**Project Report**

**Project overview – Recognizing Hebrew Text**

This project aims to detect Hebrew Characters in images and recognize them. We implemented the project with python and OpenCV library.

Phase 1- Detect the contours of the words in the image:

In the first part of the project we used the “East model” in order to detect words. In order to use the model, we had to construct a blob from the image and then forward it in the different layers of the model to obtain the two output sets. The output geometry map used to derive the bounding box coordinates of text in our input images. And similarly, the scores map, containing the probability of a given region containing text.

Phase 2 - Rectify and recognize the detected boxes:

In the second phase we used the pytesseract library. The input for the phase is the output from phase one which is the boxes. For each box which represent a word in the image we recognized the word using pytesseract function “image\_to\_string”. The function gets as input the boxes and config (we config the text to Hebrew) and print to the console the string that represent the word in the image. In order to use the function, we had to download the “tessdata” which is a pre trained model for Hebrew recognition.

**The progress in the project:**

First, we searched on the internet for articles on text detection and recognition. In these articles we have been exposed to the “East model” as the best option to implements the first phase of our project. After implementation of the first phase, we tried it on a collection of pictures and as a result we improved the project by adding padding to each box. In this part we search for a tool that can recognize the words. After consulting Jihad, we choose to use Tesseract as the optimal option.

**How to run the project:**

* Install the proper environment (Python, OpenCV, Tesseract, … ,etc.).
* Download the EAST model from [here](https://github.com.cnpmjs.org/dilhelh/opencv-text-detection/blob/master/frozen_east_text_detection.pb).
* Download the dataset for the tesseract from [here](https://github.com/tesseract-ocr/tessdata/blob/master/heb.traineddata).
* Download the image to images folder that included in the project.
* Run “python ./ Hebrew\_Recogntion/ Recogntion\_hebrew\_in\_wild.py [path to your image]”

E**xamples of the project:**

-The green color represents the output from phase one which is the boxes.

-The red represents the box after padding and also the output of phase two the text itself.

A picture containing text, sign, scoreboard

Description automatically generatedFigure 1:

A picture containing text, sign, screenshot

Description automatically generated



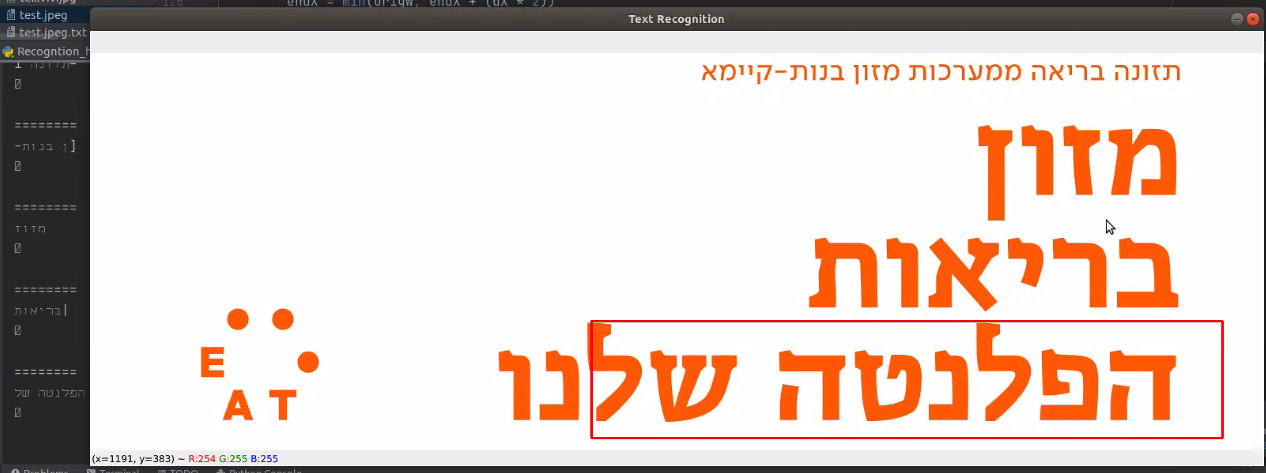
Graphical user interface, application

Description automatically generated Figure 2:

Figure 3:

Graphical user interface, text, application

Description automatically generated

 Figure 4:

Example 5:

Figure 5:

Graphical user interface, application

Description automatically generated

**Result analysist of the examples:**

1. The first figure is an example for a successful boxing process but because the sign in the figure is the inclined, we had to add some padding for the recognition process which worked successfully.
2. In figure two, the boxing process was successful because it detects all the words in the picture. We had to add some padding for the recognition part and as displayed above the tesseract didn’t recognize correctly the first word as we expected but, the rest of the words were recognized and identified correctly. We assume the reason for the bad recognition is due to the font of the word in the figure.
3. In figure three the boxing process failed to detect the words. We tried to add some padding in order to increase the boundaries of the boxing but as displayed in figure four it didn’t help for the recognition part to identify the words. In addition, we tried to use an anther tool of tesseract by changing the configuration from recognize a word to automatic text recognition and the result are displayed in figure 5.

In general, we tried to understand the why the boxing process was unsuccessful which meaning that the boxing didn’t bound the whole word. Our assumption is that the EAST model was trained on a dataset that contains only pictures with English text. In English there is threes the line center unlike in Hebrew which have two line. This may lead to a false boxing process and for a bad result.

In addition, we tried to understand the reasons for the failing of the tesseract tool. We tried to examine the pixels in magnifying conditions and the result were that there is some “background noise” in the picture meaning that some colors look the same to the naked eye, but the value of those pixels were different. Another reason for failed recognition can caused by the font of the text, specific text color against his background color and the more the figure is similar to the dataset, that the model was trained on, the better results.