Document Scanner

Filip Konopacki

GitHub: https://github.com/filipkonopacki

Table of contents

1. Vision

Documents and notes pictures are not easy to manage and maintain. When you have many notes pictures sometimes it is hard to find information you need, or photo has been taken from an angle and it is not comfortable to read. Also, rewriting notes from pictures is pointless and time-consuming. For those who have these problems, the Document Scanner is application which allows you to crop the document or note from the picture and after analyzing text save it to digital form.

2. Current State

For now, application provides user option to crop document or note from the picture automatically or by selecting document corners manually. Next step is to detect text areas in the cropped picture for further analyze.

3. Implementation and requirements

Code is written in C++ using OpenCV library and Visual Studio 2017. Source code is provided in the package with documentation in ManualPerspectiveCorrector and AutomaticPerspectiveCorrector directory. To run the code the OpenCV library must be installed and configured. Library can be found at https://opencv.org/ . After configuring OpenCV, property sheets included in package must be added to project. In package can also be found Documents directory with documents pictures examples and example usage code.

4. Working principal

First, document image must be loaded. Image could have any resolution but must has distinctive background. Next, in both versions, image size is normalized and then, in manual version, image is shown to get document corners from user by double-clicking them. In automatic version image is preprocessed. First, image is converted to grayscale and then fixed-level thresholding is applied. Thresholding is typically used to get a bi-level (binary) image out of a grayscale image or for removing a noise, that is, filtering out pixels with too small or too large values. In this case Otsu algorithm for thresholding is used. Otsu's thresholding method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels each side of the threshold, i.e. the pixels that either fall in foreground or background. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum. After thresholding, binary image can be used for finding its contours. For that, OpenCV function called findContours is used. This function returns vector of vectors containing contours points. Next step is to find the largest contour which should be the document. Typically, documents do not take more than 80% of the picture so this value is used in choosing contour. Contour has many points but to correct perspective four corners are needed. To get them, first, OpenCV function convexHull is used on set of contour points. This function finds the convex hull of a 2D point set using the Sklansky's algorithm. Then the function approxPolyDP approximates a curve or a polygon with another curve/polygon with less vertices so that the distance between them is less or equal to the specified precision. It uses the Douglas-Peucker algorithm. After several iterations of this function with incrementing precision, it returns four corners of rectangle bounding the document. Next, algorithm checks in which order points were detected to establish points in destination image. Using both, current and destination points, homography is calculated. A homography is a transformation (a 3×3 matrix) that maps the points in one image to the corresponding points in the other image. Finally, with calculated homography and OpenCV function warpPerspective, corrected image is obtained.