Wyatt McCurdy

COS 570

Computer Vision Assignment 2

# Task 1: Object Size Measurement

My book measurement processing script processes an image to detect and measure the dimensions of a book within it. The image is first loaded and converted to grayscale, followed by a Gaussian blur to reduce noise. Edge detection is then performed using the Canny edge detector, which highlights the edges in the image. The contours of these edges are found using OpenCV's [findContours](vscode-file://vscode-app/c:/Users/12078/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) function. One contour highlights the edge of the paper, while the other contour highlights the edge of the book. The edge detection works very well for this image due to the large gradient between pixel values at the edge of both the paper and the book.

The contours are approximated to polygons using the [approxPolyDP](vscode-file://vscode-app/c:/Users/12078/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) function, which simplifies the contours to a set of vertices. These polygons are then drawn on the original image for visualization. The script calculates the real-world dimensions of the inner contour (the book) based on the known dimensions of the outer contour (the paper, assumed to be 21.5 cm by 27.8 cm). The Euclidean distance between the vertices of the polygons is used to determine the width and height in pixels, which are then converted to centimeters. The calculated dimensions are annotated on the image, and the annotated image is displayed using OpenCV's [imshow](vscode-file://vscode-app/c:/Users/12078/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) function.

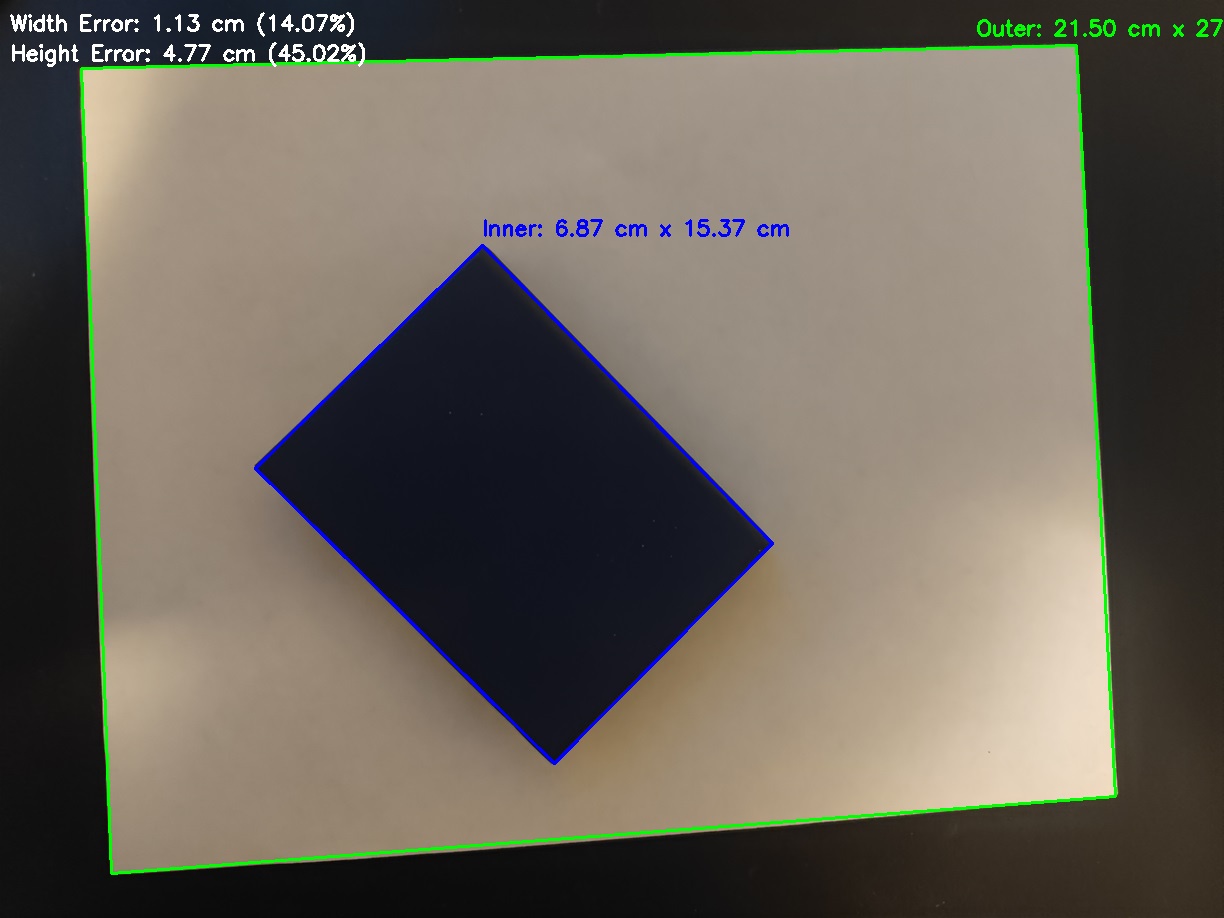


Figure 1: Measurement estimates and error for book laying on A4 paper.

# Task 2: Text Recognition

## Text Analysis

The text analysis script is designed to recognize text in a JPG image using OpenCV and PyTesseract. First, I read in the target image and convert to grayscale. Then, I apply thresholding to the grayscale image to create a binary image, which enhances the text regions for better recognition. I use PyTesseract to extract text from the thresholded image, and print it to the console. Using PyTesseract, it is a relatively simple process, relying on PyTesseract’s image\_to\_string method to do most of the heavy lifting.

A yellow rectangular object with black text

Description automatically generated

ifgure 2: Sample text original image for first text recognition task.

A black screen with white text

Description automatically generated

figure 3: original text picture for text recognition task, thresholded to highlight text.

A screenshot of a computer program

Description automatically generated

Figure 4: output for the first text recognition task.

## Sign Recognition