ARTIFICIAL INTELLIGENCE INT-401

PROJECT NAME:

EDGE DETECTION

[IMAGE PROCESSING]

Submitted by: Manish Kumar Singh

R.g:11814398

Jaspreet Singh

R.g:11802033

Section: K18GA

GitHub:

Submit to: Pro. Shabnam.

(Assistant Professor)

School Of Computer Science

LOVELY PROFESSIONAL UNIVERSITY

Transforming Education Transforming India

Introduction:

1. Overview of Project
2. Introduction to Project.

Contribution:

1. Member in This Project.
2. Their Contribution.

Implementation:

1. Concept of Implementation
2. Algorithm and Code
3. Description of Program.

Conclusion:

1. How Project helpful in Real Life.
2. What We Learnt From This Project.

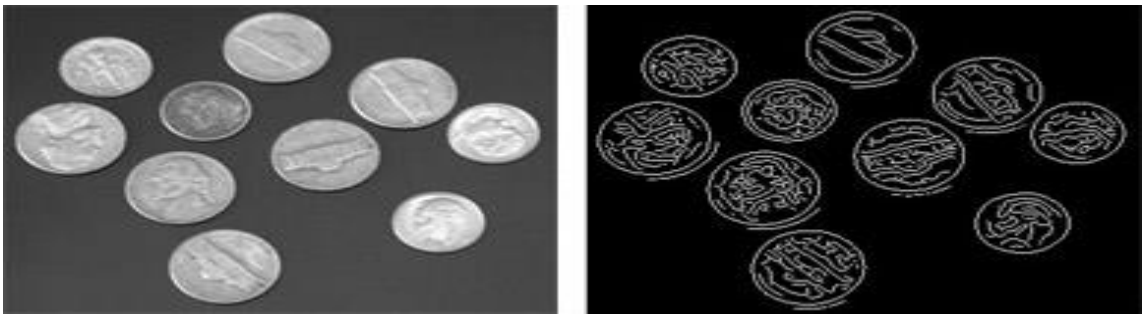
INTRODUCTION

Overview:

This project is part of image processing used in self driving car, Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too. There are two types of methods used for image processing namely, analogue and digital image processing. Analogue image processing can be used for the hard copies like printouts and photographs. Digital image processing techniques help in manipulation of the digital images by using computers. The three general phases that all types of data have to undergo while using digital technique are pre-processing, enhancement, and display, information extraction.

Introduction to Project:

Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness. Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision.



The first image is Gray Scale image of original image and second is processed image of it.

Contribution

We are two person, Manish Singh and Jaspreet Singh. We both researched, analysed, implemented and tested. It took around two month to do research, do

analysis, implementation and testing. As we are very passionate to learn Machine learning, after allocation of project we start research about Edge Detection of Image. And we both contribute equal effort to accomplish this project. And we are thanks to each other for giving such great time to work together.

Implementation

Concept:

Edge detection is one of the most basic tasks in image processing and analysis. The edges of an image consist of the location and outline of the image. These edges are basic features of the image and are widely used in feature description, image segmentation, image enhancement, image restoration, pattern recognition, and image compression. Developing methods for detecting and extracting the edges and outline features of an image has always been a hot subject of research in the field of image processing and analysis, with new theories and methods constantly emerging. Because of the influence of various external factors, during the process of image retrieval and transmission, images are subject to disturbance of noise whose frequency is similar to that of the edge points. As a result, image edges thus extracted suffer from various problems, including false detection, false positives, false negatives, and edges not being of the single image original width. Therefore, edge detection technologies for noise in image are receiving increased attention. The traditional Sobel and Canny filters, while having extensive applications, only consider locally large gradient, especially large gradient in colour and brightness.

Algorithm:

There many technique and algorithm to detect edge of an image and extract information from the image. One of the popular technique is canny.

Canny edge detector: Canny edge detector have advanced algorithm derived from the previous work of Marr and Hildreth. It is an optimal edge detection technique as provide good detection, clear response and good localization. It is widely used in current image processing techniques with further improvements.

Canny edge detection algorithm:

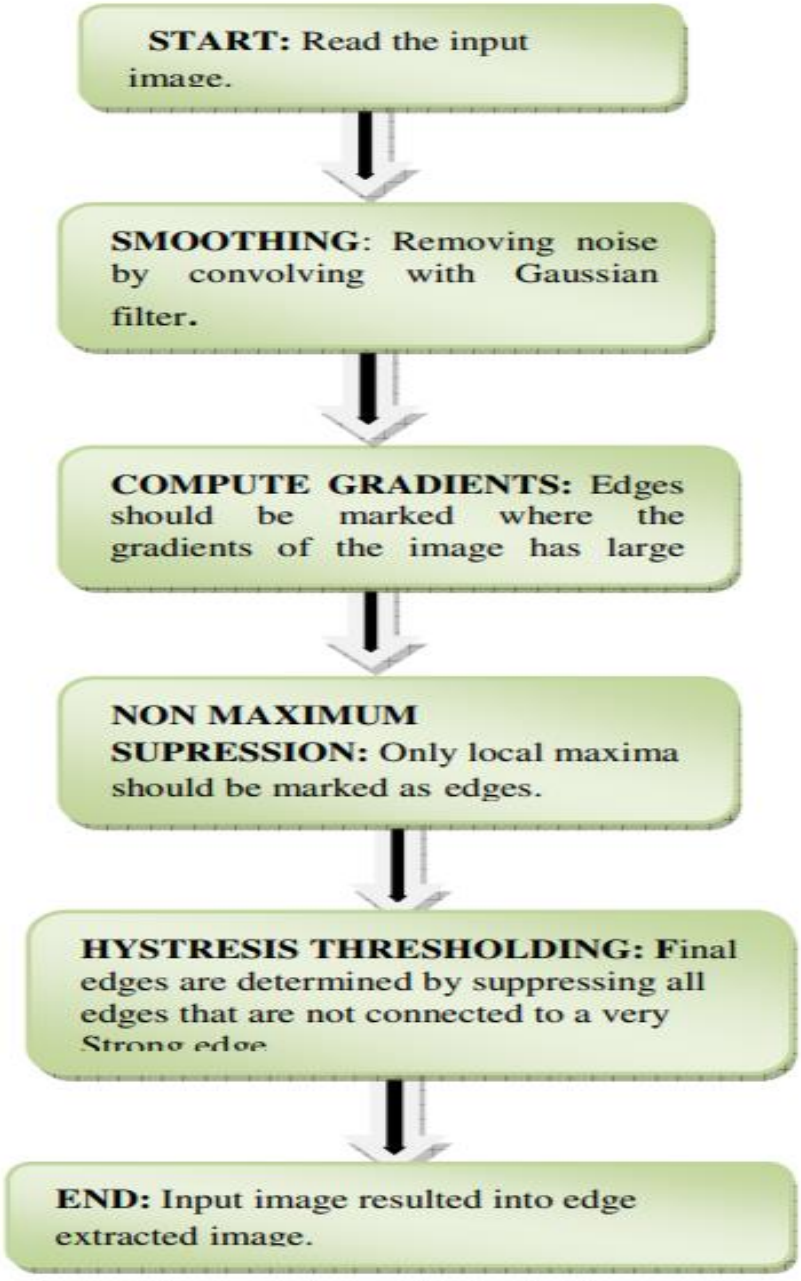
STEP I: Noise reduction by smoothing.

Noise contained in image is smoothed by convolving the input image I (i, j) with Gaussian filter G. Mathematically, the smooth resultant image is given by –

$$F(i, j) = G * I(i, j)$$

Prewitt operators are simpler to operator as compared to sobel operator but more sensitive to noise in comparison with sobel operator.

Flow chart of canny edge detection algorithm:



STEP II: Finding gradients:

In this step we detect the edges where the change in grayscale intensity is maximum. Required areas are determined with the help of gradient of images. Sobel operator is used to determine the gradient at each pixel of smoothened image. Sobel operators in i and j directions are given as

D_i = [[-1, 0, +1], [-2, 0, +2], [-1, 0, +1]] and D_j = [[+1, +2, +1], [0 , 0, 0], [-1, -2, -1]]

These sobel masks are convolved with smoothed image and giving gradients in i and j directions.

$$G_i = D_i * F(i, j) \text{ and } G_j = D_j * F(i, j)$$

Therefore edge strength or magnitude of gradient of a pixel is given by

$$G = \sqrt{(G_i^2 + G_j^2)}$$

The direction of gradient is given by theta = $\arctan(G_j/G_i)$

G_i And G_j are the gradients in the i- and j-directions respectively.

STEP III: Non maximum suppressions:

Non maximum suppression is carried out to preserves all local maxima in the gradient image, and deleting everything else this results in thin edges. For a pixel M (i, j):

• Firstly round the gradient direction theta nearest 45° ,then compare the gradient magnitude of the pixels in positive and negative gradient directions i.e. If gradient direction is east then compare with gradient of the pixels in east and west directions say E (i, j) and W (i, j) respectively.

• If the edge strength of pixel M (i, j) is largest than that of E (i, j) and W (i, j), then preserve the value of gradient and mark M (i, j) as edge pixel, if not then suppress or remove.

STEP IV: Hysteresis Thresholding:

The output of non-maxima suppression still contains the local maxima created by noise. Instead choosing a single threshold, for avoiding the problem of streaking two thresholds t_{high} and t_{low} are used. For a pixel M (i, j) having gradient magnitude G following conditions exists to detect pixel as edge:

➔ If $G < t_{low}$ than discard the edge.

➔ If G > than t_{high} keep the edge.

➔ If t_{low} < G < and t_{high} and any of its neighbors in a 3 ×3 region around it have gradient magnitudes greater than t_{high}, keep the edge.

➔ If none of pixel (x, y)'s neighbours have high gradient magnitudes but at least one falls between t_{low} and, t_{high} search the 5 × 5 region to see if any of these pixels have a magnitude greater than thigh. If so, keep the edge.

➔ Else, discard the edge.

Code and implementation:

For image (already in disk):

```
import cv2 as cv

import  numpy as np

img=cv.imread('a1.jpg')

cv.imwrite('RG.jpg',img)

img=cv.cvtColor(img,cv.COLOR_BGR2RGB)  #changing colors frames


cv.imwrite('BG.jpg',img)

img=cv.cvtColor(img,cv.COLOR_RGB2GRAY)


cv.imwrite('Gry.jpg',img)


img=cv.Canny(img,threshold1=100,threshold2=100)

cv.imwrite('Edge.jpg',img)

cv.imshow('Edge.jpg',img)

cv.waitKey(0)
```

cv.destroyAllWindows()

Result:

Original Image **Gray Scale Only** **Edges only**

For live via camera:

```
import cv2 as cv
import numpy as np import cv2 as cv
import numpy as np
cap=cv.VideoCapture(0)
while True:
    _,img=cap.read()                    #Reading Images From Capture.
    img=cv.cvtColor(img,cv.COLOR_BGR2RGB)     #Converting Frames BGR
to RGB.
    img=cv.cvtColor(img,cv.COLOR_RGB2GRAY)    #Converting Frmaes
RBG to Gray Scales.

    img=cv.Canny(img,threshold1=50,threshold2=100)
    cv.imshow('Egde',img)
    if(cv.waitKey(1) & 0xFF == ord('q')):
        break
cap.release()
cv.destroyAllWindows()
```

Result:

Description of Program:

The code for edge detection as written above has been written in python language using openCV and numpy library.

1. In code we import cv2 and numpy for using their function and method.
2. For opening camera, used cv2.VedioCapture() and run a loop. (for live).
3. Strat reading image in a variable, then changed it scale BGR to RGB and again RGB to GRAY using cv2.cvtColor().
4. Resizing the outer boundary and inner boundary by threshold value using Canny.
5. Writing image into local disk and displaying the edged image. (for live image running a loop which will be closed after pressed 'q' button).

Conclusion:

Project helpful in Real Life:

1. License plate detection: Today, cars are everywhere. Intelligent traffic control will no doubt become the future trend. So I will briefly discuss how we can apply edge detection in license plate detection. License plate detection technology is widely used in tollgates as well as parking lots in public places, companies, and residential areas. So improving this technology is of great practical value. First we conducted sample image gray scaling and QDPA operator edge detection. Below is a comparison of the two images (take this vehicle as an example)

2. Self-driving Car: in self driving car, after detecting edges, it identifies the lane lines in our image. However, there is a problem. It also identified the edges of the mountains and some of the trees. And based on various algorithm we define action and rules to our model.

3. Detecting hidden information in medical images: What distinguishes our phase-based detection methods from traditional edge detection methods such as Canny and Sobel is that we not only can detect the edges of an object; we can also detect some hidden information of the test object. It is impossible for these details to be detected via traditional methods because there is very little difference between the colour of these details and the colour of their surrounding regions.

What We Learnt From This Project:

It was a very interesting and exciting project as we are belong to background of computers. We learn lot of thing during the implementation of project and team working and research is very excited one in this project. A little description is given bellow how this project improved our intelligence and other things:

1. Canny edge detector gives better result with some positive points. It is less sensitive to noise, adaptive in nature, resolved the problem of streaking, provides good localization and detects sharper edges.
2. We learnt some module of Image Processing.
3. We learnt various functionality of opencv in python.
4. A deep knowledge of python programing and implementation of various data structure as we used array and list.
5. This project help allot to know various uses of image processing, what are the various field to use it in real life like diagnosis, self-driving car and many more.
6. We are glad after accomplished this project because it help us to know how technology are being used in real life practically.

Thank You!