Project Report for Digital Image Processing (Phase A)

Maria Eleni Chochlidaki
Michail Samaritakis

Project's Goal:

The purpose of this project is to detect the quality of stitches and incisions from a variety of images produced during a surgical operation. This can be accomplished by using computer vision and digital image processing techniques.

Project's Plan:

Functions that have been implemented but not working properly yet:

- Count the stitches in an image based on annotations

- Check if an image is bad data using the functions:
    - is_image_blurred()
    - is_image_brightness_ok()

TODO:

- Count properly the stitches in an image based on annotations

- Train our code to detect stitches from non-annotated images

- Count detected stitches
    - We need to create a real classification model. In the context of machine learning, a classification model is used to categorise data into a certain number of predefined classes.

- Check if an image is bad data

Project's Progress:

So far, our program has the capabilities mentioned below:

- Parses the annotations from the .xml file

- Preproccesses image
    - The *preprocess_image* function takes an image as input and performs two operations on it: grayscale conversion and Gaussian blurring.

    - Grayscale conversion is done using the *cv2.cvtColor* function with *cv2.COLOR_BGR2GRAY* as the colour conversion code. This converts the input image from the RGB colour space (Red, Green, Blue - the default colour space for images in OpenCV) to grayscale. Grayscale images are easier to process than colour images because they have only one channel of pixel intensity information, compared to three channels (R, G, B,) for colour images.
    - After converting to grayscale, the function applies a Gaussian blur to the image using the *cv2.GaussianBlur* function. The parameters (5, 5) specify the width and height of the Gaussian kernel (the window of pixels used for the blur operation), and 0 is the standard deviation of the Gaussian distribution used for the blur. Gaussian blurring is a common image processing technique used to reduce image noise and detail.

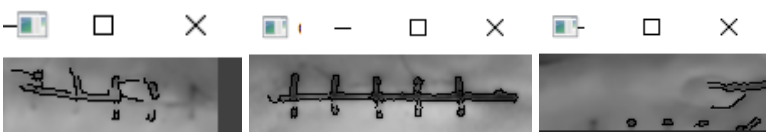
- Segments image

    - The *segment_image* function takes a grayscale image as input and performs two operations on it: edge detection and contour finding.

    - Edge detection is done using the *cv2.Canny* function with 50 and 150 as the lower and upper threshold values for the hysteresis procedure, which is a part of the Canny algorithm. The Canny algorithm is a popular edge detection method because it is robust to noise, provides clear edge lines, and is able to preserve edge location accurately.

    - After detecting edges, the function finds contours in the image using the *cv2.findContours* function with *cv2.RETR_LIST* as the contour retrieval mode and *cv2.CHAIN_APPROX_SIMPLE* as the contour approximation method. Contours are simply the boundaries of the objects present in the image. The cv2.findContours function returns a list of all contours found in the image. The _ in the function call is a common Python idiom for discarding values that are not needed (in this case, the hierarchy of contours, which is not used in this function).

- Visualises contours like so:



- Parses arguments

- Writes the results to a .csv file

How to run the Project:

In order to run the code in Visual Studio Code you can run the script by typing: *python src/run.py output.csv -v image1_filename.jpg image2_filename.jpg image3_filename.jpg* for example, into the terminal and pressing Enter.