HOCHSCHULE
RAVENSBURG-WEINGARTEN
UNIVERSITY
OF APPLIED SCIENCES

Document Scanner

Leopold Gaube and Florian Betz

May 7, 2022

Contents

1 Requirements

There are already many document scanners on the market that can digitalize a paper or similar. The quality of the document scanning depends on the used algorithms. To garantues a good scan, a perspective transformation should be used.

A comfortable library to use computer vision functions is Open Source Computer Vision Library (OpenCV). OpenCV is an open source computer vision and machine learning software library. [Ope22] You can use OpenCV with the programming languages C, C++, Python or Java.

2 Algorithm Overview

Our algorithm for scanning a document in an image consists of three main parts, each with its own challenges.

First, we will have to localize the document in the image. Then we will use a perspective transform from the corner points of the document to the entire span of the image in order to obtain a top-down view without any distortions. Finally, we will use a binary filter to distinguish text from background and save the resulting image as our scanned document.

3 Challenges

It is fairly straight forward if we work with an image like in Figure 1a. However, we also want our algorithm to work under bad lighting conditions [Fig. 1b], taken from an extreme angle [Fig. 1c], with a document on a low contrast background [Fig. 1d]. Of course the final image will suffer in quality compared to a document taken under near-optimal conditions, but our goal is to still obtain a readable document.

4 Document Localization

Some smartphone scanner apps require the user to mark the four corner points of the document manually. However, we want to automate this process by detecting the corner points using Computer Vision. Detecting the corner points accurately and consistantly is probably hardest and most crutial step of the entire algorithm.

4.1 Preprocessing

Before we do Edge Detection, it is advisable to smooth the image in order to get rid of noise. Otherwise this noise may result in detection of edges which may negatively influence the algorithms performance. –¿ Gaussian Blur

4.2 Edge Detection

–¿ Canny Edge Detection (Sobel instead?)

4.3 Contours

ToDo: Explain contours

We can assume that in our edge image there is a contour that matches the outline of our document. It is ok, if we miss the exact location of the document corners by a couple of pixels, but detecting the wrong contour will make the resulting document unreadable.

Detecting contours is easy with OpenCV using the cv2.findContours(...) method. However, we have to distinguish the one corresponding to our document from all the other contours in the image. Other contours may include text, graphs and pictures on the document itself or even objects e.g. a stapler on the desk. The contour we are looking for is one of the largest by area and we should be able to approximate it with only four points. Both of these assumptions will help us to find the right one, as it is highly unprobable another contour meets both criteria. We will just pick the largest contour that can be approximated using four points as show by the following pseudo code:

5 Perspective Transformation

top-down view

6 Thresholding

6.1 Simple Thresholding

For the last step of our project we convert the scanned document into a binary image, thus each pixel is supposed to be either black for text and graphs or white for the background. The easiest way to achieve this is by converting the color image into

grayscale and setting a threshold value. Any pixel value below this threshold would become black, pixel values above would be set to white.

6.2 Adaptive Thresholding

However, there is a problem with this simple thresholding approach, as it performs poorly under bad lighting conditions as can be seen in [Fig. 3a]. When taking pictures of documents, a user may cast a shadow with their camera or mobile phone onto the document. This can be problematic if a region in the shadows has lower pixel values for the background than the text in a well-lit region. In such a scenario, it would be impossible to find a static threshold that works well for the entire document. So instead we need an adaptive approach that looks at a set of neighboring pixels and chooses a threshold for this local region dynamically.

List of Acronyms

OpenCV Open Source Computer Vision Library

References

[Ope22] OpenCV. Opencv - about. `https://opencv.org/about/`, 21.04.2022.