Door Detection

Assignment 2

CS4053 Computer Vision

Alexandru Sulea
D Stream
#12315152
13 December 2016

Contents

List of Tables

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.

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 bitwise not is then performed on the mask so as to turn it into a foreground subtraction method. This way anything that moves in the foreground is made invisible while the background is largely unaffected The learning rate for the method was chosen at 0.005 . This was found to have the perfect balance between making the subject invisible, while at the same time keeping the moving door as part of the background.

(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

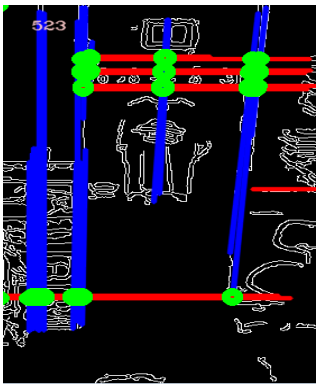
(b) bitwise not MOG2

Figure 2: Masks created from background segregation

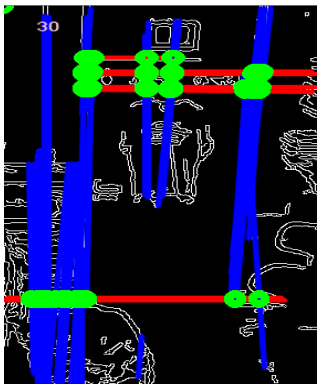
(a) given result of MOG2

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 code below performs a simple detection of houghlinesP 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;
```

Code above runs a loop for every line detected in the frame. with the help of houghlineP detection we know that the number of lines detected is lines4.size(). The angle is then calculated through basic calculus and then converted to degrees as atan2() gives an answer in radians.

```
/*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);
}
```

The code above filters all the lines detected in the frame and only gives the points of the lines which are horizontal or vertical ± 5 degrees. This is done in an effort to find the door frames. Since most standard doors have exclusively horizontal and vertical edges, it is then safe to assume that searching for those edges will identify a door, among other objects.

```
/*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;
}
```

The above code filters and counts the diagonal lines. These lines will help identify our door as a door. Even tho most objects in the world are square in shape, very few can make a diagonal line with what was once a horizontal one. This phenomenon is exclusive to a plane rotating either to or from the camera, but the edge rotating away from view and the edge rotating towards the view will create a diagonal line. Thus providing a means to identify doors.

```
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);
}}
```

Once the horizontal, diagonal and vertical lines have been detected the filter in the code above will find the points of intersection from the vertical and horizontal and draw them.

```
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);
}}}
```

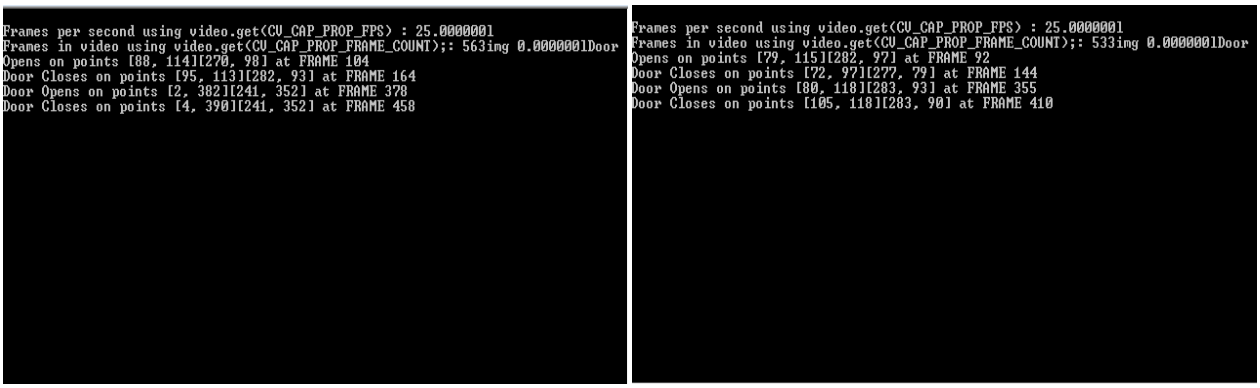
Those intersecting points are then compared with any diagonal line points that may intersect them. This will lead to the horizontal line turning into the diagonal line as discussed above. Once these lines appear, they are stored and counted. Due to watching the video we know that there are 2 sets of door opening and closing. Furthermore we know that a door cant possibly open and close within 50 frames. Thus a count is implemented to accurately detect the diagonal line when it appears but also to filter out any line repeats that may occur due to thresholding induced defects.

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.

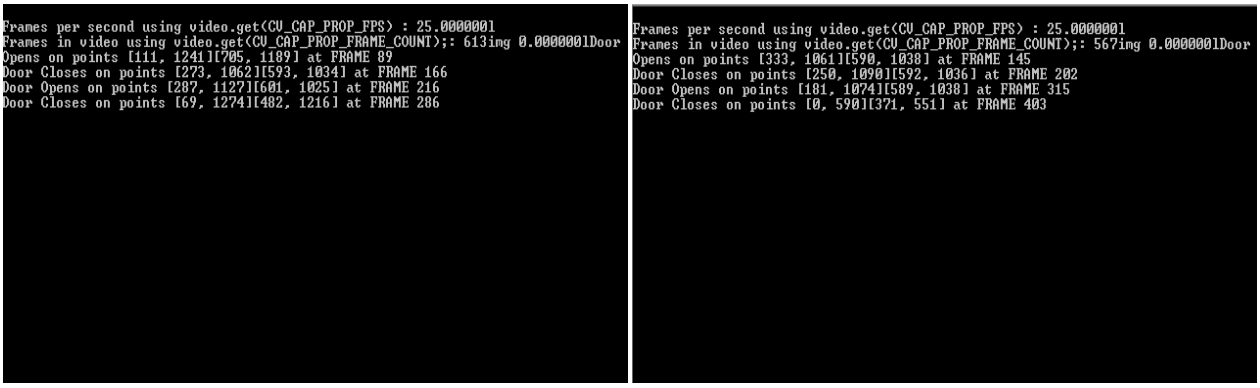
As can be seen from the scores below, the algorithm performs quite well in detecting when the door opens and closes. The algorithm begins to loose accuracy in the variable light image and also in the larger pictures.
This problem is two fold and its causes are also independent of each other. The inaccuracy from the variable light is due to the light distorting the binary outputs of the MOG2 and also the adaptive thresholding. While it is advertised that adaptive thresholding is adaptive in variable light, this has its limits and is not a complete solution. Too much thresholding can lead to loosing important features, while too little thresholding can lead to unwanted error inducing details.

The larger video is less accurate due to it having a more clear picture of the objects in it. These objects are harder to threshold out during the process and thus will induce errors when processing the video.
The higher the video quality, the higher the pixel density , thus while higher definition videos are generally more preferred, these videos take far longer to process or demand faster and more expensive hardware.



(a) door video 1 score (b) door video 2 score

Figure 5: Scores for door video 3 and 4



(a) door video 3 score (b) door video 4 score

Figure 6: Scores for door video 3 and 4

Table 1: Recorded Results

Video	1st Door Open Frame	1st Door Close Frame	2nd Door Open Frame	2nd Door Close Frame
1	104	164	378	458
2	92	144	355	410
3	89	166	216	286
4	51	107	157	315

Table 2: Ground truth

First open close	Second open close
94-211	373-490
71-184	339-460
153-251	380-492
134-230	311-432

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 be closed.