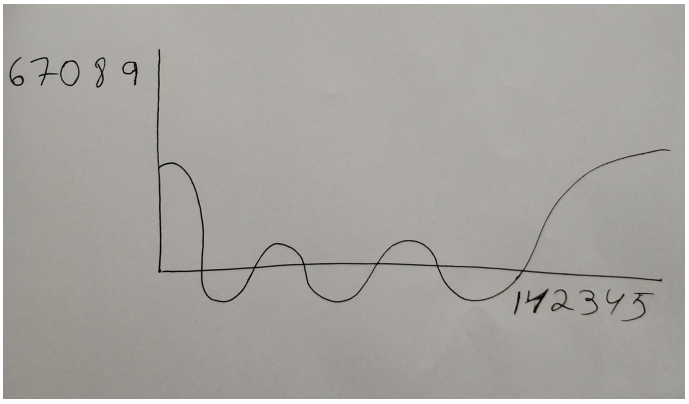
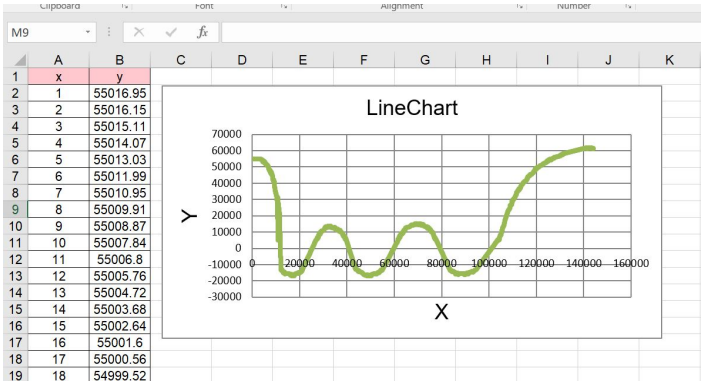
GTT

Graph To Table
Developer Manual



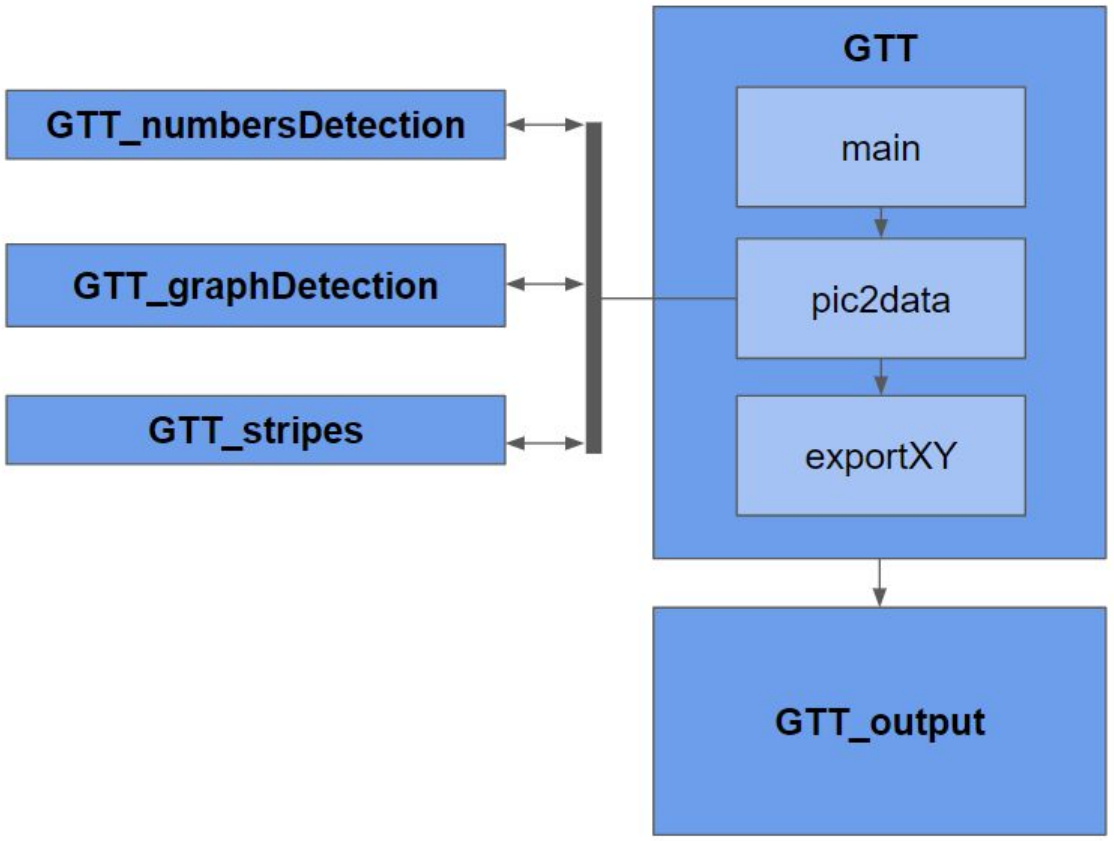


Mark Rachman and Nir Kravetzky

Table Of Contents

1. Schema

The design of Graph-To-Table can be modeled in the following form:



Note that each dark blue box represents a single module - the module name is written in bold. Smaller boxes inside GTT main module represents the important functions in our process. Interactions between the modules and functions are represented by the arrows - the data flow direction is represented by the arrow direction.
As can be seen from the model - GTT_numberDetection, GTT_graphDetection and GTT_stripes functionalities are managed and integrated by ‘pic2data’ function.

2. Modules

2.1 GTT

- Dependencies: all of GTT modules, Numpy.
- Purpose: Main file and integration point for the GTT program.
- API:
  - **stopRun(out = None)**
    Final stop - exits the program.

    Parameters:
    - out - string to be printed to the console. Contains the error message.

    Return values: None.

    Process:
    - prints the ‘out’ string to the console. If out is None prints default "Error occurred” message.

    Assumptions:
    - “out” parameter is a string.

  - **pic2data(source,logLevel,x_y_vals = None)**
    The main function - receives the source image and connects between visual recognition of graph line and numbers.

    Parameters:
    - source - the path to the source image.
    - logLevel - console line parameter, if given defines the level of the log.
    - x_y_vals - console line parameter, if given - those numbers will be the numbers inserted to the process instead of the number recognized from GTT_numbersDetection.

    Return values:
    - list of graph points scaled to the wanted x-y values.

    Process:
    - Receives all the points in the graph.
    - Sends them to GTT_stripe in order to get the graph axes and origin point.
    - Sends the source image and axes to GTT_numbersDetection.
    - Gets the graph line from GTT_graphDetection.
    - Integrates between the results from GTT_numbersDetetion and GTT_graphDetection.

- Creates a list of all relevant graph points, converted to (x,y) format and returns them.

Assumptions:

- "source" parameter is a valid path to image file.

- **exportXY(dot_list,x_scale)**

  Receives a dot list and formats it to the wanted out template.

  Parameters:

  - dot_list - the dot list to be formatted.
  - x_scale - the wanted scale for the x axis.

  Return values:

  - Formatted output - ready to be written to excel/csv file.

  Process:

  - Goes over all the dots in dot_list - verify that for every x point there is a matching y point. If not generates it according to the assumption that between every two dots in the dot_list runs a straight line.

  Assumptions:

  - Between every two dots in the dot_list runs a straight line.

2.2 GTT_numbersDetection

- Dependencies: GTT_utils, OpenCV, PIL, Keras, Numpy, Tensorflow
- Purpose: Recognize the handwritten digits in the given image.
- API:
  - **recognize(imagePath, x_axis = None, y_axis = None, logLevel = 0)**

    The image recognize all digits inside the image.

    Parameters:
    - imagePath - the path to the image.
    - X_axis - the recognized X axis in the image - if it is not None digits will be recognized only below the X axis.
    - Y_axis - the recognized Y axis in the image - if it is not None digits will be recognized only from the left side of the Y axis.
    - logLevel - console line parameter, if given defines the level of the log.

    Return values:
    - recognized_digits_list - first element contains the number near X axis, and the second element contains the number near Y axis. In case of error: [-1, -1]
    - rect_list - returns all the rectangles that contains digits.
    - tempFolderName - return the folder name that contains log files. In case of disabled log level - None.

    Process:
    - Convert image to gray colors.
    - Search for rectangles in the images that may contain potential digits.
    - In case of X_axis / Y_axis the rectangles will be placed below/in left of the axis.
    - Filter rectangles:
      - By size (minimum and maximum threshold for each rectangle size).
      - Overlap rectangles.
    - Run pre-trained model to detect each digit inside of the rectangles.
      - Filter rectangles in case of low detection percentage.
    - Union digits to numbers by their position in the image.

    Assumptions:
    - “imagePath” parameter is a valid path to image file.

2.3 GTT_graphDetection

- Dependencies: OpenCV, Numpy
- Purpose: the module purpose is to recognize the actual line graph
- API:
  - **getGraphRaw(source, rect_list = None)**
    The function returns all the linear lines in the image. Each line length between 0px to 50px. In case of rect_list different then None, the function will filter all the lines that is contained at least in one rectangle.
    Parameters:
    - source - the path to the image
    - rect_list - rectangles list.

    Return values:
    - Lines list - each item contains start and end point.
    - Height - image height.
    - Width - image width.

    Process:
    - Convert image to gray colors.
    - Perform gaussian blur function on the image.
    - Run canny edge algorithm on the image in order to detect curves.
    - Run Hough line algorithm in order to find lines in the image.
    - Filter lines inside rectangles.

    Assumptions:
    - “source” parameter is a valid path to image file.

  - **getGraphPoints(source, x_axis, y_axis, rec_list)**
    Returns all relevant graph line points for the output.

    Parameters:
    - source - image path
    - x_axis - list of points of X axis
    - y_axis - list of points of Y axis
    - rec_list - rectangles list

    Return values:
    - Filtered point list - point list of the graph line
    - X_pixel_max - the x-pixel value of the maximum point in the graph
    - Y_pixel_max - the y-pixel value of the maximum point in the graph

    Process:
    - Run getGraphRaw on the image with rect_list.
    - Filter points by their location with the relevant axis.
    - Calculate the global maximum point in the graph.

    Assumptions:
    - “source” parameter is a valid path to image file.

2.4 GTT_stripes

- Dependencies: GTT_utils, Numpy
- Purpose: Recognize the axes and then use them to deduce the graph origin point.
- API:
  - **getGraphAxesAndOrig(pointList, height, width)**

    The function recognize axes and origin point for the image.

    Parameters:
    - pointList - all the points in the graph.
    - height - the height dimension of the image.
    - width - the width dimension of the image.

    Return values:
    - x_axis - the recognized X axis in the image.
    - y_axis - the recognized Y axis in the image.
    - orig - the intersection between x and y axes.

    Process:
    - Split the image into horizontal and vertical stripes - each stripes is about 3% from the original image.
    - Find the x_axis according to the next logic:
      - The stripe must be horizontal.
      - The stripe with the maximum number of points.
    - Find the y_axis according to the next steps:
      - The stripe must be vertical.
      - The stripe must intersect with the x_axis (notice we already found the x_axis in the previous step).
      - Start scanning the stripe list from the left most stripe in the image to the right most stripe in the image. Note the first intersection point you find - it will be the **origin point**. Keep only stripes that intersect with the x_axis at a similar **origin point** (similar will be considered until a certain pixel threshold).
      - From those stripes choose the one with most points.

    Assumption:
    - pointList, height and width are not None.

2.5 GTT_output

- Dependencies: csv, operator, openpyxl
- Purpose: Handle all output methods for GTT. Generates CSV or Excel file according to user choice.
- API:
  - **createCSV(fileName, dic)**

    The function create a csv file with the given dictionary.
    Parameters:
    - fileName - output file name.
    - dic - dictionary for the table (key + value).

    Return values: No Return value.
    Process: Create the csv file.
    Assumption: None.
  - **createExcel(fileName, dic)**

    The function create excel file with the given dictionary.
    Parameters:
    - fileName - output file name.
    - dic - dictionary for the table (key + value).

    Return values: No Return value.
    Process: Create the excel file with a graph of the given data.
    Assumption: None.

2.6 GTT_utils

- Dependencies: Image.
- Purpose: Contain all utility functions for the program.
- API:
  - **checkIfFileExists(filePath)**
    Check if the given file exists in the file system.
    Parameters:
    - filePath - file path

    Return values: True if the file exists, otherwise returns False.
    Assumption: None.

  - **clacDistnace (x1, y1, x2, y2)**
    Calculate distance between two given points (x1, y1) and (x2, y2).
    Parameters:
    - x1, y1 - represents the first point.
    - x2, y2 - represents the second point.

    Return values: The calculated distance.
    Assumption: None.

  - **clacDistnacePoint (pt1, pt2)**
    Calculate distance between two given points.
    Parameters:
    - pt1 - first point.
    - pt2 - second point.

    Return values: The calculated distance.
    Assumption: None.

  - **createFolder(folderName = ‘’)**
    The function create a new folder in the current directory.
    If the folder name is in use, the function will add number as a suffix.
    folderName default value is ".temp".
    Parameters:
    - folderName - default folder name.

    Return values: Return the new folder name.
    Assumption: None.

  - **deleteFolder(folderPath)**
    The function delete the given folder.
    Parameters:
    - folderPath - path to the given folder.

    Return values: None.
    Assumption: None.

- **getEmptyImageName(folderPath, image = ‘a’, extension = ‘png’, startWithNumber = False)**
  The function returns a valid image name that doesn't exists in the system.
  Parameters:
  - folderPath - path to the given folder
  - image - default image name. In case of none, the image name will be ‘a’.
  - extension - default image extension name.
  - startWithNumber- append number to the image.

  Return values: Full path to the image.
  Assumption: None.

- **convertImageToPNG(source)**
  The function convert the given image to a png format.
  Parameters:
  - Source- the image source.

  Return values:
  - Output - the image output.
  - Create_image - true if the function created a new image.

  Assumption: None.

- **appendToFile(filePath, text)**
  The function append the given text to the file.
  Parameters:
  - filePath - the path to the file.
  - Text - the text to append.

  Return values: None
  Assumption: None

- **deleteFile(filePath)**
  The function deletes the given file.
  Parameters:
  - filePath - the path to the file

  Return values: None.
  Assumption: None.

- **deleteAllFilesByTypeInFolder(filePath, extension)**
  The function delete all files in the given folder with a given extension.
  In case if extension = None, the function will delete all files in the directory.
  Parameters:

- filePath - the path to the file.
- Extension - the required extension files to delete. In case of None, all the files inside the folder will be deleted.

Return values: None

Assumption: None

- **getFileNameAndExtension(img)**
  The function return file name and extension of a given file.
  Parameters:
  - img - path to the image.

  Return values:
  - File_name - file name without extension.
  - file_extension - file extension.

  Assumption: None

- **checkIfPathExists(path)**
  The function check if the given path exists in the file system.
  Parameters:
  - Path - the given path to check.

  Return values: True if the path exists in the system.

  Assumption: None.

3. Digit Recognition

In order to recognize handwritten digits, we trained a classifier in KERAS with TensorFlow. We used the following neural network:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 30, 24, 24)        780
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 12, 12)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 15, 10, 10)        4065
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 15, 5, 5)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 15, 5, 5)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 375)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               48128
_________________________________________________________________
dense_2 (Dense)              (None, 50)                6450
_________________________________________________________________
dense_3 (Dense)              (None, 10)                510
=================================================================
Total params: 59,933
Trainable params: 59,933
Non-trainable params: 0
```

All the classification of the digits is made inside GTT_numbersDetection.py with this trained classifier.

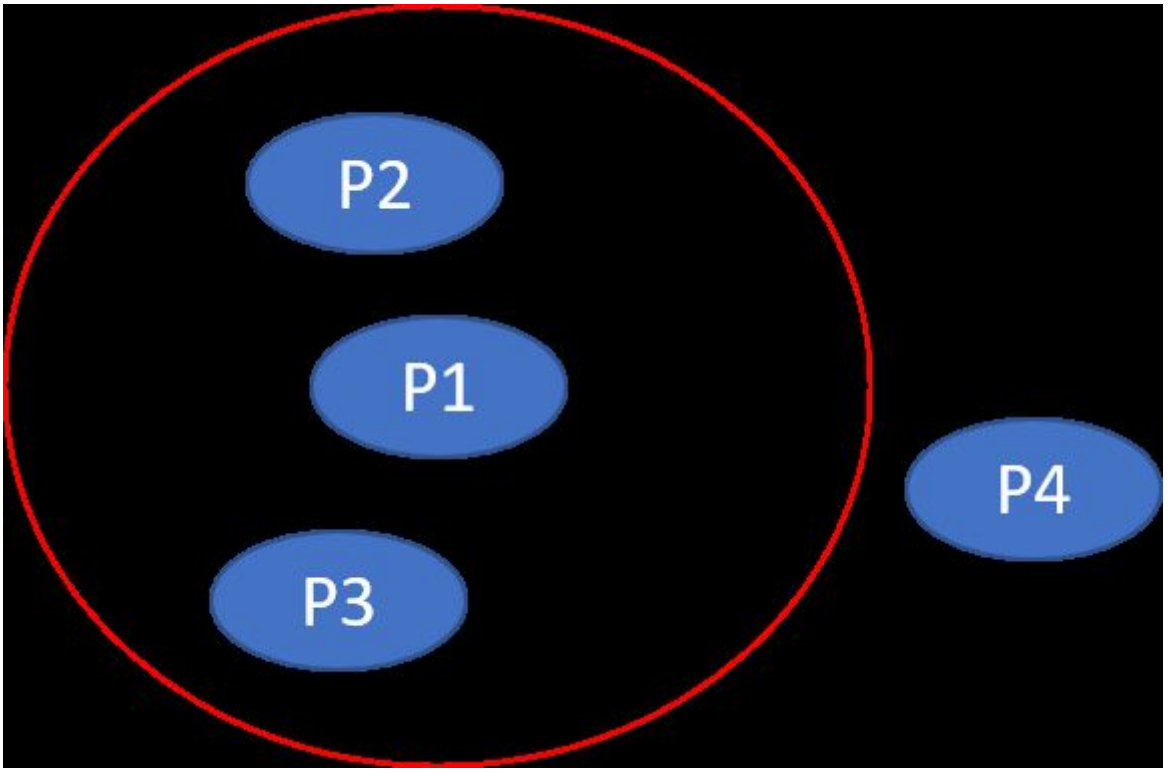
4. Axes Detection

For axes detection we run the following algorithm with only one assumption:

Assumption - The longest vertical linear line is the X axis.

Algorithm:

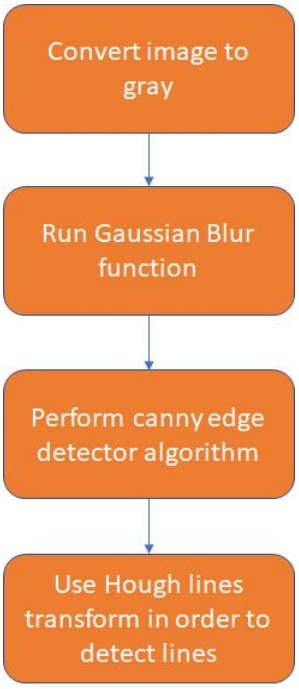
- Find the X-axis (under the mentioned assumption).
- Then, divide the images into equal horizontal stripes. Each horizontal line width is about 3% of the total width of the image.
- For each point in the image, remove all the other points that are located in epsilon area of the given point. For example, in the following case GTT will remove P2 and P3:



- We go over the stripes and find the most-left stripe that intersect with the X axis. We save that point as the “Origin Point”.
- We save all stripes that has a similar “Origin Point” intersection with the X axis (will be considered as similar if the intersection point is inside an epsilon area around the “Origin Point” we found).
- From those stripes we choose the stripe with the most dots - this will be the Y axis.
- The origin point will be the intersection of the X and the Y axis.

5. Object Detection (inside the image)

GTT runs the following algorithms in order to convert an image to a series of points:



All of those algorithms are functions inside the OpenCV Python library.