Handwritten Flowchart Generator

A Project Report Submitted
in Partial Fulfillment of the Requirements
for the
Digital Image Processing Course

by

Sajith Kumar Erasani and Kalle Karthik
(111701012 and 121701010)

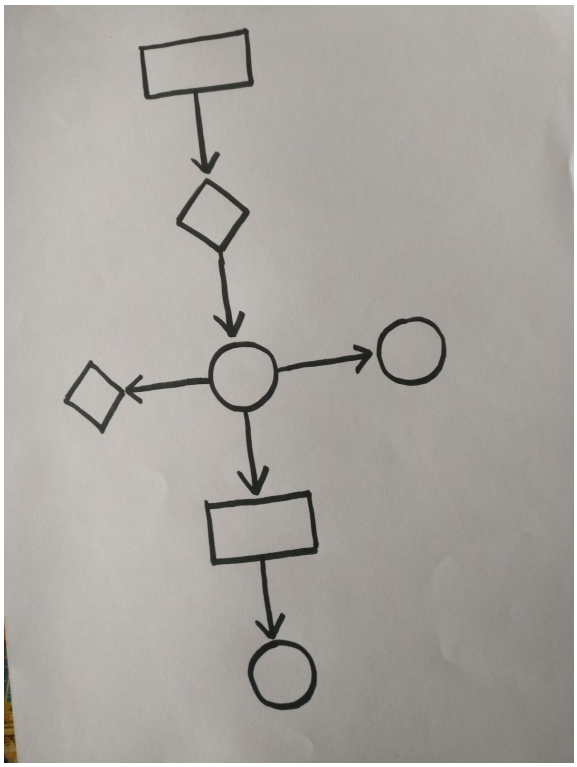
INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

Introduction

In real life, it could be super time consuming to draw up a well-organized flow-chart onto our documentation files or presentation. So, it could be very useful if we can take a photo of the flow chart drawn on paper and convert it into a digital flowchart. In daily life like while documenting or while taking notes, this tool can be very helpful and reduces time consumption.

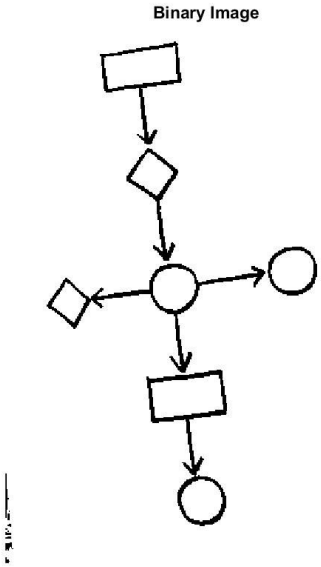
Methodology and Image Processing Techniques:

1. **Input:** The input is an image in .jpg format as an RGB.

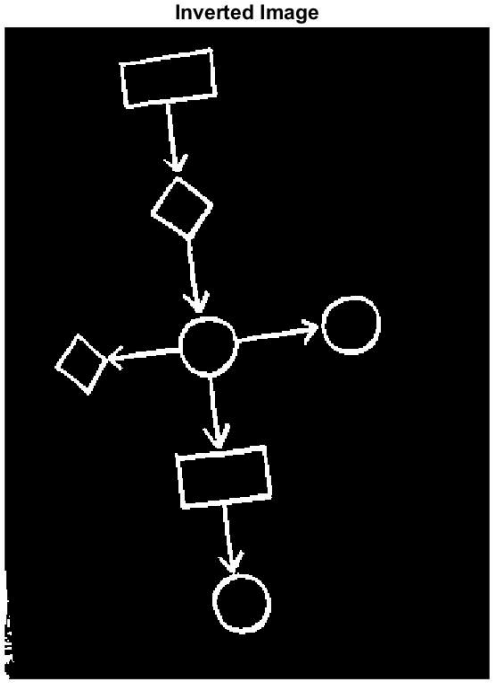


2. **Convert and Resize:** Convert it into GrayScale and resize so the image aspect ratio is maintained.
3. **Adaptive Thresholding and Binarization:** Use a locally adaptive threshold and perform binarization, to accommodate for different lighting conditions in different areas.Tune the filter size carefully since it is sensitive to contrast and

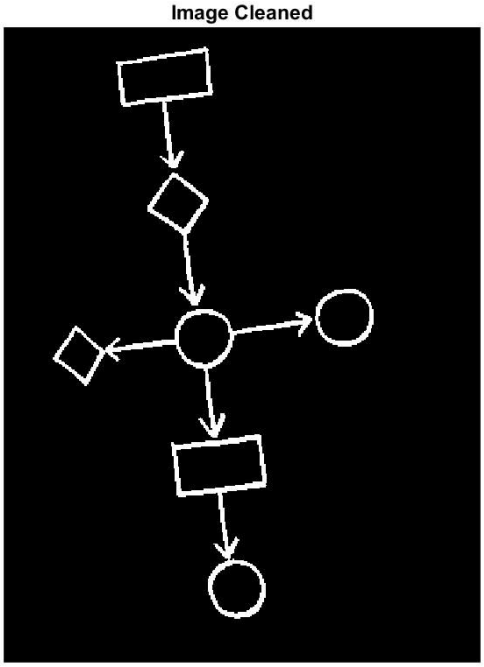
luminous changes. (In this case, we have an image with black front-ground flowchart shapes and white background)



4. **Invert the Image:** Perform a bitwise inversion, to swap the front-ground and background colors.

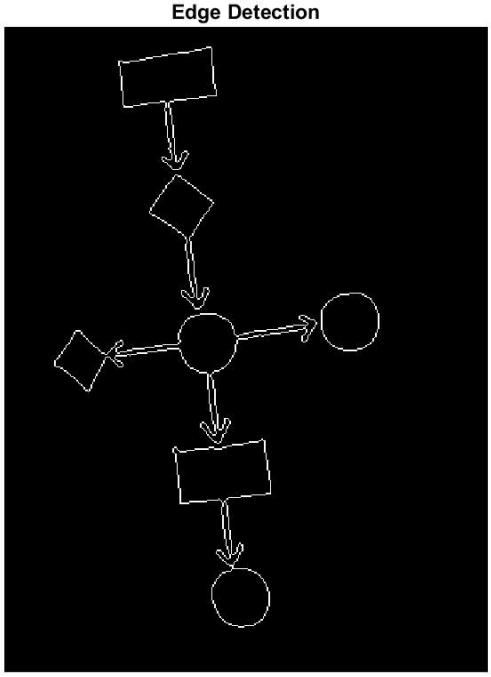


5. **Remove noise at corners:** There can be some noise generated during binarization because of stain on the paper or the extra region covered out of the paper. This can be removed by connected components, calculating their areas, labeling them, and removing ones with areas less than the threshold we give. This will remove all the unnecessary white regions and creates a clean black background.

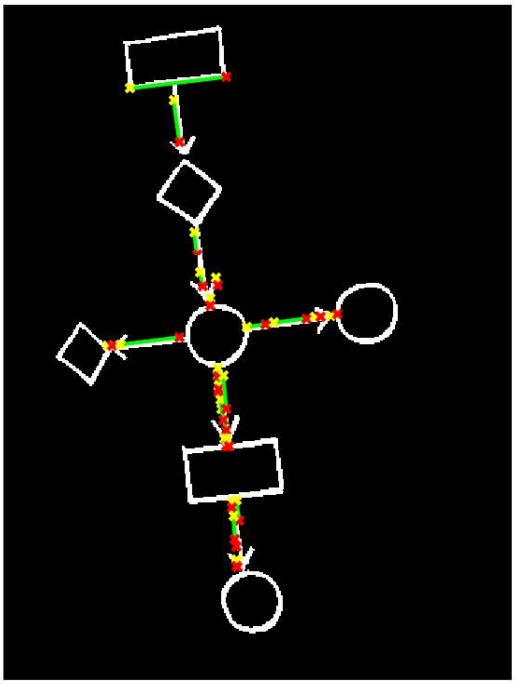


6. **Hough Transform and Rotation:** Fill all the shapes(holes) in the image, and find edges using this filled image with zero cross edge detection. Use these edges and perform a Hough transform to extract their angles and find out the most commonly occuring angle. Then the image is rotated to restore its correct orientation.

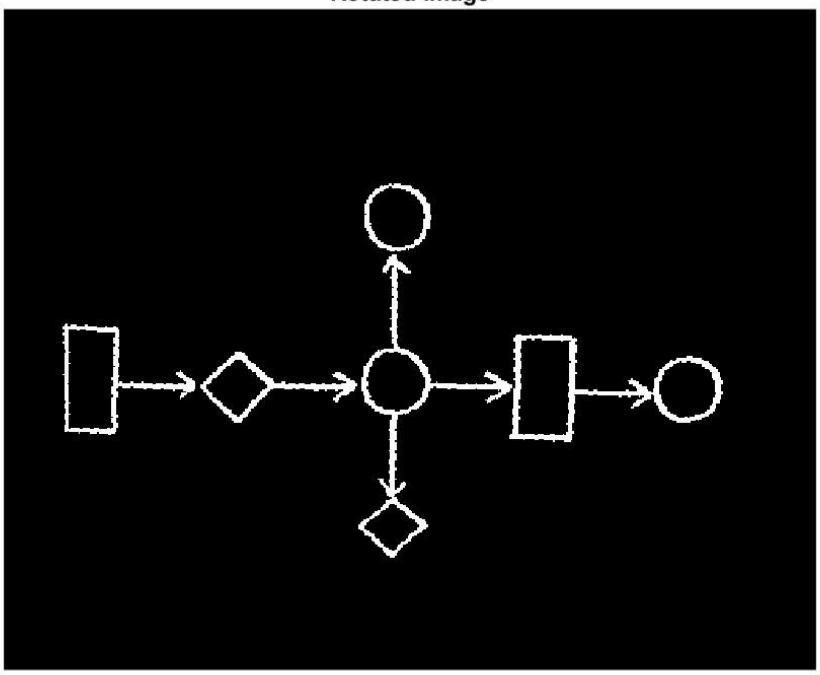
Edge Detection



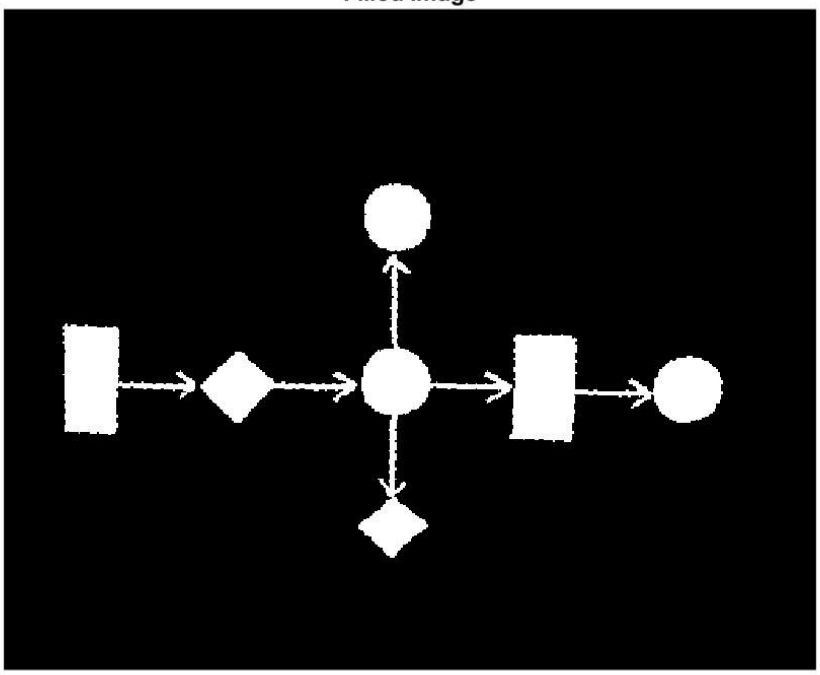
Detected Lines



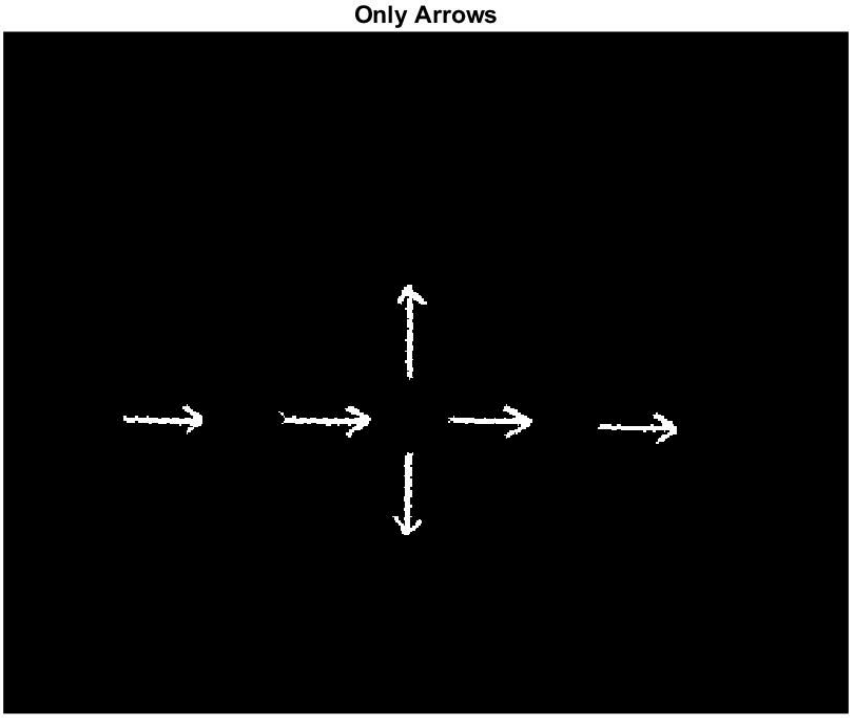
Rotated Image

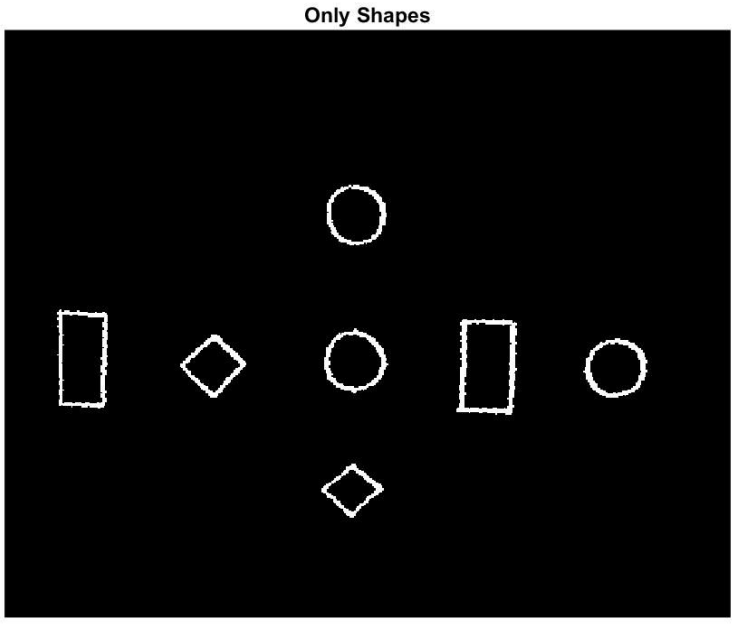


Filled Image



7. **Decomposition into arrows and shapes:** Now we have only arrows and filled shapes from the rotated image. We can decompose the flowchart by removing the arrows by Morphologically Opening Image. It will gradually remove the white space till the arrows will disappear and then widen to recover the shapes to their original size. Then retrieve the arrows by subtracting this image from the image we got from the last stage. And then subtract the arrows from the filled,rotated image to get all the shapes. Now if we observe the corners of the shapes, there is some noise caused by the open image operation. So, we need to redo the operation in step 5 with appropriate threshold to remove that noise.

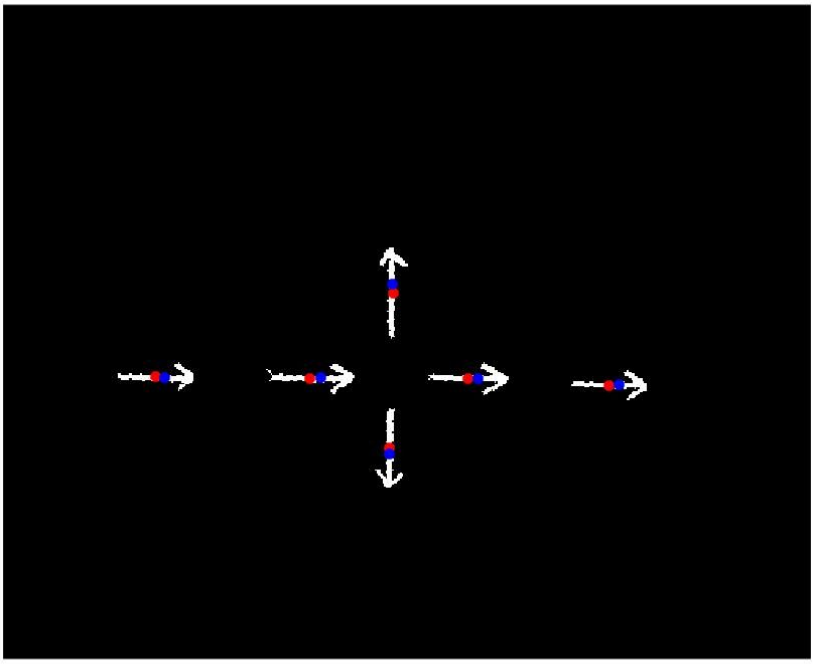




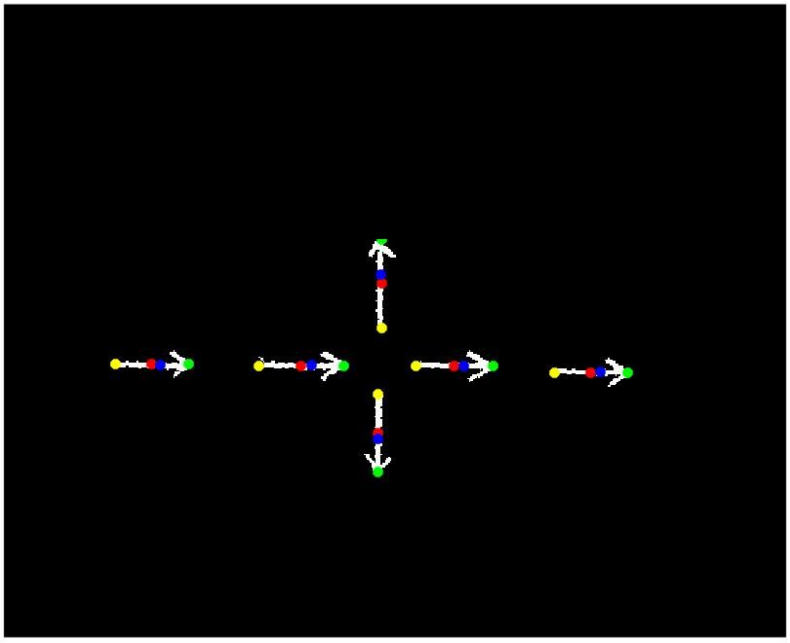
8. **Decompose Shapes into Circles, Rectangles and Diamonds:** Now we are gonna to find boundary boxes(smallest rectangular region enclosing each shape) to determine the shapes.Now we can filter out each shape one by one by using the shape properties, we can find the circles by using a ratio of the perimeter and area, and we can find rectangles and diamonds by calculating the ratio of area of boundary box to the shape. For rectangles the ratio is close to 1 while for diamonds the ratio is around 0.75.

9. **Find Arrow Orientation, Arrowhead and Arrow Tail:** For each arrow, compute the boundary box and then find the centre and centroid of that boundary box. Given the centre the centroid the arrow head must be oriented towards the centroid. By comparing the positions of center and the centroid of -the shape, the corner that is closest to the centroid can be found. So, the closet corner can represent the head of the arrow, and the opposite corner would approximate the tail of the arrow.

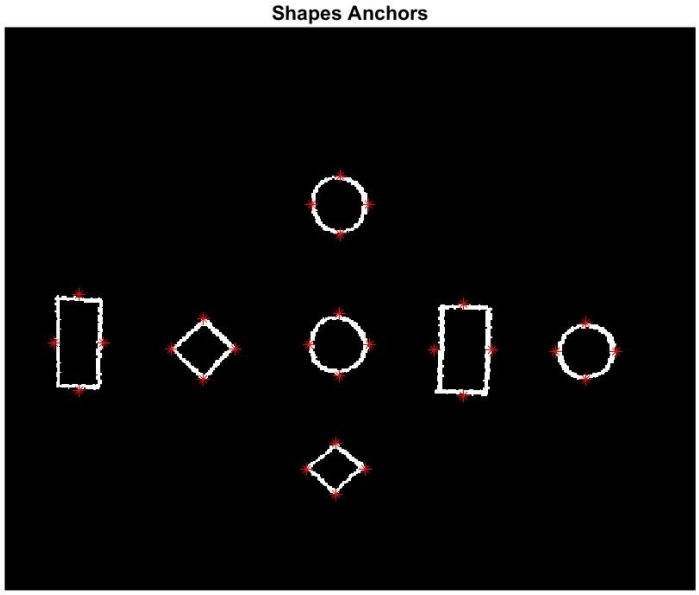
Arrow Centres and Centroids

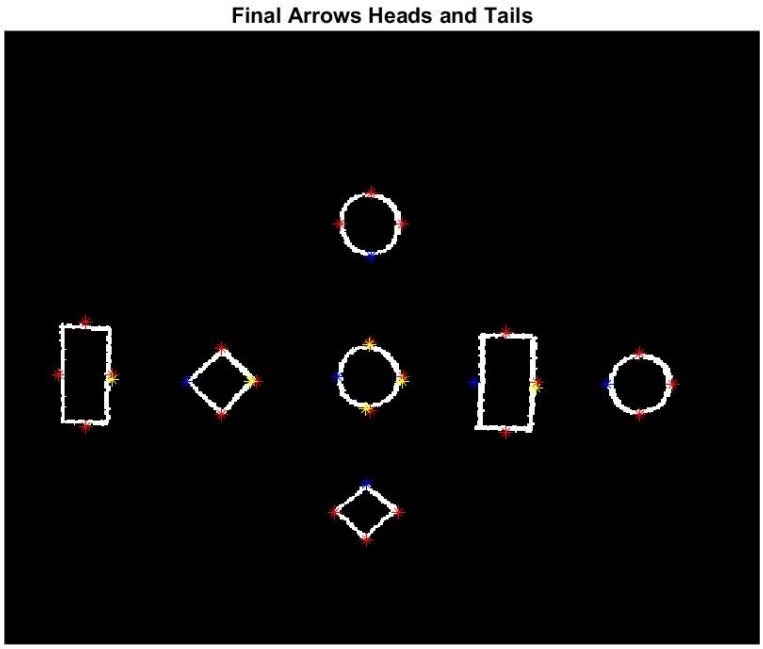


Arrows Heads and Tails



10. **Find Closest Shape to Arrowhead:** For rectangles anchors are set on the centre of each side. For diamonds anchors are set on each of its corners. From the data we have on the arrowheads and arrow tails, we loop through all corners and find the anchor closest to it.





11. **Reconstruction of arrows and shapes:** Finally using different tools provided by mathworks we can reconstruct all the shapes and arrows.

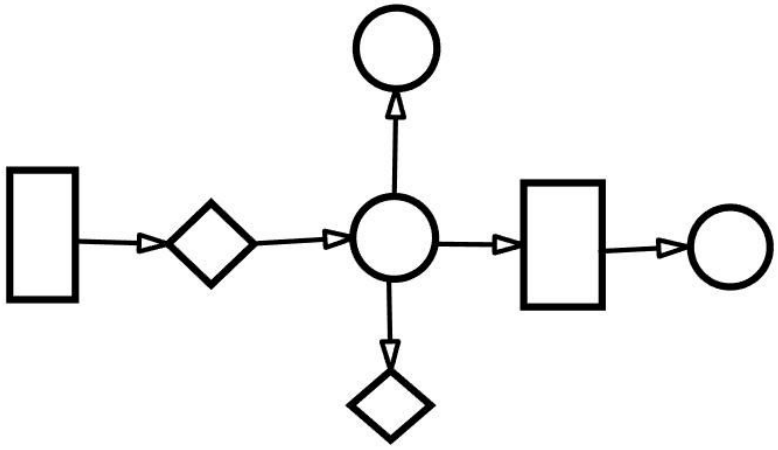
    Rectangles can be drawn with “rectangle()”

    Diamonds can be drawn with “patch()”

    Circles can be drawn with “viscircles()”

    Arrows are drawn using an external file provided by mathworks “arrow.m”

Output Flowchart



CONCLUSION:

This Flow Chart Generator can detect rectangles, circles and diamonds. And it is very easy to use and user friendly.

Presentation: This presentation is made by both of us. First half is presented by Kalle Karthik and Second half is presented by Sajith Kumar.

Link for video:
https://drive.google.com/file/d/1wLTBoIdRgyNAQbRotMOei-6zrF8SlwNj/view?usp=sharing

Link for pptx:
https://drive.google.com/file/d/10pW-7xeLGslOfGKDlXXA2kfYOJVKU8JG/view?usp=sharing

REFERENCES:

1. **Hsu_Lyu_Zhang_Yu.pdf (stanford.edu)**
2. **sayhitosandy/Flowchart_Converter: Automatic Handwritten Flowchart Converter (github.com)**
3. **arrow - File Exchange - MATLAB Central (mathworks.com)**