IMAGE PROCESSING USING MATLAB

MINIPROJECT: TRAFFIC SIGN DETECTION

-Adithya TG
PES1UG21CS035
-Adhesh Dheeraj
PES1UG21CS029
-A B Sagar Yadav
PES1UG21CS743

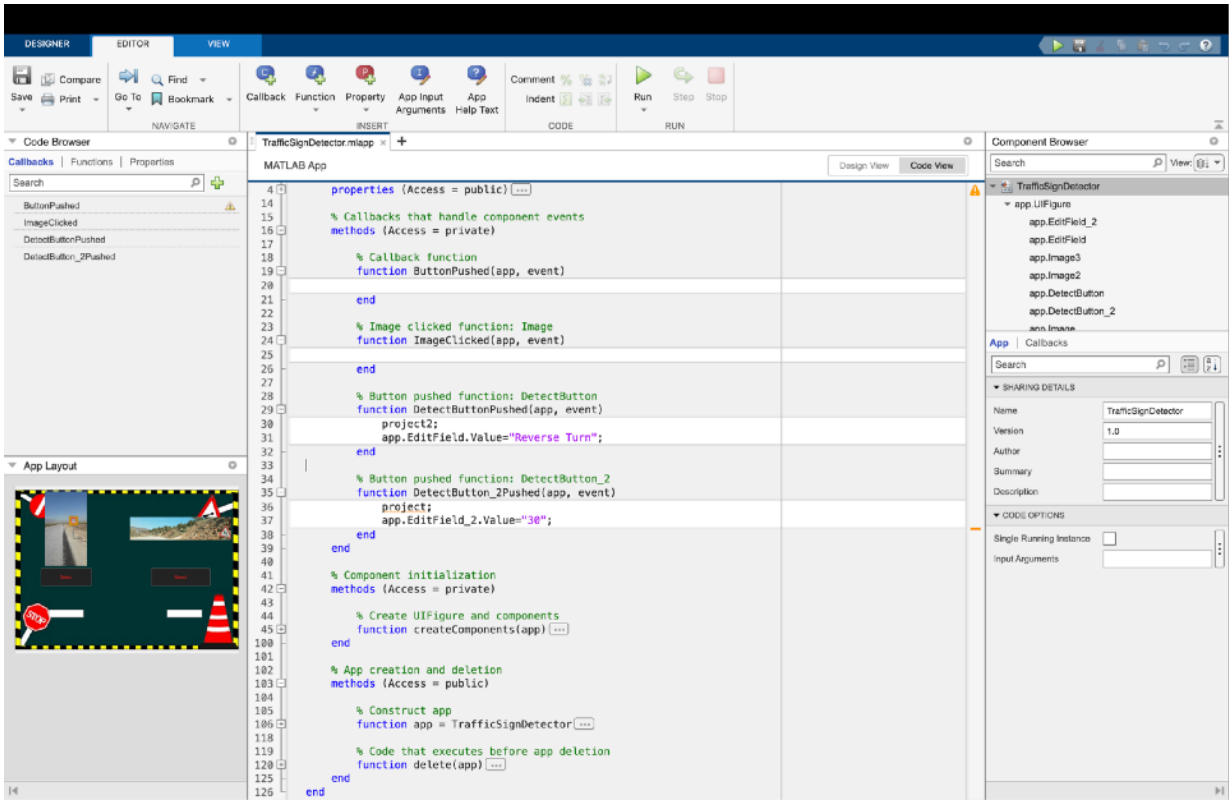
Abstract:

This code presents an automated image processing algorithm to detect and localize red objects in images. The algorithm first extracts the red pixel regions from the input image by applying thresholding in the HSV color space. Next, edge detection using the Sobel operator is employed to identify object boundaries. The largest objects are then selected and processed for efficient analysis. The code iterates through various scales and positions to fit a polygon (four-sided shape) to the object. The best-fitting polygon is determined based on the number of overlapping pixels with the object. The scale and position of the best fit are recorded to indicate the object's localization. The algorithm is designed for real-time processing and produces visually interpretable results, making it useful for applications involving red object detection and localization in images.

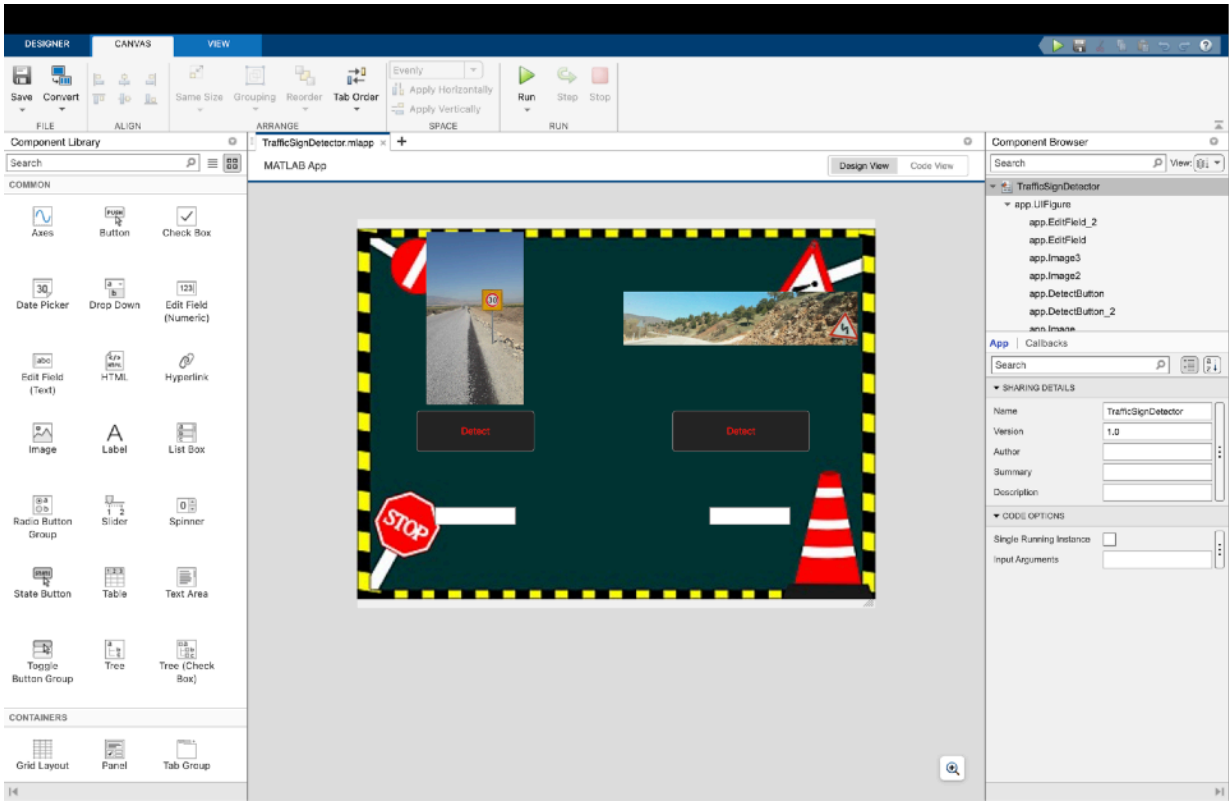
What it does?

1) The code starts by reading an image and extracting its dimensions.
2) It then creates a blank image R1, which will store only the red pixels from the original image.
3) The code converts the original image from RGB to the HSV color space and sets some thresholds to identify pixels that are within a certain color range (Hue, Saturation, and Value). These pixels are then copied to R1.
4) Next, it applies edge detection on the red pixel image (R1) to identify the edges of objects.
5) The code creates a larger binary image (BW2) by filling the holes in the detected edges.
6) It selects the three largest objects from the binary image (BW2) and stores them in F1.
7) The code then extracts the relevant parts of the image based on the positions of the largest objects and reduces its resolution for efficiency.
8) For different scales and positions, the code tries to fit a polygon (specifically, a four-sided shape) onto the extracted image. It calculates the number of overlapping pixels between the polygon and the image for each scale and position combination.
9) The best-fitting polygon (i.e., the one with the most overlapping pixels) is chosen, and its scale and position are recorded.
10) Finally, the original image, the red pixel image, the edge-filled image, the largest object image, and the best-fitting polygon overlaid on the processed image are displayed in separate subplots.

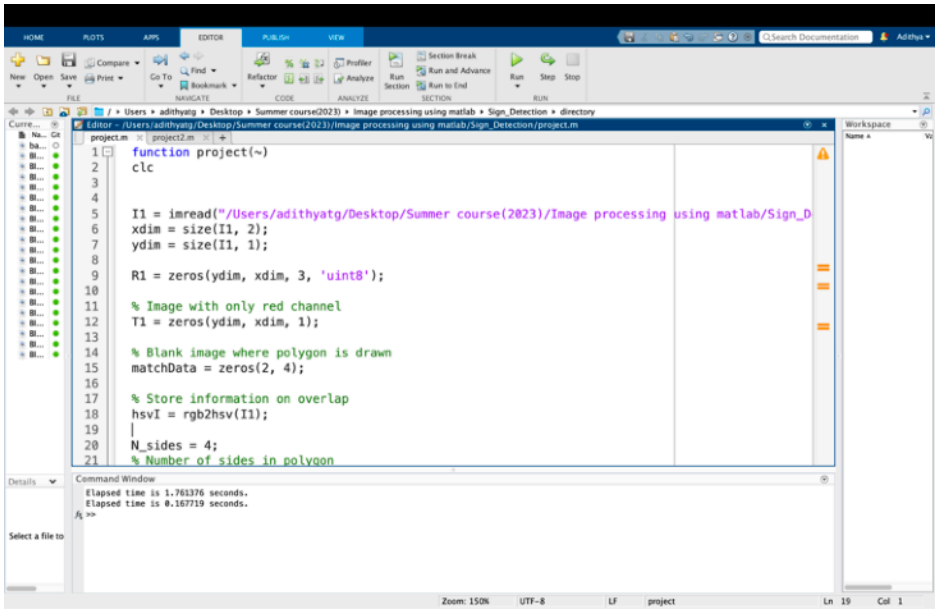
Screenshot:



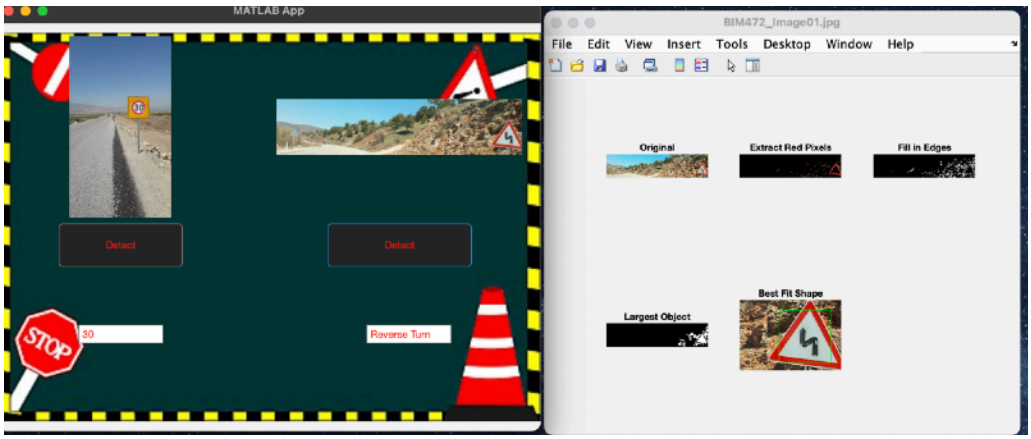
App code



App layout



Backend code



App with output

App