Computer Vision SS22 Assignment 1: Document Scanner

Felix Hamburger
Student ID: 35925
Computer Vision SS22
Computer Science Master
Ravensburg Weingarten University
Email: felix.hamburger@rwu.de

Mario Amann
Student ID: 35926
Computer Vision SS22
Computer Science Master
Ravensburg Weingarten University
Email: mario.amann@rwu.de

This paper presents a usecase for a DIN A4 document scanner. The explanations contained in this paper will be divided into three steps. First the corner points of the document will be detected utilizing the canny edge corner detection. By performing a perspective transofmation the document will then be provided in a top-view. With the help of a filter the document will then be changed into a binary format.

1 INTRODUCTION

In everyday life, there are often situations in which it is advantageous to scan a document that has been received and keep it as a copy. Unfortunately not everyone has a ready-to-use scanner with them to scan the document. Though most of the people have a device capable of doing just this - a cell phone with a camera. Using various methods, it is possible to scan documents with the help of the cell phone camera, similar to a scanner, and save them in a suitable foramt. This paper will give your step by step guide on how it is possible to implement these methodologies.

2 PREPROCESSING

The first step is to read the image. This is done with the imread function of openCV which outputs the image as numpy array with the shape (Height, Width, Channel). By default the decoded image is stored in the BGR-format. Which means, the first channel contains the blue color channel, the second one contains the green color channel and the last the red one. The original image is seperately stored, because for further steps the photo is needed in a grayscale format. This convertion is made with the cvtColor[Docd] function of openCV. The convertion of colorspace is made with the following formula[Docb]:

$$Y = B * 0.114 + G * 0.587 + R * 0.299$$

The next preprocessing step includes the smoothing/blurring of the grayscale image. This is done to remove noise from the image[Doci]. In our case the Gaussian Blurring[Docf] is used. The image is convolved with the Gaussian Kernel. As size of the kernel the size (5,5) is used. Because a document could be landscape or portrait format, it makes sense to use same values in x and y direction in the kernel size, as well as for the parameters sigmaX and sigmaY. These parameters are used to calculate the gaussian filter coefficients. In the function call sigmaX and sigmaY are set to zero, which means the Gaussian Kernel computes these variables from the given ksize[Docg].

$$\sigma = 0.3 * (0.5 * (ksize - 1) - 1) + 0.8$$

The gaussian filter coefficients are computated with the following formula[Docg]:

$$G_i = \alpha * e^{-\frac{(i - \frac{(ksize-1)}{2})^2}{2*\sigma^2}}$$

3 EDGE DETECTION

After we took our preprocessing step and chose a suitable noise reduction algorithm we start with the edge detection. For edge detection we use the Canny edge[Can86] detector which is a multi-staged algorithm:

1. Noise reduction.
2. Calculating the intensity gradient of the image.

3. Non-maximum supression.
4. Hysteresis thresholding.

The noise reduction step was allready described in the Preprocessing section of this paper.

The intensity gradient of the image is calculated using the **Sobel Filter**:

$$S(I(x,y)) := \sqrt{(S_x * I(x,y))^2 + S_y(*I(x,y))^2} \quad (1)$$

Generally we can define the gradient of the Image I as:

$$G(I(x,y)) := \sqrt{I_x^2 + I_y^2} \quad (2)$$

In addition to the gradient we calculate the the orientation given by:

$$\phi(x,y) = arctan(\frac{g_y}{g_x}) \quad (3)$$

After getting the gradient magnitude and orientation of each pixel, the edges have to be reduced to a thickness of one pixel which will be done with **non-maximum supression.**

To implement this methodologie in the document scanner, the function which implements the full Canny edge detector[Doca] can be called.

$$Canny(blurred\_image, 75, 180,$$
$$L2gradient = True, apertureSize = 3)$$

The functions input is the noise reduced image described in Preprocessing. For Hysteresis Threshold 75 for the lower-bound and 180 for the upper bound are chosen. In several trials this settings could sufficiently provide the overall best results, when setting the aperture size to 3. Instead of the predefined L1-Norm [Weia], the more precise L2-Norm[Weib] is used.

The canny edge detection filter returns an binary output image with edges beeing set to 1 and other points being set to 0.

4 CORNER DETCTION

To find the corner of the document the previously generated edges in Edge Detection have to be evaluated. For this purpose, contours are formed which are represented by a curve that connects all continuous points with each other.[Sb85]

The OpenCV function[Doce] is used as follows:

$$contours, hierarchy = findContours(edges,$$
$$RETR\_LIST, CHAIN\_APPROX\_SIMPLE)$$

Where the input are the previously generated edges, the retrieval mode[Doch] which is set to retrieve the contours without establishing any hierarchical relationships and the contour approximation mode[Docc] which is set to ouput only their endopints. (E.g: A rectengular contour is defined by the 4 corner points)

5 TRANSFORMATION

6 BINARY IMAGE

7 SUMMARY

REFERENCES

[Can86] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[Doca] OpenCV Documentary. Canny. https://docs.opencv.org/4.x/dd/d1a/group__imgproc__feature.html#ga04723e007ed888ddf11d9ba04e2232de. Accessed: 2022-05-09.

[Docb] OpenCV Documentary. Color convertion rgb to gray. https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html#color_convert_rgb_gray. Accessed: 2022-05-09.

[Docc] OpenCV Documentary. Contourapproximationmodes. https://docs.opencv.org/4.x/d3/dc0/group__imgproc__shape.html#ga4303f45752694956374734a03c54d5ff. Accessed: 2022-05-12.

[Docd] OpenCV Documentary. cvtcolor. https://docs.opencv.org/3.4/d8/d01/group__imgproc__color__conversions.html#ga397ae87e1288a81d2363b61574eb8cab. Accessed: 2022-05-09.

[Doce] OpenCV Documentary. Findcontours. https://docs.opencv.org/4.x/d3/dc0/group__imgproc__shape.html#gadf1ad6a0b82947fa1fe3c3d497f260e0. Accessed: 2022-05-12.

[Docf] OpenCV Documentary. Gaussianblur. https://docs.opencv.org/4.x/d4/d86/

group__imgproc__filter.html#gaabe8c836e97159a9193fb0b11ac52cf1. Accessed: 2022-05-09.

[Docg] OpenCV Documentary. getgaussiankernel. https://docs.opencv.org/4.x/d4/d86/group__imgproc__filter.html#gac05a120c1ae92a6060dd0db190a61afa. Accessed: 2022-05-09.

[Doch] OpenCV Documentary. Retrievalmode. https://docs.opencv.org/4.x/d3/dc0/group__imgproc__shape.html#ga819779b9857cc2f8601e6526a3a5bc71. Accessed: 2022-05-12.

[Doci] OpenCV Documentary. Smoothing images. https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html. Accessed: 2022-05-09.

[Sb85] Satoshi Suzuki and KeiichiA be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[Weia] Eric W. Weisstein. L1-norm. https://mathworld.wolfram.com/L1-Norm.html. Accessed: 2022-05-09.

[Weib] Eric W. Weisstein. L2-norm. https://mathworld.wolfram.com/L2-Norm.html. Accessed: 2022-05-09.