Open CV color detector

Michael Bagge-Hansen

March 9, 2020

The image is first extracted from the command line as it is passed in from the user, first the program checks to see if the user had added an agrument to the calling of the program, if not it shows a message and returns -1, shown in listing 1.

```
if(argc < 2){
      cout << "Could not open or find the image!\n" << endl;
      cout << "Usage: " << argv[0] << " <Input image>" << endl;
      return -1;
   }
```

Listing 1: argument check

The inputed image is then stored as a matrix using open cv, this image is then croped to only display the center of the image. This is done because in all of the sample images the opject required to detect is in the center. The crop is done using ROI (Region of intrest) calculations. first the box which we want to crop is defined, the top coner of the ROI is defined to be $\frac{1}{4}$ the size of the image from the top corner of the image, the height and width is devined as half the size of the image. A new matrix is created to store the croped image, shown in listing 4.

```
    /* Set Region of Interest */

    cv::Rect roi;
    roi.x = src.size().width /4;
    roi.y =  src.size().height /4 + 20;
    roi.width = src.size().width /2 ;
    roi.height = src.size().height / 2 ;

    /* Crop the original image to the defined ROI */

    Mat crop = src(roi);
```

Listing 2: pixel count example

A function was created to calculate the most common color whos definition is shown below:

```
string mostCommonColor(Mat picture,int sat_tol, int val_tol);
```

Listing 3: most common color definition

This function goes cycles between colors, and calles a function, shown in listing **??** which will return the number of pixles within the range of the speciyed color,as shown in listing 4 . Once all colors have been counted, an algroithim is used to determin which color has the greatest number of pixels, shown in listing 5. the function returns the name of the color with the greates number of pixles which is then displayed to the console.

```
    //blue       100 - 130
    int blue_pix = numPixels(src,101,130,sat_tol,val_tol);

    printf("Blue: %d\n",blue_pix);
    pix[5] = blue_pix;
```

Listing 4: pixel count example

```
    int max_pix = 0, max_index = 0;

    for (int i = 0; i < 7; i++)
    {
        if(pix[i] > max_pix){
            printf("value of %d is bigger than value of %d:\t",i,max_index);
            printf("%d is bigger than %d\n",pix[i],max_pix);
            max_pix = pix[i];
            max_index = i;
        }
    }
```

Listing 5: get max algorithm

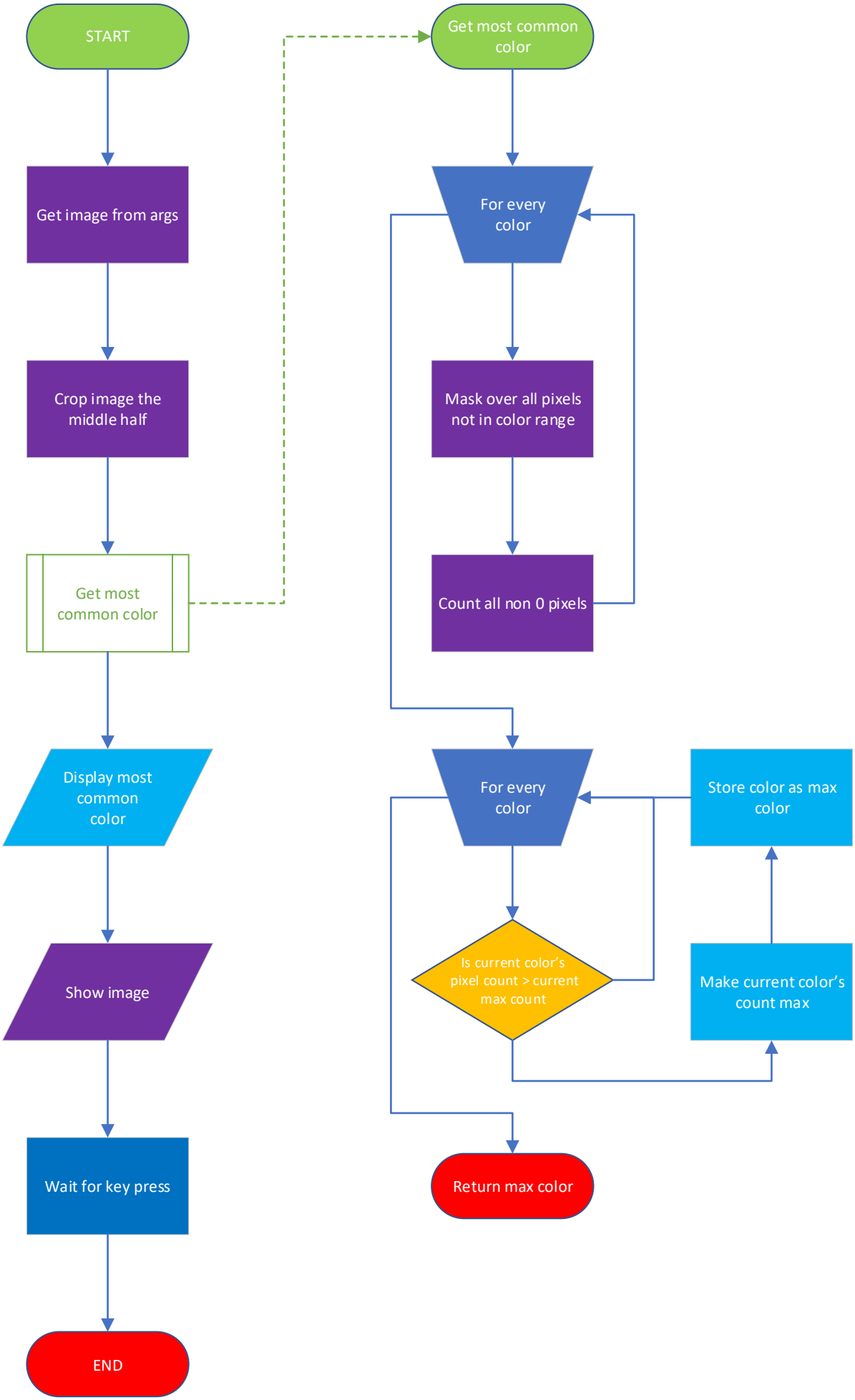
```
    int numPixels(Mat img,int low_hue,int high_hue,int sat_tol,int val_tol){

    Mat frameHSV;        // Convert the frame to HSV and apply the limits
    cvtColor(frame, frameHSV, COLOR_BGR2HSV);
    inRange(frameHSV, Scalar(low_hue, sat_tol, val_tol), Scalar(high_hue, (255-sat_tol), (255-val_tol) ),
     frameHSV);

    int pixels = 0;
    pixels = countNonZero(frameHSV);
    return pixels;
}
```

Listing 6: get number of pixles in range function

listing 6 shows a function to find the number of pixles within a range of hue values. This is done by first converting the image to HSV format, then masking the image not within the range of the hue valuses specifyed using the `inRange` function, a non zero value for saturation and value must be use when speciying the range of colors to mask so black and white pixles are not counted buy the function. Once the image is masked the function `countNonZero` can be used to find the number of pixels.


START
Get image from args
Crop image the middle half
Get most common color
Display most common color
Show image
Wait for key press
END
Get most common color
For every color
Mask over all pixels not in color range
Count all non 0 pixels
For every color
Is current color's pixel count > current max count
Store color as max color
Make current color's count max
Return max color