# 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

The following lab was based on the procedure of determining when a door is opened and closed in a room given that there is a person walking through the door and that the light in the scene can vary depending on the lighting conditions in the room but also from the light propagating from other sources.
The lab was thus conducted with the objective of detecting a door and identifying when said door closes and opens with respect to time as defined in frames.

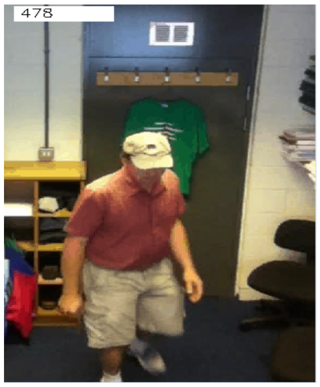
### 1.2 Procedure

Taking notes of what was discussed and shown in class together with the theory and description on the official Open CV page, the following procedure was planned out. The scene itself consists of two distinct parts. The first being a man walking from foreground to background and back, the second being that a door opens and closes at arbitrary intervals. The person in the picture serves no real purpose for this lab other than to induce uncertainty of when the door opens and closes. Thus the first operation is to perform a background subtraction on the person and have them removed as much as possible from the scene. This will not only allow us to see the door more clearly, but it will also help clean up the frames and make Hough Lines more accurate later on.

Once the person is removed from the picture, the door detection must begin.
Here a number of already built in Open CV procedures can be used such as HoughLines() , HoughLineP() or Line Segment Detector. Although LSD is somewhat more accurate then HoughLineP(), the later is also more easily customizable and faster to process. Given how door frame tracking throughout every video frame was not the objective and only opening and closing, HoughLineP() was chosen to help with the door detection. Here whatever lines were detected in the scene by HoughlinesP(), only the lines that were horizontal or vertical, given 10 degrees of error were accepted as the true vertical and horizontal lines.
After the horizontal and vertical lines were filtered through, their determinate is calculated and compared to all the other vertical or horizontal lines as to check if the lines form a box at any point. Should the lines intersect, a circle is drawn at the intersection points.
Then a diagonal line is searched for, its point calculated as well as its determinate. Should the diagonal line intersect one of the points where the horizontal and vertical lines intersect, then this is assumed to top side of the door opening, thus creating an angled line.

The steps for creating the solution were as follows:

The video file is first loaded in to MOG2 which is a background subtraction method, MOG2 was used over MOG due to MOG2 being more adaptable to changing light conditions and also to covering more of the subject with the mask.
A bit$_{w}isenotisthenperformedonthemasksoastoturnitintoaforegroundsubtractionmethod.Thiswayanythingthatmovesinth$



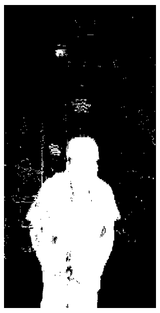
(a) unaltered image

Figure 1: Figure of normal color frame

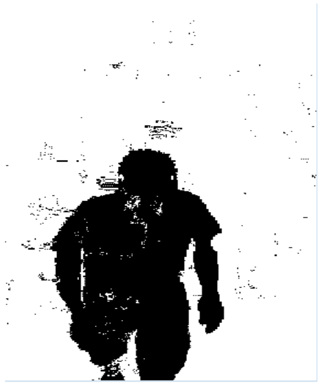
```
        pMOG2->apply(frame_vid_N, fgMaskMOG2, 0.005);
bitwise_not(fgMaskMOG2, fgMaskMOG2);
absdiff(frame_vid_N, fgMaskMOG2, vid_ands);
//imshow("wvvb", vid_ands);
```

Following this procedure, the image was ready to be converted to be able to accept HoughLinesP(). To produce the desired lines, the image was converted to gray scale, so as to be easier to perform adaptive thresholding Adaptive thresholding was chosen due to its adaptability in varying light conditions. A binary picture was then received which needed to be blurred somewhat to regain some of the lost line continuity. The picture was now ready to be inserted into canny for border detection and then converted back to BGR so as to be able to be combined with other images.

```
cvtColor(frame_vid_N, frame_grey, CV_BGR2GRAY);
adaptiveThreshold(frame_grey, frame_g_tresh, 75, ADAPTIVE_THRESH_GAUSSIAN_C, CV_THRESH_BINARY, 13,
    4);
GaussianBlur(frame_g_tresh, frame_blur, Size(5, 5), 1, 1, 1);
Canny(frame_blur, dst, 50, 200, 3);
cvtColor(dst, cdst, COLOR_GRAY2BGR);
```



(a) MOG2



(b) bitwise not MOG2

Figure 2: transition

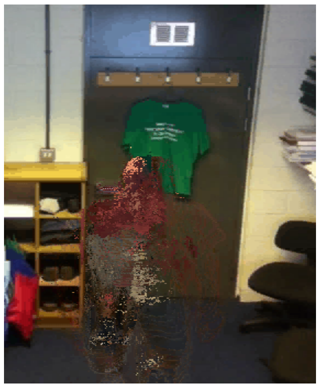
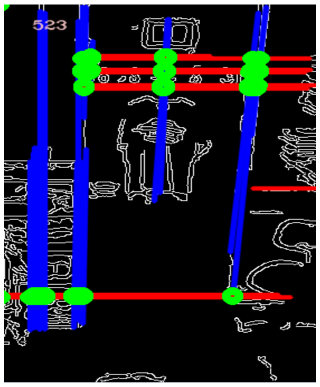
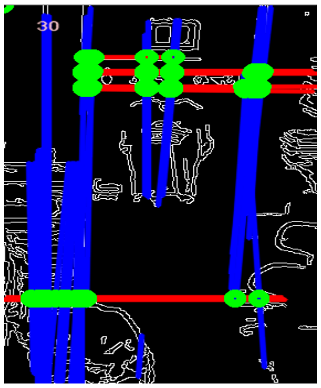

(a) given result of MOG2

Figure 3: foreground is now transparent

```
HoughLinesP(dst, lines4, 1, CV_PI / 180, 80+60, 50, 50);
int vert_line_count = 0;
int hor_line_count = 0;
```



(a) video 2 door houghlinesp



(b) video 1 door houghlinesp

Figure 4: houghlinep

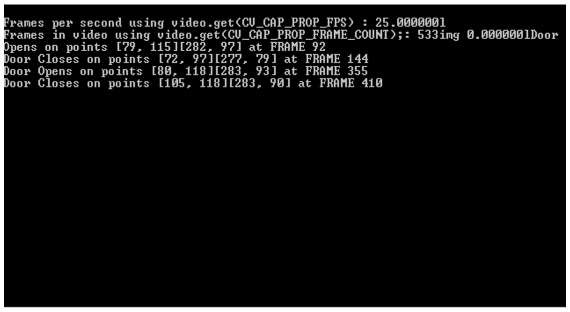
The rest of the code has to do with simple detection of houghlines given their angles. Thus only the horizontal , vertical and in specific cases, diagonal lines are stored. The code below shows how this was achieved. All code was sourced from basic calculus with the exception of the intersection algorithm which was taken from an opencv online tutorial.

```
for (size_t i = 0; i < lines4.size(); i++)
{
Vec4i l = lines4[i];
/*get angle of line*/
double angle = atan2(l[3] - l[1], l[2] - l[0]) * 180 / CV_PI;
/*horizontal lines*/
if (angle <5 && angle >= -5 || angle <-175 && angle >= 175)
{
hor_line_count++;
h_l.push_back(l);
```

```
//cout << "HORIZONTAL LINES " <<Point(l[0],l[1])<<Point(l[2],l[3]) <<endl;
line(cdst, Point(l[0], l[1]), Point(l[2], l[3]), Scalar(0, 0, 255), 3, CV_AA);
}
/*vertical lines */
if (angle< 95 && angle >= 85 || angle < -85 && angle >= -95)
{
vert_line_count++;
v_l.push_back(l);
//cout << "VERTICAL LINES " << Point(l[0], l[1]) << Point(l[2], l[3]) << endl;
line(cdst, Point(l[0], l[1]), Point(l[2], l[3]), Scalar(255, 0, 0), 3, CV_AA);
}
/*if door not closed*/
if (angle <-5 && angle >= -25 || angle <175 && angle >= 150 || angle >-175 && angle <= -150)
{
if (frame_thresh > frame_count - last_frame) {
open_close_count++;
oc_l.push_back(l);
//cout << "DIAGONAL LINES " << Point(l[0], l[1]) << Point(l[2], l[3]) << "FRAME " << frame_count
    << endl;
}
else {
last_frame = frame_count;
diag_line_count++;
diag_l.push_back(l);
line(cdst, Point(l[0], l[1]), Point(l[2], l[3]), Scalar(0, 200,200), 3, CV_AA);
if (diag_line_count % 2 == 0) {
cout << "Door Closes on points " << Point(l[0], l[1]) << Point(l[2], l[3]) << " at FRAME " <<
    frame_count << endl;
}
else {
cout << "Door Opens on points " << Point(l[0], l[1]) << Point(l[2], l[3]) << " at FRAME " <<
    frame_count << endl;
}
}}}
/*Draw circle at line intersection*/
for (int i = 0; i < h_l.size(); i++)
{
for (int j = 0; j < v_l.size(); j++)
{
Point poi = intersection(Point(h_l[i][0], h_l[i][1]), Point(h_l[i][2], h_l[i][3]),
    Point(v_l[j][0], v_l[j][1]), Point(v_l[j][2], v_l[j][3]));
circle(cdst, poi, 6, Scalar(0, 255, 0), 5, 8,0);
}}
for (int i = 0; i < h_l.size(); i++)
{
for (int j = 0; j < oc_l.size(); j++)
{
Point poi = intersection(Point(h_l[i][0], h_l[i][1]), Point(h_l[i][2], h_l[i][3]),
    Point(oc_l[j][0], oc_l[j][1]), Point(oc_l[j][2], oc_l[j][3]));
if ( (Point(h_l[i][0], h_l[i][1]) == Point(oc_l[j][0], oc_l[j][1])) || (Point(h_l[i][2],
    h_l[i][3]) == Point(oc_l[j][2], oc_l[j][3])) || (Point(h_l[i][2], h_l[i][3]) ==
    Point(oc_l[j][0], oc_l[j][1])) || (Point(h_l[i][0], h_l[i][1]) == Point(oc_l[j][2],
    oc_l[j][3])) )
{
line(cdst, Point(oc_l[i][0], oc_l[i][1]), Point(oc_l[i][2], oc_l[j][3]), Scalar(0, 200, 200), 3,
    CV_AA);
circle(cdst, poi, 6, Scalar(200, 0, 200), 5, 8, 0);
}}}
```

## 1.3 Performance

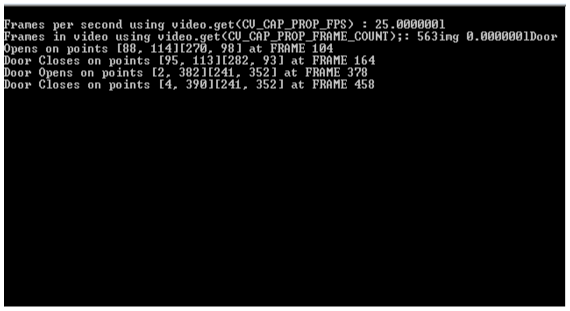
Given as how the ground truth was provided or this lab in terms of door frame position and opening and closing frames, it was quite simple to compare the achieved results to the ground truth.



```
Frames per second using video.get(CV_CAP_PROP_FPS) : 25.0000001
Frames in video using video.get(CV_CAP_PROP_FRAME_COUNT);: 533img 0.0000001Door
Opens on points [79, 115][282, 97] at FRAME 92
Door Closes on points [72, 97][277, 79] at FRAME 144
Door Opens on points [80, 118][283, 93] at FRAME 355
Door Closes on points [105, 118][283, 90] at FRAME 410
```

(a)

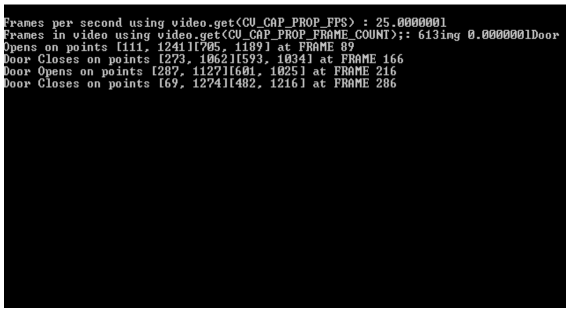
Figure 5: Tdoor1 score



```
Frames per second using video.get(CV_CAP_PROP_FPS) : 25.0000001
Frames in video using video.get(CV_CAP_PROP_FRAME_COUNT);: 563img 0.0000001Door
Opens on points [88, 114][270, 98] at FRAME 104
Door Closes on points [95, 113][282, 93] at FRAME 164
Door Opens on points [2, 382][241, 352] at FRAME 378
Door Closes on points [4, 390][241, 352] at FRAME 458
```

(a)

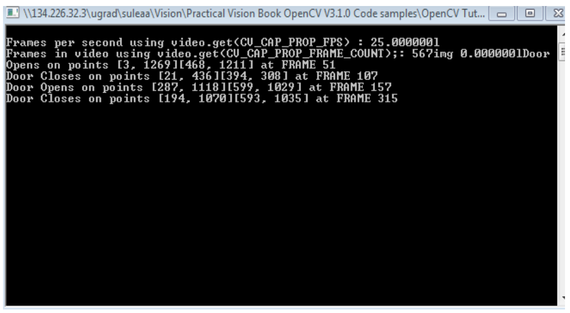
Figure 6: Tdoor2 score



```
Frames per second using video.get(CV_CAP_PROP_FPS) : 25.0000001
Frames in video using video.get(CV_CAP_PROP_FRAME_COUNT);: 613img 0.0000001Door
Opens on points [111, 1241][705, 1189] at FRAME 89
Door Closes on points [273, 1062][593, 1034] at FRAME 166
Door Opens on points [287, 1127][601, 1025] at FRAME 216
Door Closes on points [69, 1274][482, 1216] at FRAME 286
```

(a)

Figure 7: Tdoor1 score



\\134.226.32.3\ugrad\suleaa\Vision\Practical Vision Book OpenCV V3.1.0 Code samples\OpenCV Tut...

```
Frames per second using video.get(CV_CAP_PROP_FPS) : 25.0000001
Frames in video using video.get(CV_CAP_PROP_FRAME_COUNT);: 567img 0.0000001Door
Opens on points [3, 1269][468, 1211] at FRAME 51
Door Closes on points [21, 436][394, 308] at FRAME 107
Door Opens on points [287, 1118][599, 1029] at FRAME 157
Door Closes on points [194, 1070][593, 1035] at FRAME 315
```

(a)

Figure 8: Tdoor2 score

## 1.4 Discussion

While solving the problems inherent in detecting a door, a number of problems arose. The main being that, even though the solution was designed to fit a wide array of cases, it will fail in many more due to a variety of problems such as:

1. The precision and accuracy of the solution is heavily dependent of the lighting conditions present and also the color difference of the door as opposed to its background.
   While doing the lab, algorithms continually had problems detecting where the door ended and the door frame ended, due to the fact that the two were very flush and of the same design and color.

2. The door shape itself would need to have 4 corners to provide an accurate door representation, any lift doors or star trek/ star wars type doors would not be detected.

3. The next group of problems arise once the video is recorded of a viable door opening and closing Assuming that the door conforms to the standard door dimensions and that the video angle is not skewed then the question of lighting comes into effect. Too much or little light, such as when a powerful lamp is turned on or when the lights are off can seriously effect the accuracy of the algorithm. One option to offset this issue is to use infra red enabled cameras so as to get a more light independent picture.

4. Even if a door is detected, the continual detection of the door will depend on a clear line of sight from the camera to the door and also a preferably higher definition image for improved accuracy. People or objects existing between the camera and door line of sight can seriously hamper the accuracy. Thus to mitigate this effect, a back projection algorithm is generally advised.

The problems outlined above are just the general outline. There are many more problems which may be case specific. In manual doors for example, a door opening or closing at an uneven rate can also decrease the accuracy and precision of the door detection.
Lastly there are the accuracy errors which were due to false positives and false negatives in the video.
The false positives were mostly due to noise in the original video, which through thresholding and canny promulgated in the HoughLines video. This was easy to fix by adjusting the thresholding of the video to a value at which there was a minimum amount of error but still enough detail for lines to be drawn.
Light reflectance was found to be a major cause of the earliest false positives. As the door is not mat but glossy. This property combined with light shining directly on the door cause the door to somewhat reflect the objects in front of it but behind the camera. Giving the impression that the door may be moving, when in fact it is static.

The false negatives were somewhat harder to solve as important details in the video may be lost through canny. Thus a line that existed and was clearly defined in the color, unaltered version of the video may have dissipated. This can be somewhat corrected by extending or merging lines in the HoughLinesP() function, although this does not work for all cases.

Lastly, the solution presented does not solve for all cases. One example may be that if a door is continuously rocked back and forth. Such as in a busy hallway where someone may hold the door for many other people. The algorithm will see the back and forth movement of the door as opening and closing. As there is no state of a door being opened. Only that a door is about to be opened or about to eb closed.