**ABSTRACT**

Image Processing problems involve the recognition of certain features from the image provided by the user. This project involves the recognition of the rectangular layout from the image and detection of the horizontal and vertical edges from the layout. The Image Processing algorithms are implemented using the MATLAB Software.

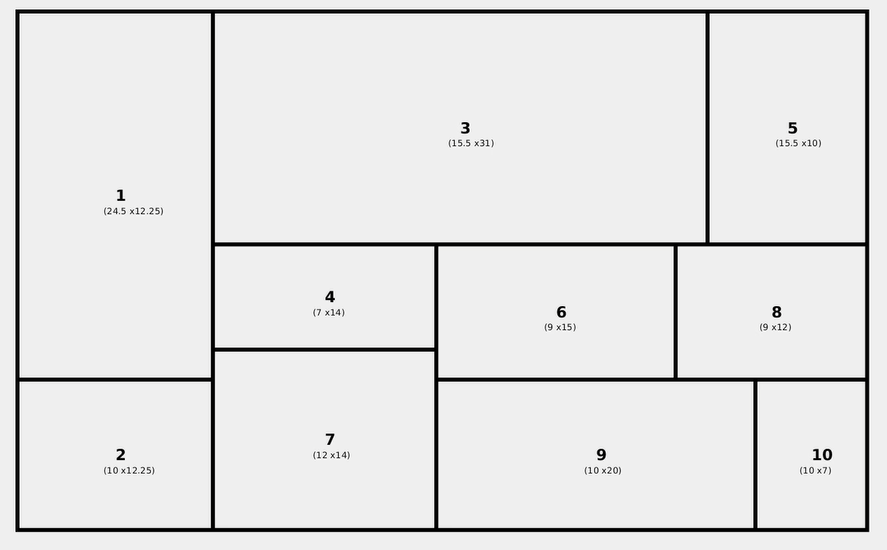
The objective is to obtain the encoded matrix from the rectangular layout, depicting the indexing of the rectangles and their adjacency relations using the grids formed by the vertical and horizontal lines and then obtain the dimensions of all the rectangles. The dimensions of the rectangles will be used to find their aspect ratio which can be used in many different problems.

Contents –

1. Introduction …………………………………………………………………. 5
2. Image Recognition
   1. Detecting horizontal and vertical edges………………… 6
   2. Detecting edge end point coordinates………………….. 7
   3. Finding the grid lines…………………………………………….. 8
   4. Indexing the rectangles…………………………………………. 9
   5. Finding the dimensions…………………………………………. 10
3. Summary……………………………………………………………………….. 11
4. Appendix……………………………………………………………………….. 12
5. Introduction –

The Image Processing Algorithm involves 5 major steps starting with the image provided by the user and finding the encoded matrix and the dimensions of all the rectangles. The image should necessarily be a rectangular layout which is the partition of a rectangle into a finite set of interior-disjoint rectangles.

The first step is to convert the image to a binary image and then extract two images from the image provided, one containing only the vertical edges and the other one containing only the horizontal edges of the given rectangular layout. The next step is to find the coordinates at the end points as well as the intersection points of the vertical and horizontal edges. Then, the horizontal grid lines are obtained using the location of the horizontal edges in the image and similarly the vertical grid lines are obtained. After the grid lines are obtained, the indexing of the rectangles is done column-wise and their dimensions are stored along with their index. The Algorithm has been tested successfully on different images. The algorithm has been implemented using the sample image shown below.



2.1. Detecting Horizontal and Vertical Edges –

The input image is read using the **imread** function in MATLAB and converted to a binary image containing 0’s and 1’s using the **imbinarize** function. The **bwareaopen(BW, p)** function is used to removes all connected components (objects) that have fewer than P pixels from the binary image BW , producing another binary image, BW2 . This operation is known as an area opening.

The **imgradient** function returns the gradient magnitude and the gradient direction of the binary image. An image Gradient is a directional change in the intensity of colour in the image. The orientation is in the direction that is **perpendicular** to the edge. To find vertical lines, the direction that is perpendicular to a vertical line is horizontal, which is 180 degrees or -180 degrees with respect to the Cartesian plane. As such, for each orientation of the edge points that are detected, if the orientation is either -180 degrees or 180 degrees, then vertical lines are present. The vertical edges are obtained using the **bwareaopen** function by removing connected components having fewer than 30 pixels from the image having gradient angle around 180 or -180 degrees.

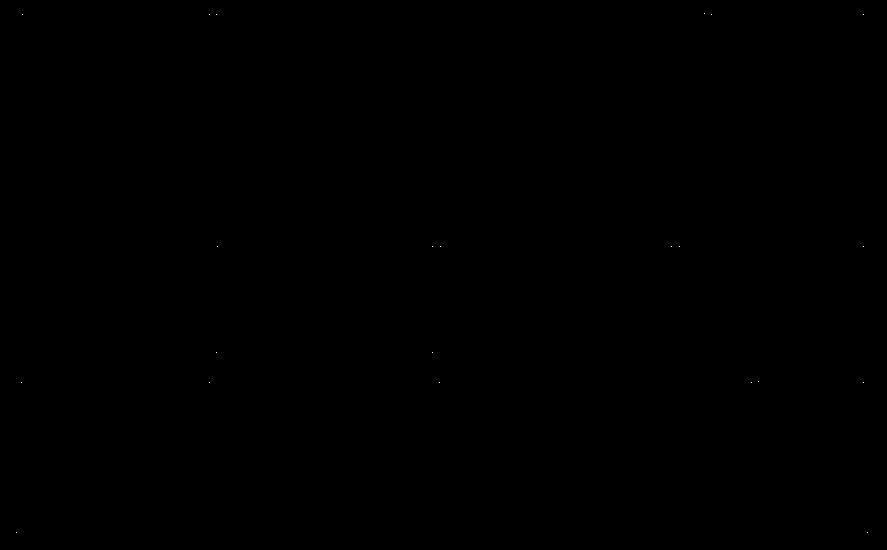
The image is then rotated by 90 degrees and the same functions are applied to obtain the horizontal edges of the image. The detected horizontal and vertical edges of the sample image are shown below.





2.2. Detecting Edge end point Coordinates -

To create a skeletonized image, the **bwmorph** function is used on both the images of horizontal and vertical edges with the **skel** operation. Then, the skeletonized images are fed to the **bwmorph** function with the **endpoints** operation which will remove all the edges except their endpoints.





The above images contain only the endpoints of the edges.

Now, the x and y coordinates of the points having size greater then 0 are stored in an array using the **find** function where the 1st column stores the x coordinates and the 2nd column stores the y coordinates of the points.

2.3. Finding the Grid Lines -

The coordinates of the points are sorted in ascending order using the **sort** function.

Now, in many cases we observe that there are 2 or 3 endpoints differing by a few pixels corresponding to a single endpoint. That is due to the thickness of the edges in the given image. We resolve this problem by merging the 2 or more points of the same endpoint to a single coordinate.

Now that the coordinates of endpoints are extracted, we form the x and y grid line arrays by simply counting the number of different x and y coordinates of the end points.

For the image used for reference, there are 5 horizontal grid lines and 7 vertical grid lines. So, we can conclude that there will be 4\*6 = 24 rectangles in the layout.

2.4. Indexing the Rectangles -

The pixel list of all the points in an edge (horizontal or vertical) are extracted using the bwconncomp and regionprops function. The **bwconncomp** function returns the connected components in the binary images containing only the horizontal and vertical lines. It returns a structure with the fields: Number of connected components, Image Size, Number of objects and Pixel list. Now, the result of the bwconncomp function is passed to the **regionprops** function with the **pixellist** property. The regionprops function returns measurements for the set of properties specified for each connected component in the binary image.

So, the list of pixels for each edge is obtained. The x and y grid coordinates are checked if they belong to any of the pixel lists obtained. If yes, then there is a line joining those two points. Hence, we form an array of the x, y coordinates of the both the end points where an edge is present in the layout.

Now, we traverse column-wise and check if there is an edge to the right and an edge below the rectangle then we store the top-left and right-bottom coordinates of the grid and index it to the counter initialized to 1. After each assignment, the counter is incremented. Following the process, we index all the rectangles.

Now, we traverse through the grids having an edge to the right but no edge to the bottom. For such grids, we index them with the same number as the grid below them. Similarly, for the grids having an edge to the bottom but no edge to the right, we index them with the same number as the grid to their right. Lastly, for grids having no edge to the right and bottom as well, we index them with the same number as the grid to their right or bottom.

The indexing of the rectangles in the given layout is shown below.

1 6 6 6 8 8

1 3 5 9 9 9

1 4 5 9 9 9

2 4 7 7 7 10

2.5. Finding the Dimensions –

Now, that the indexing is done, we traverse through each of the rectangles and store the top-left and right-bottom x and y coordinates in an array. Then we extracted the height of the rectangles by subtracting the y coordinates of the two points and the width by subtracting the x coordinates of the two points.

These dimensions will be used to calculate the aspect ratio of all the rectangles, which can be used in further implementations.

The dimensions of the rectangles in the given image are:

dim =

368 196

151 196

106 223

180 223

135 240

233 494

151 319

233 160

135 191

151 112

Here, the rows represent the height and width of the rectangle having index equal to the row number. The first column represents the height and the second column represents the column.

3. Summary –

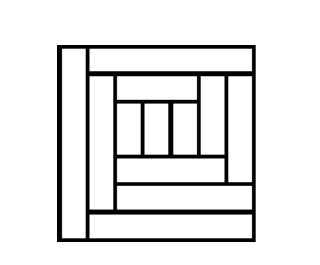
We started with the rectangular layout image and obtained the encoded adjacency matrix and the dimensions of the rectangles.

The algorithm is summarized as follows:

1. The vertical and horizontal edges of the image is obtained.
2. The end points of all the edges are obtained and the grid lines are located based on the presence of horizontal and vertical edges.
3. The rectangles are indexed starting from 1 in a column-wise fashion.
4. The dimensions - height and width of all the rectangles are found using the x and y grid lines.

4. Appendix –

The Algorithm is also implemented on other images where it successfully detects the encoded matrix and the rectangle dimensions.



The encoded matrix is the following:

1 9 9 9 9 9 9

1 2 5 5 5 7 10

1 2 3 4 6 7 10

1 2 8 8 8 8 10

1 2 11 11 11 11 11

1 12 12 12 12 12 12

The detected edges are:

