Effects Of Shadow On Canny Edge Detection through a camera

Srajit Mehrotra

Abstract: Shadow causes errors in computer vision as it is difficult to detect objects that are under the influence of shadows. Shadow hides the true properties of objects and instead shows false edges and features. Moreover, shadow causes artificial colour nature and shape deformation of objects, which degrades the quality of image and acts as a hurdle in computer vision and object recognition algorithms. In path detection, strong shadows on the path troubles the detection of the boundary between clear path and obstacles, making clear path detection algorithms less robust. Shadows affect many object and edge detection algorithms. This paper focuses on eliminating the errors due to shadow and varying light intensity in Canny edge detection algorithm.

Introduction

Canny Edge Detection is a popular edge detection algorithm developed by John F. Canny in 1986. Canny also produced a computational theory of edge detection explaining why the technique works. A lot of people consider the Canny Edge Detector the ultimate edge detector to get clean, thin edges that are well connected to nearby edges. The canny edge detector is a multistage edge detection algorithm. The steps are:

1. Preprocessing

   First step is to remove the noise in the image with a 5x5 Gaussian filter.

2. Calculating gradients

   The magnitude of gradient is $m = \sqrt{G_x^2 + G_y^2}$. The direction of gradient is $\theta = arctan(\frac{G_y}{G_x})$. Here, G_x and G_y are the X and Y derivatives at the point being considered.

3. Non-maximum suppression

   A full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighbourhood in the direction of gradient.

4. Hysteresis thresholding

   The values of upper and lower threshold are set to filter out the noise remaining after applying Gaussian filter.

Shadows and varying intensities of light within an area cause changes in gradients and threshold and increases noise. It also effects non-maximum suppression and are difficult to remove as are of nearly the same gradient and threshold as that of the edge [1][2][3].

Materials and Methods

Live input of the image was taken from Logitech C270 HD Webcam to RoboRealm and OpenCV. Canny edge detection is used to determine the path for a robot. Then Canny edge detection was applied by:

1. Smoothing the image to remove noise.

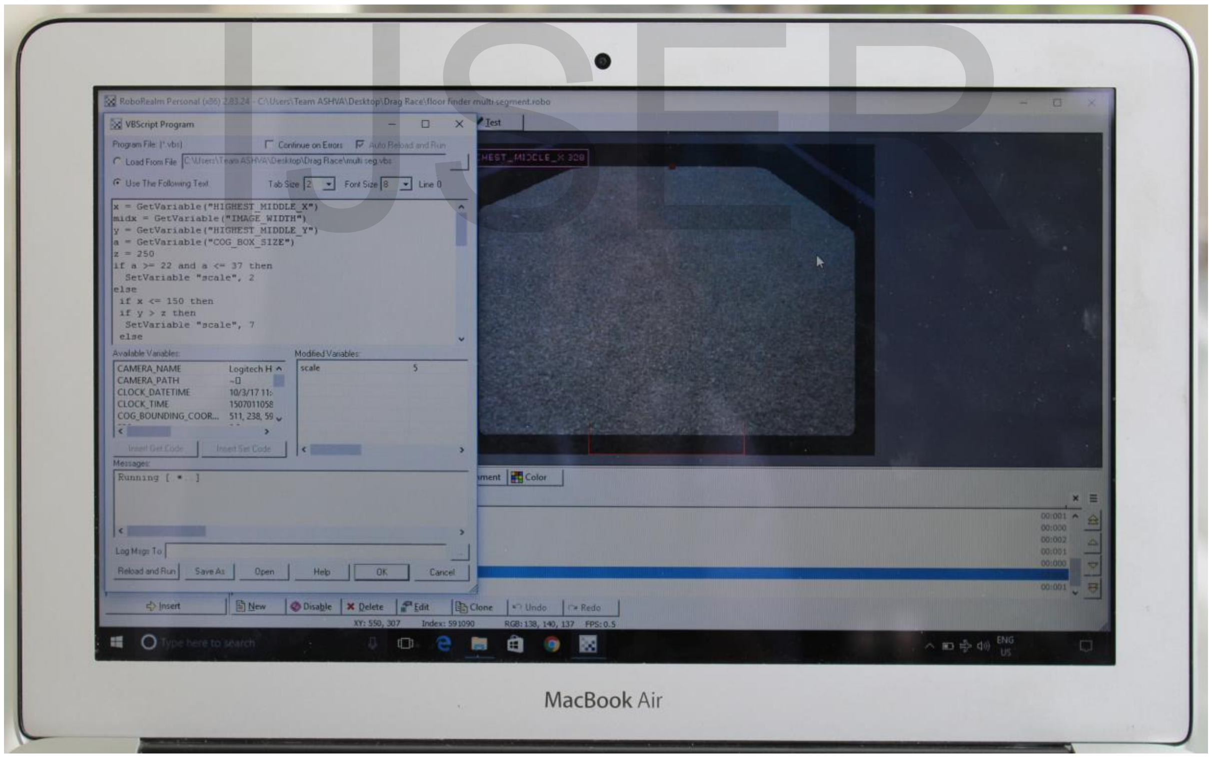
2. Finding gradients where the edges should be marked

3. Non-maximum suppression

4. Determining Potential edges through Double Thresholding and

5. Edge tracking by hysteresis (supressing the edges that are not certain)

The screen is horizontally divided into five and outputs are given based on the direction of the path.



```
x = GetVariable("HIGHEST_MIDDLE_X")
midx = GetVariable("IMAGE_WIDTH")
y = GetVariable("HIGHEST_MIDDLE_Y")
a = GetVariable("COG_BOX_SIZE")
z = 250
if a >= 22 and a <= 37 then
  SetVariable "scale", 2
else
 if x <= 150 then
 if y > z then
  SetVariable "scale", 7
 else
```

Modified Variables: scale 5

Messages: Running [• .]

Fig 1: Image is smoothened and divided horizontally into segments on the system which runs the algorithm.

Results

When two or more edges are close to each other, at times, canny edge detection is not able to determine a particular edge and catches different edges. The focus keeps on shuttling between the edges, resulting in the programme to give wrong values (and garbage values). When a detector analysis an image, it should determine all the edges in a particular threshold range and so you should get all the edges irrespective of their separation. Strangely, no. The Gaussian smoothing blurs out the image, making the edges, and corners and junctions harder to detect. Different light intensities on different edges leads to different values of threshold and intensity along the same edge, that results in the programme being confused between different edges.

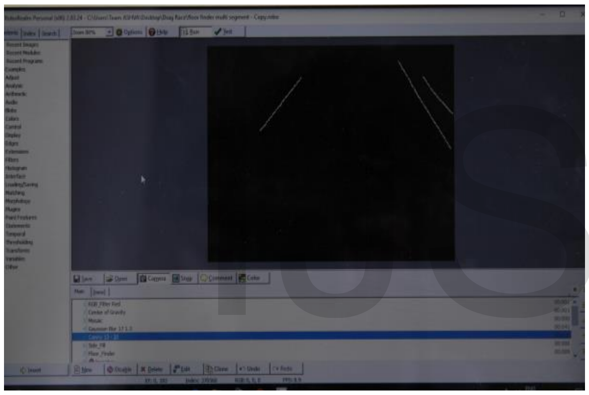
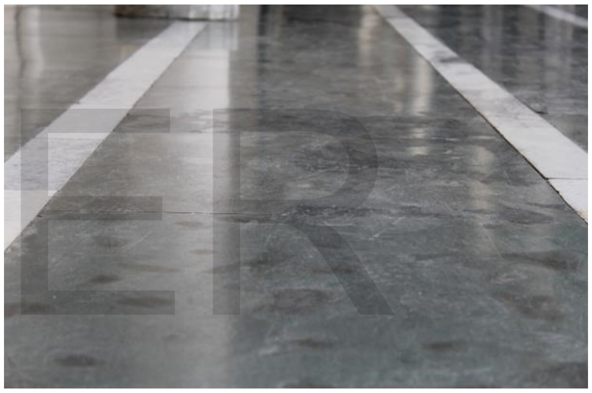
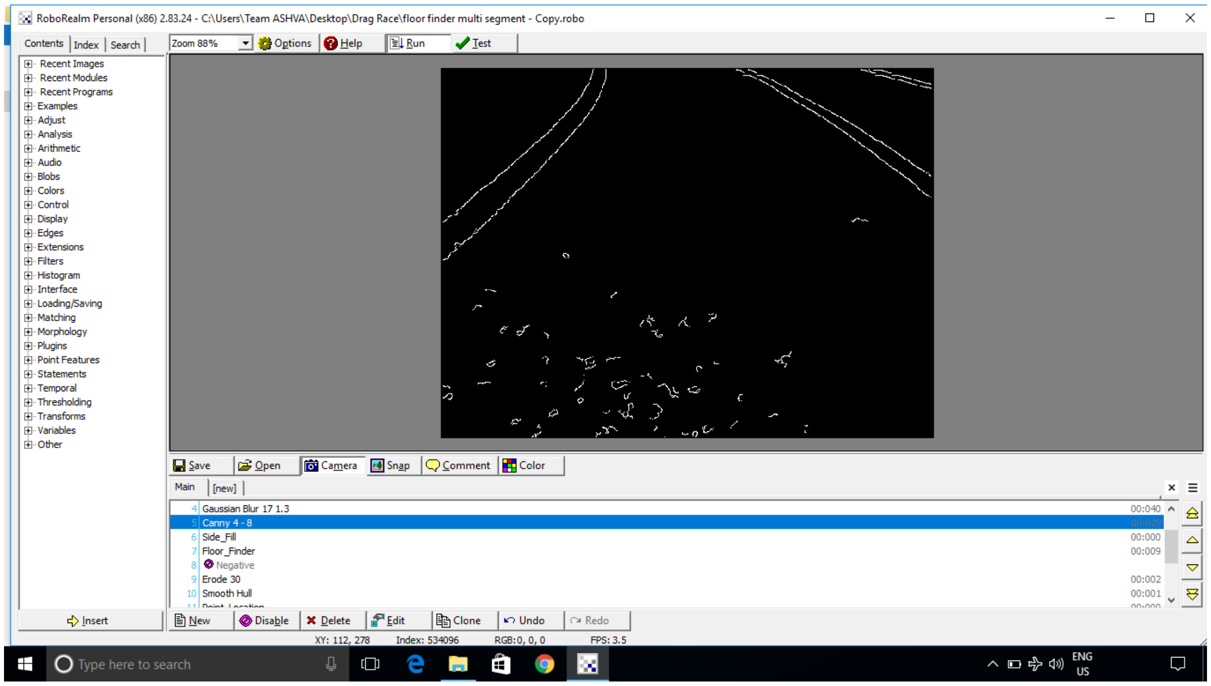



Fig 2: Left – Canny Edge detector is not able to capture a proper edge.
Right – The image that is analysed by the detector.

On rough surfaces, canny edge detection detects many edges. According to theory, these edges can effectively be filtered by adjusting threshold and Gaussian blur or through hysteresis. But in practice, it is not so. Many of the edges of the surface still show up. The difference in intensity of lights causes the threshold values and focus to change. At the places of low light (or shadow), the contrast of the image increases, that results in more edges to show up. On bright spots, due to more scattering of light and Gaussian smoothing, the location of the edges might be off.


RoboRealm Personal (x86) 2.83.24 - C:\Users\Team ASHVA\Desktop\Drag Race\floor finder multi segment - Copy.robo

```
 4 Gaussian Blur 17 1.3   00:040
 5 Canny 4 - 8            00:029
 6 Side_Fill              00:000
 7 Floor_Finder           00:009
 8 Negative
 9 Erode 30               00:002
10 Smooth Hull            00:001
```

XY: 112, 278 Index: 534096 RGB:0, 0, 0 FPS: 3.5

Fig 3: Noise on rough surfaces that is difficult to be removed due to their shadow.

When processing images in an area of varying light intensity, often canny edge detection fails. It is not able to detect edges where there is low intensity. Further when there is a large difference in intensity between two adjacent pixels, the algorithm considers it as an edge. If the shadow of any object falls on an edge or on coarse objects, the algorithm is not able to determine the actual edges and corners. Moving shadows causes large errors in object localization and recognition. As a result, the actual properties of the edges are not shown and the algorithm is either not able to determine a single value or gives the wrong output.

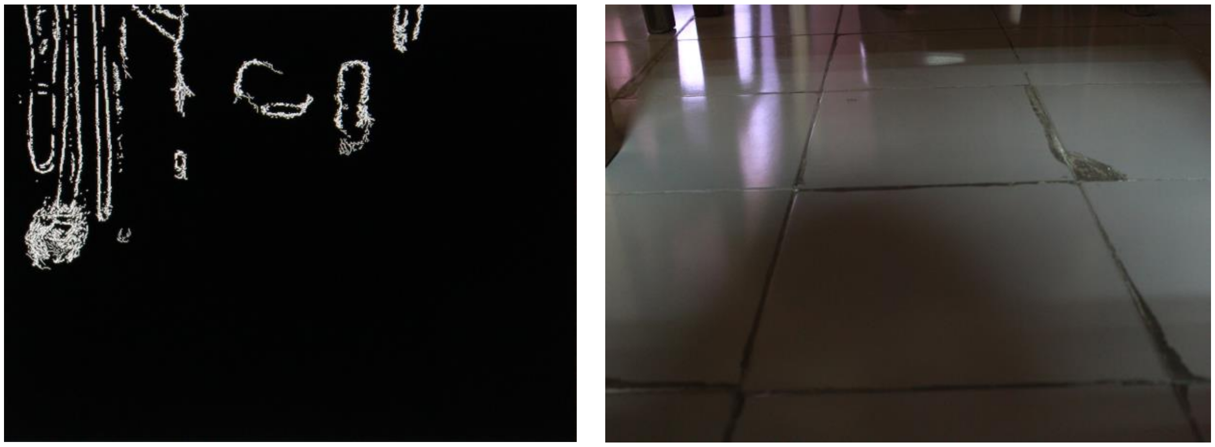
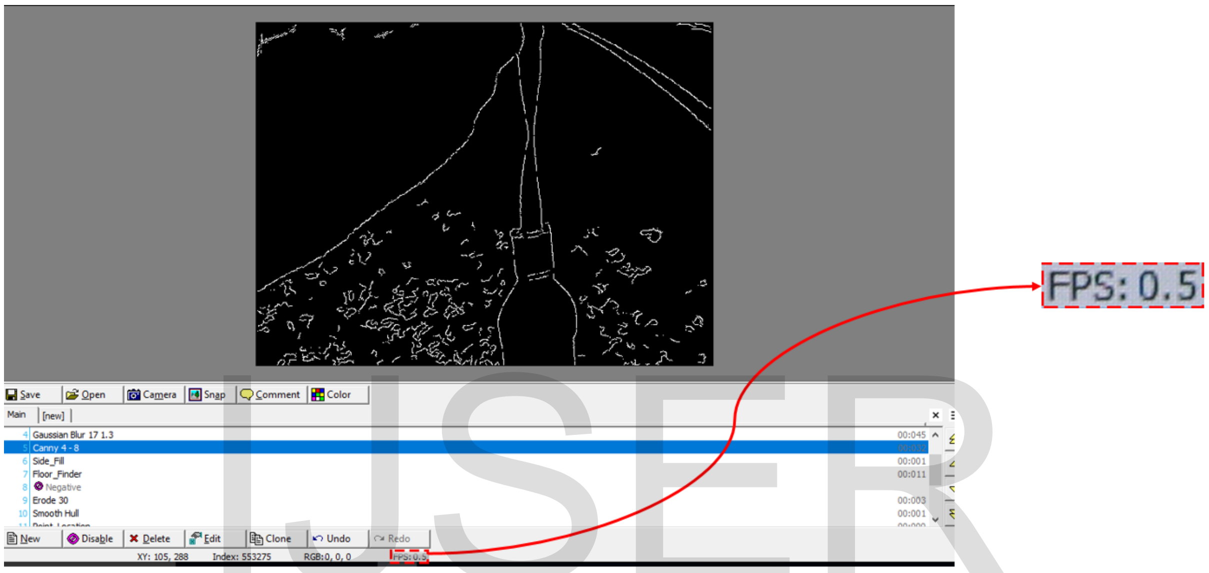


Fig 4: Left – Canny edge detector is not able to detect edges due to low light intensity.
Right – The image that is analysed by the detector

A drawback of Canny edge detector is that it contains a lot of steps, which are much more complex than steps in some other edge detectors. In Canny's original paper, the derivation of the optimal filter led to a Finite Impulse Response filter, which can be slow to compute in the spatial domain if the amount of smoothing required is important. Large range of double threshold values and too much noise in an image often results in FPS lower than normal. Environmental factors such as low light and shadows also cause FPS to drop as it requires more processing to remove shadows and prevent false detection. For example, in more complicated images, FPS drops down and the processing of the system increases.



```
 4 Gaussian Blur 17 1.3        00:045
 5 Canny 4 - 8                 00:032
 6 Side_Fill                   00:001
 7 Floor_Finder                00:011
 8 Negative
 9 Erode 30                    00:003
10 Smooth Hull                 00:001

XY: 105, 288    Index: 553275    RGB:0, 0, 0    FPS: 0.5
```

Fig 5: Low FPS in complicated images.

Discussion

The detector catches different edges when the image is not in focus and there in a variation in the intensity of light. Changing camera focus such that the true edge comes in focus, while the noise is still smoothened solves the problem as long as there is not too much noise. On rough surfaces, there is a lot of noise. Many edges are captured as each edge has its own shadow [4], which is not filtered by hysteresis and double threshold. This can be prevented by switching to greyscale. In greyscale, the colour of the whole path changes into shades of grey and there is not much difference between the shadow [5] of the edge and the rest of the path. Here, adjusting the threshold values removes noise from the image. Sometimes, altering the contrast of the image might also be required as, due to greyscale few edges, with low light intensity, might also be filtered out by double threshold.

Varying intensity of light can be countered by changing the intensity of the image [6]. When intensity of light is low, edges can effectively be seen by using the ‘shift to white’ filter of intensity. On the other hand, when intensity of light is high, contrast can be increased by ‘shift to black’ filter of intensity. Another way of achieving the same result and countering false detection due to low light conditions is to place a dark film in front of the camera. The film should not be very dark, but such that the shadows of edges is not seen through it. The dark film would hence hide the shadows and noise and lead to more accurate results.

Canny edge detection works slow due to high complexity of its functions. In more complicated images, the number of filters required is higher than normal. This results in lower FPS. Further, the calculation and mapping the raw data into results increases the load on the parent system. Using optimum number of filters would lead to better results and lower FPS drop in more complicated images. Changing the angle of camera during path detection through canny edge detection would decrease focus and increase smoothening, in other words simplifies the image up to an extent. This results in higher speed of detection.

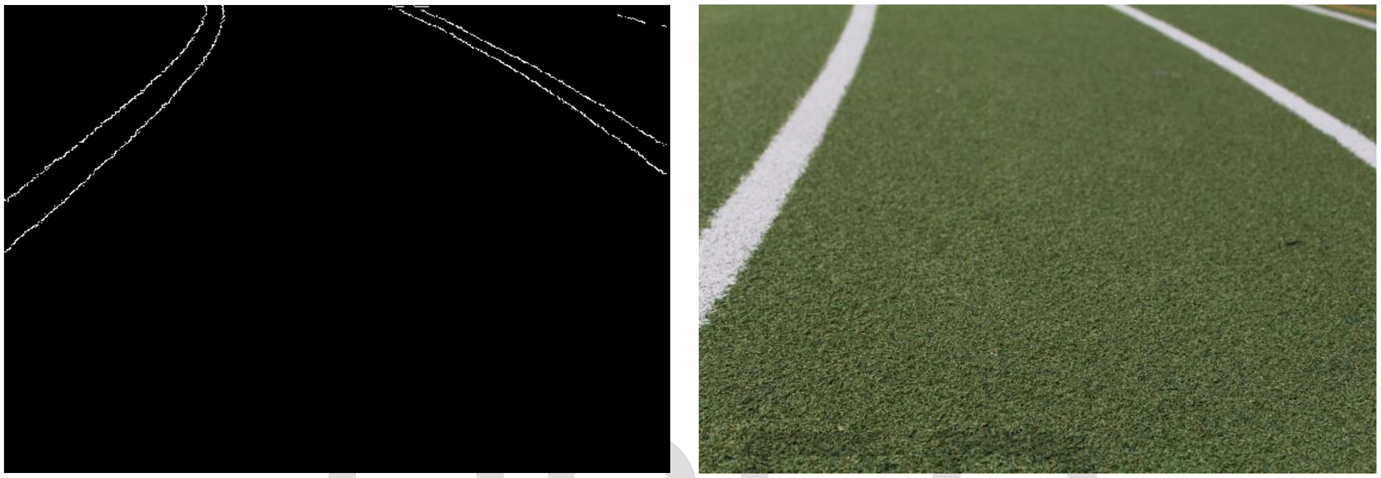


Fig 6: Left – Edges detected through Canny edge detection algorithm after making required changes. Right – The input image for Canny edge detection algorithm.

Conclusion

In this paper, first the effects of varying intensity of light within the image and the effects of shadow on edge detection in discussed. Experiments are performed based on theory and the problems that aroused are stated. At last, various methods are proposed to counter those problems and achieve better accuracy.

References

[1] Canny Edge Detection on Open Source Computer Vision (OpenCV) -- http://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html
[2] Indian Institute of Technology Delhi, “Canny Edge Detection” (Canny 09gr820), March 2009.
[3] John Canny, “A computational approach to edge detection. Pattern Analysis and Machine Intelligence,” IEEE Transactions on, PAMI-8(6):679–698, Nov 1986.
[4] Mei Xiao, Chong-Zhao Han and Lei Zhang., “Moving shadow detection and removal for traffic sequences,” International Journal of Automation and Computing, March 2006.
[5] Sharma, Prateek & Sharma, Reecha., “Shadow Detection and its Removal in Images: A Review,” An International Journal of Engineering Sciences, Vol. 17, January 2016.
[6] Arbel E., Hel-Or H., "Shadow Removal Using Intensity Surfaces and Texture Anchor Points," IEEE Transactions on Pattern Analysis and Machine Intelligence,vol.33, Issue 6, pp 1202-1216, June 2011.