Documentation

# Milestones

1. ~~Basic IO – loading images into workable formats, displaying them with additional information such as corners and edges detected and storing them.~~
2. Corner detection – ~~Detect corners~~, their orientation and size.
3. Filter out the corner quadruplet that roughly represents the outline of the page (excluding skewing and bends).
4. Detect edges – Detect edges, their direction and intensity.
5. Filter out edges that pass through the four corners and represent the outline of the page
6. Calculating transformation matrices and operations necessary.
7. Perform Transformation to de-skew, crop image.
8. Apply Otsu’s method to turn the document black and white using an appropriately selected threshold.

# Log

21/10/2018

I have started the program with the file IO, image files are specified through command line arguments, and they are then opened and processed one by one.

23/10/2018

I have begun the corner detection, however due to copyright issues the SIFT library is not available. Consequently I have tried to use ORB instead. I’m having troubles to figure out the size and directions of corners.

03/10/2018

I cannot figure out how to get the size and orientation of corners using the libraries available in opencv and decided to change the corner detection to a feature matching algorithm where instead of finding size and direction I will provide a sample page corner from which a descriptor is computed, then the four corners closest to this are selected.

# Introduction

The goal is to write a python program that accepts images of documents (i.e. images taken with a camera), detects the main document in the image, its corners and edges to then de-wrap and transform the image to flatten the document and crop it using corner and edge data. Lastly It should convert the resulting image into a black and white image using Otsu’s method.

Previously existing page de-wrapping algorithms work by detecting lines of text and calculating the transformations required to straighten these lines. However, this requires horizontal text and is less suitable for documents containing a significant portion of vertically oriented text and graphics. Thus, this project explores a combination of edge and corner detection to solve this problem. The program aims to identify the page as a whole and successfully de-wrap the image regardless of content.

This approach may be faster as it simply finds four corners and then transforms the image, however it may not work well pages that are bent.

# Problems

Opencv corner detection and feature detection libraries do not provide corner size and orientation information, thus instead of finding size and direction I will provide a sample page corner from which a descriptor is computed, then the four corners closest to this are selected.

# Achievements

# Analysis

# Conclusion