RUBIK'S CUBE RECOGNITION AND IDENTIFICATION

OBJECTIVE: To recognize a Rubik's cube in an environment and to identify all the colours for each face using MATLAB.

BLOB (Binary Large Object): BLOB refers to a group of connected pixels in a binary image. An image consisting of some solid portions, when converted into its binary domain (with some threshold) has connected white pixels corresponding to those solid portions. These white regions are known as BLOBS.

PRINCIPLE: The algorithm employed here for Rubik's cube recognition uses BLOB extraction and BLOB classification techniques.

APPROACH: Each face of Rubik's cube comprises of 9 solid coloured squares. When the image of any face is taken and converted into its binary domain using some optimum threshold, then the 9 colours can be identified as white squares and hence are BLOBs. But, the direct conversion of the RGB image of a face into its binary domain may result in loss of some colours like Blue and green in dim light, as they appear black and cannot be identified as BLOBs. For the authentic identification of all the colours of the face, it's three component R indexed, G indexed and B indexed images are preferred over the single RGB image. With this each colour is identified in at least one component image and hence there will be no loss of colours. This sums up BLOB extraction. But image also contains other solid objects which will be identified as BLOBs. To separate these from the required ones, the geometric properties of BLOBs are used. We can filter out the unnecessary BLOBs using the measures like Area, dimensions and centroid.

ALGORITHM:

- Take R, G and B indexed images of a face and convert each into its binary domain.
- Extract BLOB features for each indexed image using the MATLAB toolbox function called *regionprops.* For a binary image, this function outputs a struct of features for each BLOB present in the image. The basic features include Area, Centroid, Majoraxislength, Minoraxislength, Boundingbox surrounding each blob and so on.
- Impose a constraint (threshold) on the area of a BLOB to exclude the unnecessary BLOBs.
- Using the property that each colour on the face of a cube has a square geometry, impose a constraint that Majoraxislength(width) equals Minoraxislength(height) for the required BLOBs. This is done for all the 3 images and bounding boxes can be drawn around the identified squares.

- After the identification of all 9 colours, the image is cropped to contain only the cube. This requires X_{min}, Y_{min}, width and height .

  (X_{min}, Y_{min})= Coordinates of centroid of the top left square.

  Width= X_{max} - X_{min}

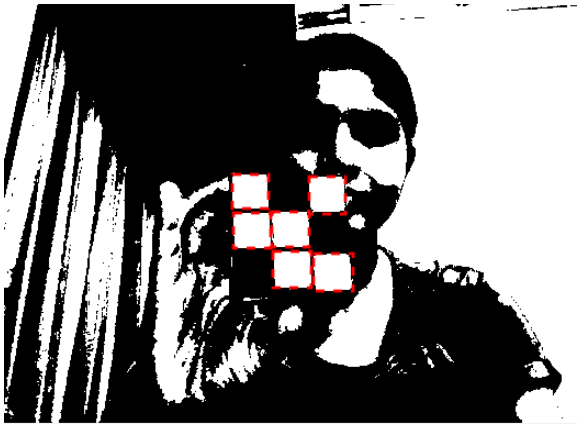
  Height= Y_{max} - Y_{min} (These coordinates can be obtained by sorting all the X coordinates and Y coordinates of Centroids for all the required blobs.)

- The cropped image can now be sent for colour identification.

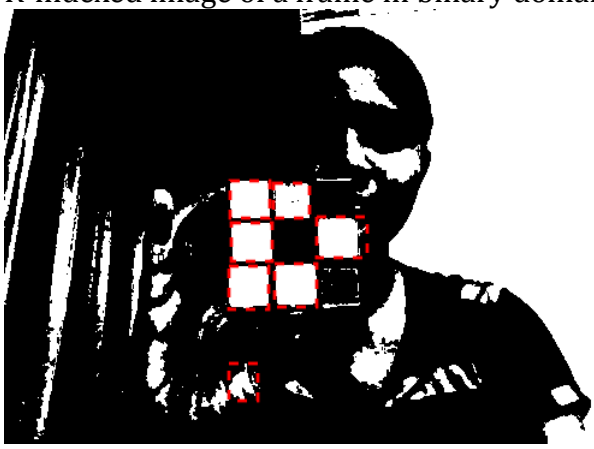
ELIMINATION OF BACKGROUND SQUARES: In order to eliminate any unnecessary squares from the background, median distance technique is employed. In this technique, median centroid of all centroids of identified squares is taken and the distance of all square's centroids from the median centroid is computed. Assuming all squares 9 squares on the face of the cube are identified (the median would mostly be the square in the middle) and using the fact that all 9 squares are closely packed, the squares with centroids far away from the median, which do not come in the given distance-threshold range are eliminated. Thus only the closely packed 9 squares are identified therefore, identifying the face.

IDENTIFICATION OF COLOURS OF A FACE: The cropped face of the cube that is detected in the random environment using the RGB extraction is the input to the colour identification algorithm. The cropped image is converted from RGB to HSV space for reliable colour identification. The face of the cube is discretized into 9 areas, each area representing one colour. Each region ideally is a single colour, so the median colour value of the region should provide the actual colour for the entire region. Hence the median 'hue' value is found for the given region and based on the hue, colour of the region is identified. A range of hue values are found out experimentally for each colour in different lighting conditions and these are used in colour identification.

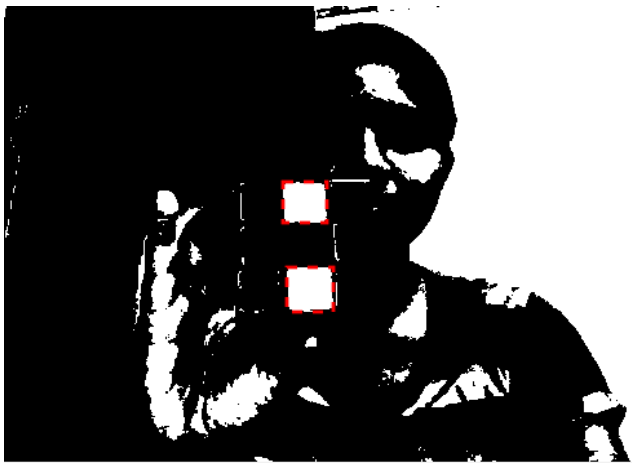
ILLUSTRATION:
Following figures depict the BLOBs in each indexed image and the final cropped face of the cube.



R-indexed image of a frame in binary domain



G-indexed image of the frame in binary domain



B-indexed image of the frame in binary domain

The cropped image of the cube separated from the environment.

- It can be observed that from the R-indexed image, yellow, orange, red and white were identified as BLOBs , from the G-indexed image, green, yellow, white and blue(here) were identified and from B –indexed image , blue and white squares were detected.
- So, each colour is identified in at least one image without loss of any colour and the final cropped image of the cube is obtained.
- This cropped image is sent for colour identification and colours were identified using the hue values.

- Hue value range for each colour:

  Orange(o) : 0.01-0.09

  Yellow(y) : 0.1-0.28

  Green(g) :0.3-0.49

  Blue(b) : 0.5-0.68 (Saturation > 0.5)

  White(w) : 0.6-0.7(Saturation <0.5)

  Red(r) : 0.8-1.0

- The colours for the above face are identified accurately as

  ```
  y b o
  y r g
  g w o
  ```

- Digital Signal processing refers to application of various algorithms and techniques on the samples of a continuous variable in time, space and frequency domains. DSP includes audio and speech processing, digital image processing, signal processing for bioengineering and so on. In the current project of Rubik's

cube recognition and identification, many image processing techniques were applied which include binarization of an image, extraction and classification of BLOB features in a binary image and HSV domain of an image for colour identification. So, this can be viewed as an application of Digital Signal Processing.

- The MATLAB toolbox function ***regionprops*** is used for BLOB extraction here. There are many algorithms employed for BLOB extraction and fall under Connected component Analysis domain. A BLOB consists of connected pixels and whether two pixels are connected or not is defined by the 2 connectivity techniques: 4 connectivity and 8 connectivity.

  4-connectivity: Group of white(square) pixels are said to be 4-connected, taken any two pixels in that group, all the neighbours connecting these two share only an edge.

  8-connectivity: Group of white pixels are said to be 8-connected, taken any two pixels in that group, all the neighbours connecting these two share either an edge or a vertex.

  The **Recursive Grass-Fire Algorithm** is a technique employed for extracting the BLOBs using 4 or 8 connectivity methods. According to this algorithm, scanning starts from the top-left corner pixel of the binary image and proceeds to reach a white(object) pixel. A label is assigned to this white and also to each of the white neighbours(depending on connectivity method used) , proceeds further and this recursively happens for all the white pixels, where all the connected pixels get a same label and finally pixels with same label are extracted as BLOBs.

- **ROBUSTNESS IN DISTANCE OF THE CUBE FROM CAM**: To make the algorithm robust in this aspect, we need to give the area condition for filtering the squares as a function of distance of cube from the camera as change in distance causes a change in area of each square and hence a variable condition on area has to be used to identify the blobs. For this, our algorithm must make a small calculation to find the distance of the object from the cam. The focal length of the cam has to be found out first using the known distance. Let's say we have an object with a known width *W*. We then place this at some distance *D* from our camera. We take a picture of our object using our camera and then measure the apparent width in pixels *P*. This allows us to derive the perceived focal length *F* of our camera: $F = (P \times D) / W$. Using this triangle similarity technique, we can now find out the distance of the object in front of the cam using $D' = (W \times F) / P$. With this

measurement, the area condition can be specified for each distance range by doing some experimental observations.

- **NOTES**: Conversion from RBG to HSV space eases the job of colour identification. RGB values of a pixel are quite susceptible to environmental changes like light intensity. On the other hand the Hue value of a pixel, fraction indicating the closeness to a pure colour doesn't vary noticeably with lighting conditions. Hence, this space is used for colour identification, making the method robust in different light conditions.
- The ***imbinarize*** function which is used to convert an image to its binary domain uses Otsu's algorithm to find out the threshold which separates pixels into black and white in the binary space. This algorithm assumes image contains 2 classes of pixels and finds a threshold intensity such that intraclass variance among intensities in each class is minimized which is also same as maximizing interclass variance. These variances for each class should be computed by varying the threshold intensity from the minimum value to the maximum intensity value of the image. The threshold which minimizes the spread in each class is the final threshold to binarize an image.
