Nupur Mukherjee, B.Tech(Ist Year)-IT

(RA1611008010405)

TASK 1 : IMAGE PROCESSING

- Using OpenCV image processing libraries, design an implementation to detect each colored Buoy in the image :

- You are required to compute the X,Y coordinates of the centre of each buoy detected.

- Colored Buoys must be isolated from similar colored objects.

- If you have no prior experience with OpenCV , we will accept Matlab implementations as well. Nevertheless, priority will be given to design using OpenCV .

Description of solution

This task involves three things - detect Red, Green & Yellow Buoy in the image, compute the X,Y coordinates of the centre of each buoy detected & isolate Buoy from similar colored objects. I have coded in Matlab and used basic image processing functions. Using imread() I read the image 'TaskImage.png' and display it using imshow(). The size of image is 720 X 960 matrix with 3 color RGB. I isolate three colors (Red, Green & Blue) into 3 images as follow

```
r = image(:, :, 1);
g = image(:, :, 2);
b = image(:, :, 3);
```

When we see the red image we find that red Bouy is darker so our first idea could be filter pixels below some threshold. All pixels over the threshold may belong to the objects. Actually what we want is not the pixels having high red component, but we want red pixels. A pixel is red if the red component is high and the others are low. Thus using idea* a simple equation for identifying the redness of a pixel is gives as follows -

```
redness = double(r) - max(double(b), double(g));
```

I then compute the threshold. I take it as 70% of max value.

```
mxr = max(max(newred));
mask = (newred > (mxr * 0.7));
```

The image 'mask' shows isolated red Buoy.
I compute the center location of the identified red Buoy by computing the centroid of the object. (using function call computefinalxy(mask)). To compute the centriod I first compute mean and standard deviation of all pixels belonging to image 'mask'. Using standard deviation I isolate the 'outlier' pixels and compute centroid of the red object without using considering the outliers.

To identify the green Buoy I similarly compute greenness of the pixel as follows -
greenness = double(g) - max(double(b), double(r));

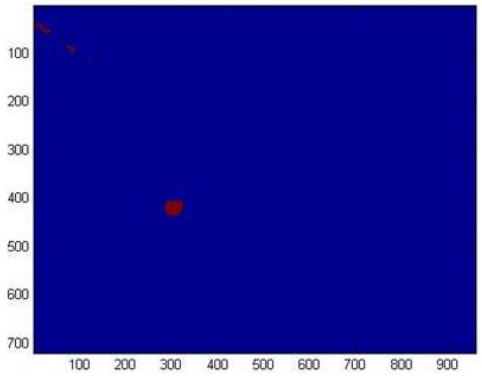
I then compute the threshold of green as 40% of max value. The threshold values are being decided based on distribution of pixel values. I used imhist() function call to displat histogram of image.

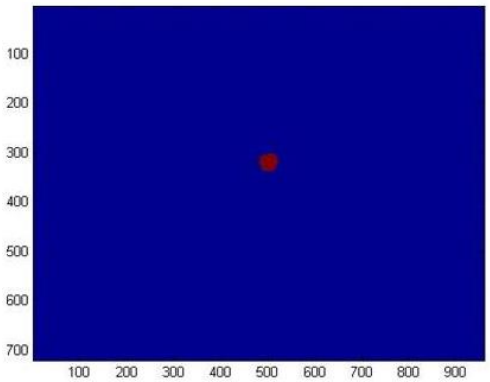
For identifying the yellow Buoy I do not follow above logic since yellow is mixture of primary color and above method may fail. I use the idea that yellow is more of red & green and less of blue. To compute yellowness of a pixel I use direct threshold on blue, red and green values of pixel as follows -
redThresh = 200;
greenThresh = 200;
blueThresh = 100;

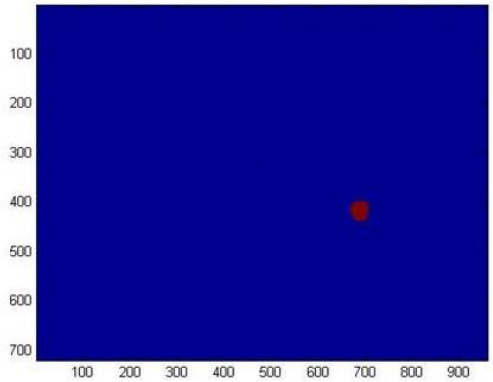
mask = (double(r) > redThresh) & (double(g) > greenThresh) & (double(b) < blueThresh)

I compute the center location of the identified Green & yellow Buoy as done in case of Red.

*https://in.mathworks.com/matlabcentral/answers/4508-modify-this-code-to-detect-yellow-colour







Coding/Algorithm Questions

Coding Problem 1

"Heading Averager": In many cases, it is useful to average a few readings from the AUV's compass before proceeding. All headings are integers in the range 0 to 360. Write a heading "averager" that accepts user input and returns an average heading when requested by the user.

Description of solution

Class ***HeadingAverager*** stores values for headings (upto 1000) & total number of heading stored till now. It also defines functions for taking user inputs, compute average heading & display results. The average heading computation is based on the idea of mean of circular quantities (like angles & time) since it is different from arithmetic mean e.g. mean of angle in degree 0 and 360 is 180 (but 360 is same 0). As the arithmetic not appropriate for angles I use the idea of converting all angles to cos & sin then compute the arithmetic mean of these points i.e.

avg cos = (cos a1 + cos a2 + ... + cos an)/n

avg sin = (sin a1 + sin a2 + ... + sin an)/n

then convert that point back angle = arctan(avg-sin/avg-cos)

Since arctan returns angle between −90 and +90 we need to appropriately modify the answer to bring it between 0 and 360. Based on the quadrant in which the average angle falls, the final answer is modified as follows bring answer numerically between 0 and 360 -

- if average angle between 0 & 90 then no modification
- if average angle between 90 & 270 then add 180 to arctan answer
- if angle between 270 & 360 then add 360 to arctan answer

Coding Problem 2

Suppose you are receiving a double from a sensor every 0.1s. Design an efficient storage algorithm for saving the most recent 64 samples using a constant sized buffer.

Description of solution

Class ***CircularBuffer*** defines buffer array (max 64 values) that stores most recent double values coming from a sensor & use it as queue (i.e. tracks front & rear of the buffer queue). It also

defines functions for adding data to buffer, read data from buffer and display current buffer state. These functions appropriately maintain front and rear based on state of the buffer (i.e. empty or full buffer) and operations being done (i.e. add data or read and remove data from buffer). They also implement wrap around effect of circular queue to maintain latest values in buffer.

While adding data check for following -

- first data then front / rear = 0

- if (rear == MAX-1) then rear = 0

- if (rear == front) (i.e. Buffer is full) then increment front also (to keep indication that buffer is full and data is now being overwritten)

While reading data (and removing after reading)

- if (front == -1) i.e. buffer is empty then give message "No Data in Buffer" and flag error

- if (front == MAX-1) i.e reached end of array then front = 0 i.e. wraparound

- if (front == rear) i.e. only one element then after read buffer is empty and front / rear = -1

While displaying the buffer I took care of cases like Buffer is empty and wraparound.

I simulated sensor data generation randomly and using probability of reading the data I read the data occasionally. This simulated different conditions like buffer getting full etc.

Coding Problem 3

Write a program to verify a Sudoku board, specifying your own input format. That is, given a completed 9x9 Sudoku board in a format of your choosing, output 'true' iff it is correctly solved

Description of solution

Class ***SudokuGame*** defines 2 dimensional array (9 X 9) to store input digits and binary tester array used for testing uniqueness check of sudoku. It also defines functions for checking validity of horizontal lines, vertical lines and 3x3 squares. It defines functions for displaying sudoku board and uses function randomInitialize() to generate a solution.

For checking validity of horizontal lines I check for each line if digits in that line are unique. if so then the line is flagged as valid. Similarly vertical lines and 3x3 squares also checked. If all check are passed then the solution is correct.