Door Detection

Assignment 2

CS4053 Computer Vision

Alexandru Sulea
D Stream
#12315152
13 December 2016

Contents

List of Tables

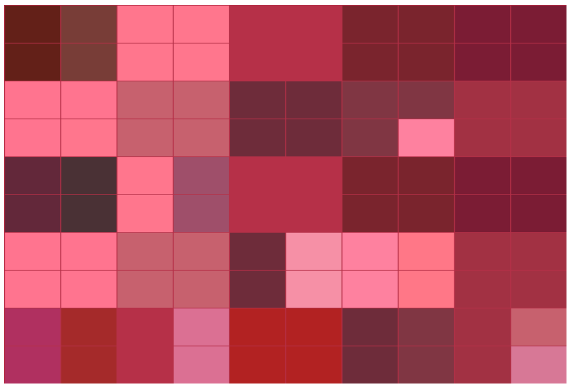
1 RED PIXEL DETECTION

1.1 Intro

In this section the procedures and results for detecting red pixels are outlined. For this part of the assignment the lecture notes, recommended reading and online openCV recources were used to plan an adequate procedure for completing the assignment.

1.2 Procedure

Taking notes of what was discussed and shown in class together with online forms and the opencv page, I chose to plan out my procedure based on a histogram and backprojection detection model. To use the histogram though, red samples were needed to, in essence teach the opencv program what red is, and more specifically what type of red was required to be recognised for the assignment.
The histogram could not be constructed out out of the unaltered sample picture provided due to the program not understanding what it was required to look for.A histogram needs to be instructed what type of data to look for.
A histogram is a collected count of data, the data is not limited to color hue, the data can be anything from intensity values to gradients and directions. The histogram for this tutorial was chosen to only count the hue. This decision was taken due to the objective of the assignment was to find different road signs, all of which had a red contour. Thus the easiest way to find red is to search for all the ranges of the red hue included in road signs.


(a) The second last red sample

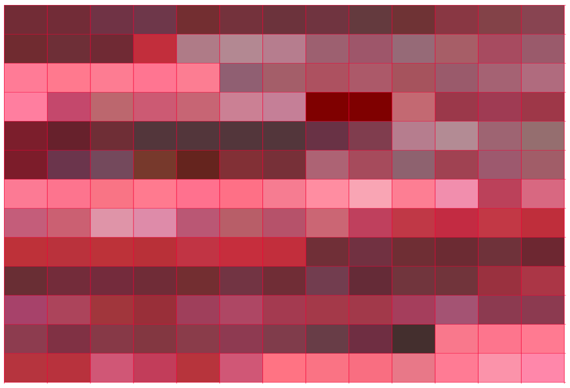

(b) The red sample image used for the current results

Figure 1: Figure of constructed red sample images

Thus, a new sample red image had to be created to make a histogram which would only contain the red present in road signs. To make the red sample image the photoshop program gimp was used to make a simple grid in which different red hues were stored. The red hue samples were taken from the sample road signs image. The images road signs were sampled at different points so as to get an even, accurate representation of the red color used in the road signs.
Once the new sample red image had been created using different points on road signs in the sample road signs image the code could finally be written in.
The first step is to load in the images and check that the correct images have been loaded by the program without any exceptions or errors being thrown. That is where this part of the code comes in.

```
Mat roadsign_test2 = imread("filename.JPG", CV_LOAD_IMAGE_UNCHANGED);
```

```
if (roadsign_sample.empty()){
     printf("Cannot open video file: \n");
     return -1;}
   else { ...}
```

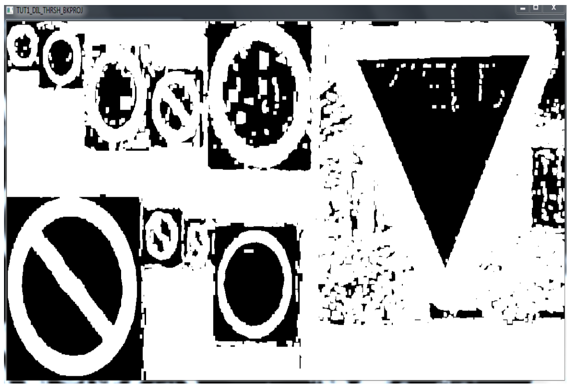
Next the image is converted to HSV or Hue Saturation Value so that we can separate the hue from the rest of the image. Since we are only interested in the color red and its multiple shades we will only be using hue, thus creating a 1D Red Hue Histogram.

```
//hue is from 0 to 180
     float huerange[] = {0, 180};
     //use only hue value
     hue_red.create(hsv_sample_red.size(), hsv_sample_red.depth());
     mixChannels(&hsv_test, 1, &hue_test, 1, ch, 1);
     /*Calculate and Normalize Hist*/
     calcHist(&hue_red, 1, ch, Mat(), hist_red, 1, &histSize_red, &ranges, true, false);
     normalize(hist_red, hist_red, 0, 255, NORM_MINMAX, -1, Mat());
     //drawing out the red histogram
     Mat histImg = Mat::zeros(w, h, CV_8UC3);
     for (int i = 0; i < bins; i++)
     {
        rectangle(histImg, Point(i*bin_w, h), Point((i + 1)*bin_w, h -
             cvRound(hist_red.at<float>(i)*h / 255.0)), Scalar(0, 0, 255), -1);}
     calcBackProject(&hue_test, 1, 0, hist_red, backproj, &ranges, 1, true);
     threshold(backproj, thresh_backproj, 8, 255, CV_THRESH_BINARY);
     dilate(thresh_backproj, dil_thresh_backproj, Mat(), Point(-1, -1), 2, 1, 1);
```

The above code shows how the red samples image was then loaded in, converted to HSV, then split up so that only the hue would be used and finally used to make up the histogram.
Once the histogram had been set up all that remained was to back project the test image with the histogram. However the test image could only be back projected if it also had been converted to a hsv format and then had only its hue used.
Once the two images were successfully back projected together a third image was created. This third image represented where the opencv program found red, thus the back projection. Red being the sample reds which were given to it.
For this project multiple tests were conducted with different and increasingly detailed red sample images due to the inaccuracy of the earliest back projected images. The more red samples were given to the histogram the more accurate the histogram would become. This process has its limitations however, for example there could be red samples present in the test image that were not present in the sample image or vice versa, creating false positives and false negeatives. Thus 100 percent accuracy can never truly be achieved.
Once the image was back projected, it needed to be thresholded to remove any patches the program believed had a low probability of being red, such as orange brick. Dilate was used on the image so as to fill in empty spots in images and try to make them more accurate. Dilate was also used in an effort to close the road sign circles so as they are later recognised as circles by floodfill. This process however had minimal success.
Morphology was also used in an effort to try and close the back projected round road signs. Morphology was better at closing holes and gaps, but for a best result process, dilate was used in conjunction with dilate.

Despite best efforts, the back projected circles could not be closed due to the back projected image not recognising the red hue in some road signs. This did not affect the red samples significantly, but it did have a noticible impact on the black and white pixel detection. Floodfill only works on closed shapes, never opened ones, thus in effect any shape that was not closed would be lost in the process along with accuracy and precision.

To preserve the traffic sign shapes in the image throughout the contour hierarchy process the final red pixel detection process was altered to allow for the black and white pixel detection later on.


(a) The back projected image

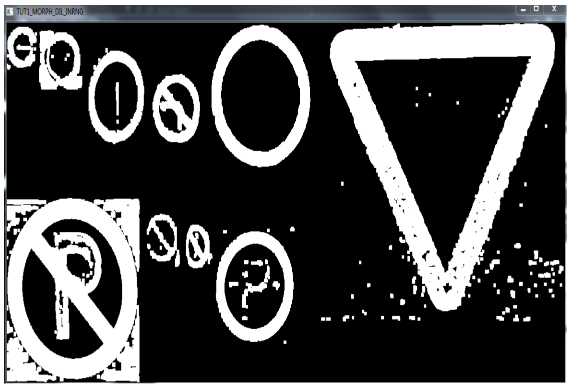

(b) The inRange thresholded image

Figure 2: Figure of constructed tresholded images

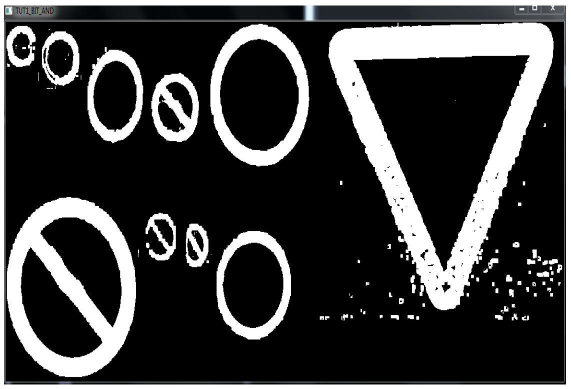
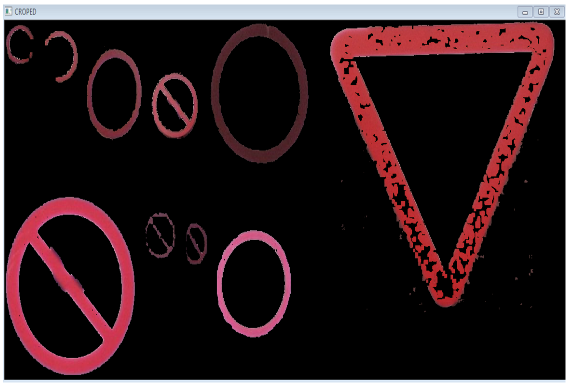

(a) The two images above ANDed give this result

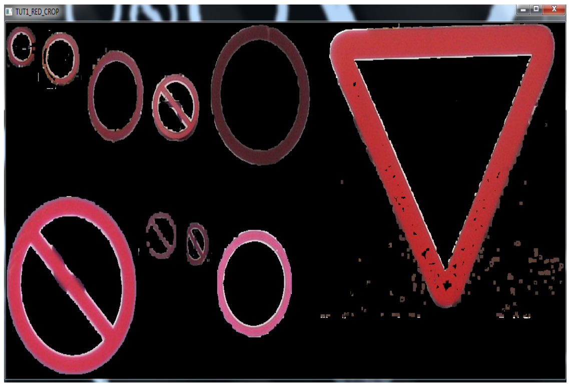
Figure 3: Figure of constructed of the two tresholded images

```
inRange(hls_test, Scalar(10,10,55), Scalar(210, 190, 200), inrange_test);
dilate(inrange_test, dil_inrange_test, Mat(), Point(-1, -1), 2, 1, 1);
// Two images, back proj and inrange merged together
bitwise_and(dil_thresh_backproj, morph_dil_inrange_test, band_rng_bkprj);
dilate(band_rng_bkprj, band_rng_bkprj, Mat(), Point(-1, -1), 1, 1, 1);
/*Dilate and morphology again to close the remaining gaps*/
morphologyEx(band_rng_bkprj, band_rng_bkprj, MORPH_TOPHAT, element, Point(-1, -1), 50);
bitwise_and(roadsign_test,roadsign_test,redcrop, band_rng_bkprj);
```

The code above shows how inRange was used to get a similar but different thresholded image which has fuller contour of the road sign pictures but also a lot of other false positive shapes. The thresholded image was also somewhat inaccurate, but by merging the two pictures together using AND a much clearer and more accurate representation of the red pixels in the image would be achieved. The image was then dilated and closed to help with remaining holes and breaks in the shapes. Finally the resulting image was used as a mask to construct a composite image to show how much of the red pixels from the original image were captured.



(a) A previous, less succesfull composite image using the previous (a) red sample image



(b) The composite image showing the current red mixels detected

Figure 4: Figure of constructed tresholded images

```
CompareRecognitionResults(band_rng_bkprj, ground_red);
```

The above code shows the use of the function CompareRecognitionResults to compare the example thresholded red pixel image with the thresholded ground pixel image to achieve a score for the performance method. The ground image is converted to HLS and then used inRange on it as discussed before. The only difference to this is that the ground image has only one overall value for each color present, making it very easy to threshold out the colors. Thus the following code was used to get the ground thresholded images of the three colors.

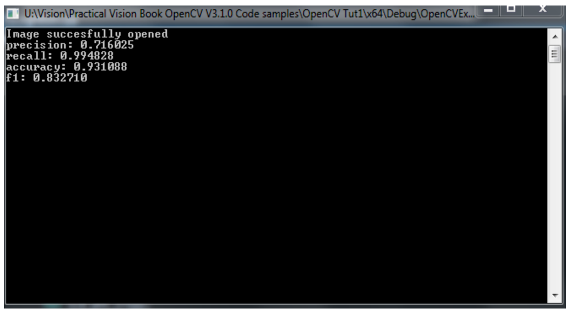
```
inRange(hls_ground, Scalar(0, 0, 255), Scalar(0, 0, 255), ground_white);
inRange(hls_ground, Scalar(0, 255, 255), Scalar(0, 255, 255), ground_red);
inRange(hls_ground, Scalar(0, 0, 0), Scalar(0, 0, 0), ground_black);
```

1.3 Performance

The performance of the thresholded image was tested using the function provided in the practice tutorial. As seen in the picture and table, the precision for the color red is lower than that of black or white pixels. This is due in part to the difficulty in constructing a red sample image for the histogram. The innacuracy is also due to the dilation used on the shapes to connect the circles. The methadology helped in improving black and white pixel detection but at the cost of decreasing red pixel detection precison.
As per the accepted definitions of accuracy and precision. The two are independent of each other. Thus a method can be neither precise nor acurate, precise and not accurate, accurate and not precise or for best results, accurate and precise.
Accuracy defines how close and measurement is to the standard or known value, thus the red sample values to the accepted red values. Precision defines how close a measurement is to another measurement. Thus how close the red sample values were to the ground sample values.
Looking at the score for precision it is clear that this methadology was accurate but not very precise in determining the red pixels.

Table 1: RED PIXEL DETECTION VALUES

TYPE	VALUE
Precision	0.716025
Recall	0.994828
Accuracy	0.931088



U:\Vision\Practical Vision Book OpenCV V3.1.0 Code samples\OpenCV Tut1\x64\Debug\OpenCVEx...

```
Image succesfully opened
precision: 0.716025
recall: 0.994828
accuracy: 0.931088
f1: 0.832710
```

(a) The image shows the scoring achieved for the red pixels

Figure 5: The figure shows the scores for red pixel recognition

1.4 Discussion

One of the earliest problems noticed with hand sampling the red from the road signs was that the quality of the pictures was quite low. The pictures were also taken at an angle and with different cameras, thus the sharpness of the image also varied from picture to picture.

The biggest problem with sampling the sample road signs image was the fact that a lot of the road signs had two, three or even more layers of shadowing on them. This created a problem which was only noticed when the process of constructing the sample image by busket fill in the various squares. I would, to my knowledge pick a red point in a sign only to discover that this point looked oddly brown or purple when transferred to my sample red image that was inserted into the histogram. These brown and purple points also caused a problem further down the road when dark bits of the background were identified as red due to those brownish squares.

The performance for red pixels detection is high although it could have been higher. One major obstacle in a more precise detection algorythm is the quality of the pictures themselfes. Another being the background of the picture. While performing this assignment I have found it quite diffucult to separate a red road sign from its red brick background. Should even more red objects had been present in the picture, a more advanced form of Canny would have to be used to separate the objects and only then perhaps perform back projection.

By observing the threshold images provided for red pixel detection it can be clearly noticed that the picture has false positives, where it recognized parts of the background building as parts of the road sign due to the closeness in hue. It also contains false negatives where, it did not recognize parts of the red borders in the road signs, possibly due to the color being altered by a shadow to a hue outside the target zone. These errors, which were due to the factors previously discussed, posed the biggest challenge to making an accurate and precise road sign detecting opencv program.