Edge detection

Objective

- What is edge detection
- Purpose of edge detection in image processing
- Edge detection techniques

1. What is edge detection

Edge detection is a process of finding the boundaries of objects in an image. Edges are usually occur when there are changes in image intensity. A typical edge might be for example the skyline that separates the earth and sky or edges in a cube. There are 3 types of edges: horizontal, vertical, and diagonal. A curve edge can be represented as multiple consecutive diagonal edges. Every edges at a point has it own properties and can be calculated based on its neighbor.

2. Purpose of edge detection

In image processing, identify and classify objects need to go through many step and different techniques. Edge detection is one of them and it's the most important process because many techniques in object isolation are rely heavily on this process. The purpose of edge detection is

to find and marks all the spots where there are sudden changes in intensity of an image. Based on these boundaries, we could recognize and distinguish between objects and the background or locate certain objects.

3. Edge detection techniques

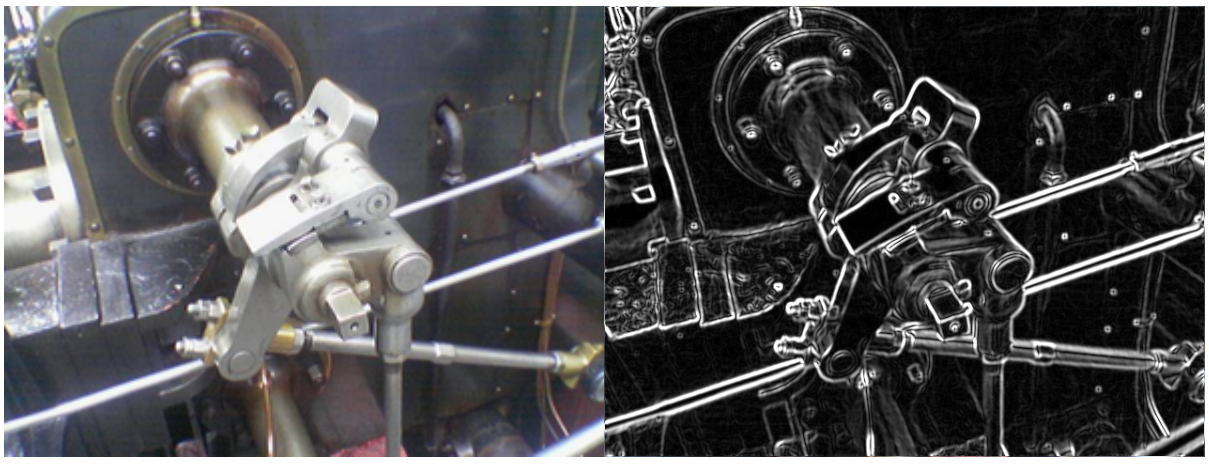
Sobel operator

Sobel operator is a edge detection method by using Gradient magnitude. There are also other techniques like Rober operator, Prewitt operator but Sobel operator produces the best result and it is common used in edge detection.

Sobel operator method can detect horizontal and vertical edge, using two 3x3 asymmetric kernels. These kernels are designed to work in detecting horizontal and vertical edges. When perform the convolution process using these kernels on a 2-D image, we could calculate the gradient in 2 direction x and y

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} and\ G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude $G = \sqrt{G_x^2 + G_y^2}$



Laplacian of Gaussian method

The problem with the above method is that it doesn't perform very well when there are noise in the image. In order to counter this weakness, we could perform a second order derivative or Laplacian method.

This method comprises 2 step:

1. Apply Gaussian filter to smooth the image in order to remove the noise
   Gaussian function:

$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2-y^2}{2\sigma^2}}$$

2. Perform second order derivative
   Laplacian operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

From the above equation, we produce the kernel for this method and apply convolution process

$$H = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Canny method

The canny method can be broken down to 5 step:

1. Apply Gaussian filter to smooth the image in order to remove the noise
2. Find the intensity gradients of the image
3. Apply non-maximum suppression to get rid of spurious response to edge detection
4. Apply double threshold to determine potential edges
5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

Comparision between methods

```
import sys
import cv2 as cv
from matplotlib import pyplot as plt

scale =  1
delta =  0
ddepth = cv.CV_16S

src = cv.imread('./lenna.png', cv.IMREAD_COLOR)
src = cv.GaussianBlur(src, (3, 3), 0)
```

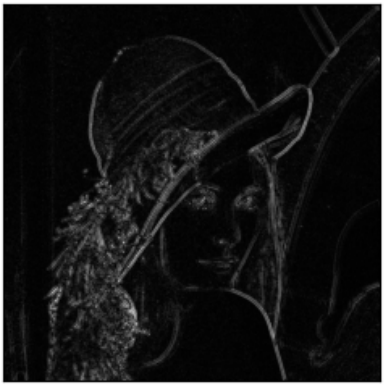
```
gray = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
grad_x = cv.Sobel(gray, ddepth, 1, 0, ksize=3, scale=scale, delta=delta, borderTyp
grad_y = cv.Sobel(gray, ddepth, 0, 1, ksize=3, scale=scale, delta=delta, borderTyp
abs_grad_x = cv.convertScaleAbs(grad_x)
abs_grad_y = cv.convertScaleAbs(grad_y)
sobel = cv.addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0)

laplacian = cv.Laplacian(gray,ddepth,ksize=3)
laplacian = cv.convertScaleAbs(laplacian)

canny = cv.Canny(gray,50,100)

plt.subplot(2,2,1),plt.imshow(gray,cmap  =  'gray')
plt.title('Original'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,2),plt.imshow(sobel,cmap  =  'gray')
plt.title('Sobel'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,3),plt.imshow(laplacian,cmap  =  'gray')
plt.title('Laplacian'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,4),plt.imshow(canny,cmap  =  'gray')
plt.title('Canny'), plt.xticks([]), plt.yticks([])

plt.show()
```

Result

Original

Sobel

Laplacian



Canny