Fig. 8. **Original Image of a colored file (a), its perspective corrected version (b) and after (c)enhancement**

  see, edge detection error occurs in this case due to the Canny algorithm takes a wrong point of background (a floor angle) as a document angle. We can have the conclusion that a Gaussian filter is necessary if we desire a correct edge detection. As for the information integrity and detection accuracy, it is more likely to be regarded as a trade off, as long as we choose a suitable kernel size and sigma, a correct document edge detection while keeping main document information is always obtainable.
- **weakness**
  During the tests of the project, sometimes the edges cannot be detected correctly. After analyzing situations that edges were failed to be detected, we get following conclusions. First, when the picture perspective is serious, we may not be able to transform documents into real aspect ratio. Second, pure background will lead to a good edge detection result. If the background is too complex, Canny detector will find extra wrong edges and will cause error in finding contours. Second, lighting conditions also influence the result. When there exist shadows cause by hand or other stuffs, the contrast of shadow area and background will reduce, which will also lead to edge detect error. Third, for different images and under different circumstances, the best alpha and beta to enhance image to a good brightness and white balance is different. However, in practice we could only set them as a fixed value. Overall, we should choose pure background and avoid bad light conditions and severe perspective when using this document scanner in order to get a better result.

V. Conclusion

In this project, first we study the relative algorithm and the usage of powerful python tools. we design a full image processing flow on the file edge detection and rectification. During the processing, multiple algorithms are applied and some functions are designed. At last we realize the pipeline real-time which could be performed lively. Basically, we achieved all the requirements and the bonus and test our code in different environments. Our work shows the desired robustness with good results.

References

[1] Gaussian Blurring with Python and OpenCV. [Online]. Available: https://medium.com/@florestony5454/gaussian-blurring-with-python-and-opencv-ba8429eb879b [Accessed: 22-Mar-2019]
[2] Canny edge detector, Wikipedia, 06-Jun-2019. [Online]. Available: https://en.wikipedia.org/wiki/Canny_edge_detector. [Accessed: 09-Jun-2019].
[3] Contour Approximation Method, OpenCV. [Online]. Available: https://docs.opencv.org/3.1.0/d4/d73/tutorial_py_contours_begin.html. [Accessed: 09-Jun-2019].
[4] A. Ghuneim, Contour Tracing. [Online]. Available: http://www.imageprocessingplace.com/downloadsV3/rootdownloads/tutorials/contourtracingAbeerGeorgeGhuneim/alg.html. [Accessed: 09-Jun-2019].
[5] Python Programming Tutorials. [Online].Available: https://pythonprogramming.net/loading-video-python-opencv-tutorial/. [Accessed: 09-Jun-2019].
[6] A. Flores and A. Flores, Gaussian Blurring with Python and OpenCV, Medium, 22-Mar-2019. [Online]. Available: https://medium.com/@florestony5454/gaussian-blurring-with-python-and-opencv-ba8429eb879b. [Accessed: 09-Jun-2019].
[7] R. David, Chapter 5: Line, Edge and Contours Detection, Robin David. [Online]. Available: http://www.robindavid.fr/opencv-tutorial/chapter5-line-edge-and-contours-detection.html. [Accessed: 09-Jun-2019].
[8] Structural Analysis and Shape Descriptors, OpenCV. [Online]. Available: https://docs.opencv.org/3.1.0/d3/dc0/group_imgproc_shape.html[Accessed: 09-Jun-2019].
[9] J. Brownlee, ”How to Index, Slice and Reshape NumPy Arrays for Machine Learning in Python”, Machine Learning Mastery, 2019. [Online]. Available: https://machinelearningmastery.com/index-slice-reshape-numpy-arrays-machine-learning-python/. [Accessed: 09-Jun- 2019].

Appendix

A. Teamwork

- Xiaojian Zheng: For this project, I learnt the Python programming and algorithms behind the project. I also engaged the work on applying the algorithm, understanding the code and anticipating the design of the analysis. Finally, I organised the report and wrote the report.

- Boya Zhang: During this project, I learnt how to code with python and write the code together with Xiaotian. I managed to understand the principles behind the existing modules. I also worked on many parts of the report.

- Xiaotian Liu: I studied edge detect and perspective transform algorithms and then built main fuctions of preprocess, transform and enhancement. I also achieved webcam scan and realtime scan functions. Besides, I analyzed weaknesses of code sand get a conclusion how to use our code better.

- Divyanshi Awasthi: During the completion of this project I learnt Python programming and got acquainted with libraries and algorithm used in the project. I unterstood the project workflow and thoroughly reviewed the code. Finally I worked on the report to explain the workflow/ functions and algorithms being used in the code.