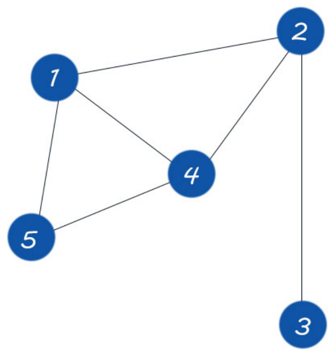
Extracting graph from a ruled sheet of paper

Mudit Rathore, Ishani Mishra, and Gugloth Lavanya

***Abstract*—The digital age has set in, all tasks are performed by machines computers, robots etc. All problems being solved can be seen as graph problems, therefore it is the need of the hour to aid our machines with graph solving skills. Generating and visualizing information as graphs is now-a-days an important step towards solving real life problems. Unlike the human eye, a computer can't differentiate between ruled lines and graph edges. Therefore, it has to be programmed to do so. The objective of this work is to extract a graph printed or hand printed on a ruled sheet of paper. Bidirectional edged, unweighted graphs with no loops have been considered.**

I. Introduction

A graph [1]**G(V,E)** is a data structure which represents a finite set of vertices **V** and set of edges **E** which connect a pair of vertices. A graph **G** can be completely determined by specifying either the **Adjacency matrix** or the **Incidence matrix** which provide an efficient way of representing a graph **G**.



$$A = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 1 & 1 \\ 2 & 1 & 0 & 1 & 1 & 0 \\ 3 & 0 & 1 & 0 & 0 & 0 \\ 4 & 1 & 1 & 0 & 0 & 1 \\ 5 & 1 & 0 & 0 & 1 & 0 \end{array}$$

Fig. 1. Adjacency Matrix representation of a simple, undirected graph

A simple graph is the graph which has no more than one edge between each pair of vertices, no loops, that is no edge can begin and end at the same vertex or $(\mathbf{v}_i, \mathbf{v}_i)$ is not possible in this case. Since the graph is bi-directional $\mathbf{a}_{ij} = \mathbf{a}_{ji}$.

An adjacency matrix is a square matrix **A** used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices $(\mathbf{V}_i, \mathbf{V}_j)$ are adjacent or not, in the graph. If vertex $\mathbf{v}_i$ is adjacent to vertex $\mathbf{v}_j$, then $\mathbf{a}_{ij} = \mathbf{1}$ and $\mathbf{a}_{ji} = \mathbf{1}$, if they are not connected then $\mathbf{a}_{ij} = \mathbf{0}$ and $\mathbf{a}_{ji} = \mathbf{0}$. In the special case of a finite simple graph, the adjacency matrix is a (0,1)-matrix of order **|V|** with zeros on its diagonal $\mathbf{a}_{ii} = \mathbf{0}$ which represents loop condition.

We have considered simple graphs in our study throughout the paper.

II. Methodology

Graphs taken into consideration here are drawn or printed on a ruled sheet of paper. The graph, we assume, is a part of the image and has two different components namely vertices and edges.

Approach:

Since the graph and the background i.e., the ruled paper, both have lines, we will have to differentiate between the two. So we have to remove all the ruled lines in order to extract the graph from the sheet of the paper.

Removal of background ruled lines:

When the term "removal of background" comes into picture in image processing, image segmentation and image thresholding [2] is what is done and thus has been implemented here. Image thresholding is a simple way of partitioning an image into a foreground and background. This technique is a type of image segmentation that isolates objects by converting grayscale images into binary images. This converts an image from color to black and white, by setting every pixel below a threshold to black and above it to white.

Let's assume threshold for a given image be **t**

$$\mathrm{I} = \begin{cases} i = 0, & i < t \\ i = 255, & i > t \end{cases}$$

where $\boldsymbol{i}$ is the intensity of a pixel before thresholding, $\boldsymbol{I}$ is the intensity of the pixel after thresholding. **"0"** is the black intensity level while **"255"** is the white intensity level.

Using the thresholding toolbox [3] provided by matlab, the image is analyzed and the best threshold value is chosen by setting histogram values of the image such that the intensity of the ruled lines lie above the threshold value and hence are set to while. Similarly, the intensity of the pixels containing the edge of the graph is lower than the threshold value and thus set to black.

In the figure shown below, the intensity of pixels of ruled lines and the remaining paper are above 45 (approximately) and those of the graph edges are below 45. So 45 is the chosen threshold for image segmentation.

This step generates a binary image of the original image which is extracted graph. This binary image is like a graph on a plane sheet of paper and thus can be used to calculate the adjacency matrix.

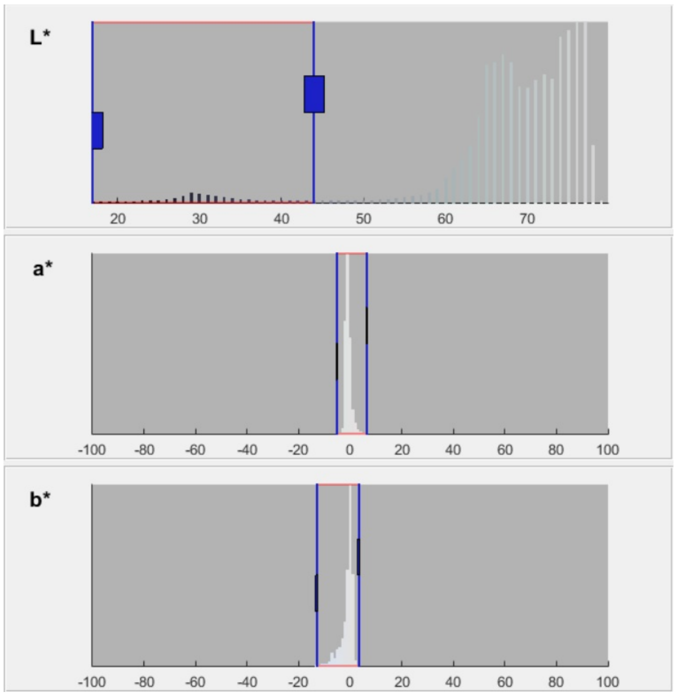


Fig. 2. Histogram of image

III. Algorithm

[BW, maskedRGBimage] = createMask()

- Convert RGB image to chosen color space (L*a*b*)
- Define thresholds for channel 1,2,3 based on histogram settings
- Create mask based on chosen histogram thresholds
- Initialize output masked image based on input image
- Set background pixels where BW is false to one.

Implementing the function:

- image = imread()
- [BW, maskedRGBimage] = createMask(image)
- imshow(BW)
- imshow(maskedRGBimage)

Result will be a binary mask BW and a composite image maskedRGBImage, which shows the original RGB image values under the mask BW.

IV. Result

Complexity Analysis: $\mathbf{O(n^2w^2)}$

$\mathbf{n^2}$**: size of image**

$\mathbf{w^2}$**: size of window**

For every pixel in the image, we have to slide the window over it. Within the window, cost of calculation of intensity of pixels will be in $\mathbf{O(w^2)}$. The window will slide over the image of size n2 incurring an $\mathbf{O(n^2)}$ complexity. Therefore, total complexity is $\mathbf{O(n^2w^2)}$.

The code was tested against various test cases. Picture of graphs were clicked under various lighting circumstances.

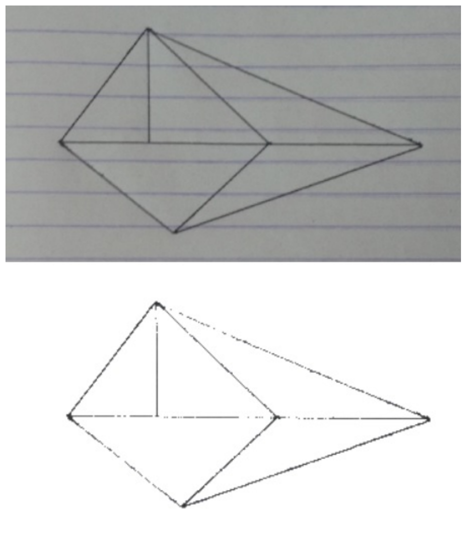


Fig. 3. (top) input image, (bottom) output image

Thickness of edges of the graphs were also varied from very thick to almost comparable to the thickness of the ruled lines.

The program was tested and following results were obtained:

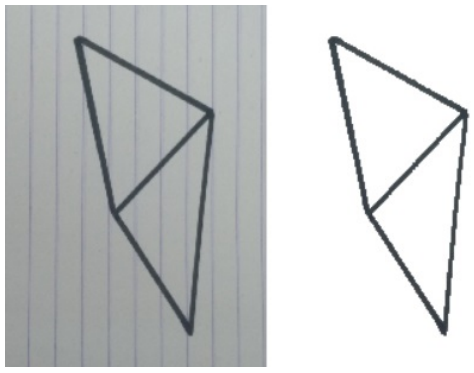


Fig. 4. (Left) input image, (right) output image

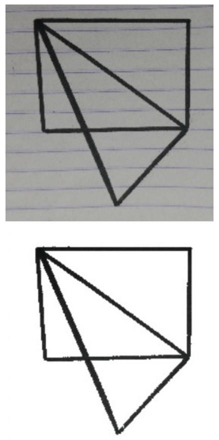


Fig. 5. (top) input image, (bottom) output image

V. Discussion

From the above results, it was inferred that when the thickness of the edges of the graph was greater when compared to ruled lines, the segmentation of graph from the background was efficient.

But as the thickness was made comparable to the ruled lines (as depicted in Fig 3), edges obtained in the output image were not smooth. Discontinuities were observed in parts of the edges, or in some cases, few ruled lines persisted in the output image as the background and foreground intensities became comparable.

In other cases, the edges of the graph in output images came zig zack. This is due to presence of a part of ruled line as well as part of the edge in a single pixel.

Therefore, if the picture of the graph is taken in appropriate light and the edge thickness is greater than the thickness of the ruled line, the code works well. The output obtained can be further used in other projects to generate adjacency matrix as discussed in paper [4].

References

[1] [Online]. Available: https://en.wikipedia.org/wiki/Graph_

[2] T. R. Singh, O. S. S. Roy, T. Sinam, and K. Singh, “A new local adaptive thresholding technique in binarization.” 2012.

[3] [Online]. Available: https://blogs.mathworks.com/pick/2005/02/02/interactive-image-thresholding-tool/

[4] “Generation of Adjacency Matrix from a Hand-drawn Simple Graph.”