

Edge Detection

Objective

The primary objective of this document is to provide information regarding to *edge detection* in Image processing. We will look through what are edges, why do we need to extract edges, and how can we do it via various methods.

Main ideas

Edge and Edge detection

1. Definition:

The sudden changes of discontinuities in an image are called edges. There are three types of edges:

- *Horizontal edges*
- *Vertical edges*
- *Diagonal edges*

Edge detection contains a variety of mathematical methods aimed to to identify these kind of sudden changes in images. This is a very useful technique with many purposes in technology and science.

2. Why

The main idea of why we detect edges is to capture the most important events or changes in an image for many purposes. Since edges are more compact than pixels, we use it for object recognition, images matching or images analysis and so on.

3. How

Based on the images, edges are likely to be caused by:

- discontinuities in *depth*.
- discontinuities in *surface orientation*.
- discontinuities in *illumination*.
- changes in *material properties*.

By applying edge detectors, we can generate a sub-image which contains a set of connected curves indicated the boundaries of the objects hence, the main events in an image. Additionally, we can also reduce the amount of redundant data to be processed and filter out any information which is irrelevant to the process. However, in real life, it is not simple to obtain successful tasks or the ideal edges due to the complexities of images.

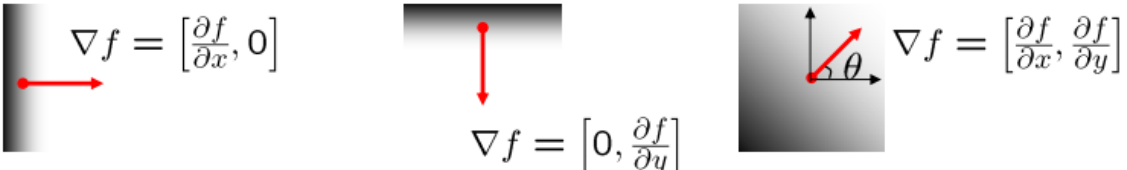
Method for extraction

There are many approaches in extracting edges of an image, as edges typically occur on the boundary between two different regions in the image, we can use a number of algorithms classified as derivative based on where the algorithm takes first or second derivative on pixels or gradient based on a gradient of consecutive pixels taken in x and y direction. In this document, we shall discuss mostly on how we can extract using image gradient (with sobel operator) and canny edge detection.

Image gradient

Image gradient is nothing but directional changes in images' intensity. Via the calculation of gradient in a small area of image and then repeat that process over and over, this approach will help us detect the edge.

The gradient of an image can be represented as function:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$


And its absolute value:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Each calculation is being done in a small area of the image and we need to repeat this process until all regions have been computed.

Based on this method, variations have been created to enhance the efficiency of detect edges, reducing amount of workloads.

Sobel operator

One of the most common used operator is the Sober operator, invented by Irwin Sobel and Gary Feldman, it uses two 3x3 matrix kernel each for x and y direction to calculate the approximation of the derivative of an image.

X – Direction Kernel

-1	0	1
-2	0	2
-1	0	1

Y – Direction Kernel

-1	-2	-1
0	0	0
1	2	1

The point is to find the amount of differences by placing the gradient matrix over each pixel of an images. It is not the best options but it can still be a very useful approach since calculation can be done quickly.

Canny edge detection

Developed by John F. Canny in 1986, Canny edge detector is an operator used a multi-stage algorithm to detect edges. In general, the process can be broken down into 5 major steps:

- 1. Smoothing:** Apply a Gaussian filter to smooth the image with the intention to remove the noise.
- 2. Gradient:** Find and compute the intensity gradients of the smoothed image.
- 3. Thresholding:** Apply threshold to determine potential edges while suppressed noises.
- 4. Non-maximum suppression (thinning):** Apply non-maximum suppression to zero out all pixels which are not the maximum along the direction of the gradient.
- 5. Tracing edges:** Suppress all the weak edges while tracing high-magnitude contours, all the strong pixels are kept.

Examples in OpenCV

Code

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

lena = cv.imread(cv.samples.findFile("Resource/lena.png"), cv.IMREAD_GRAYSCALE)

sobelx = cv.Sobel(lena, cv.CV_64F, 1, 0, ksize=5)
sobely = cv.Sobel(lena, cv.CV_64F, 0, 1, ksize=5)
canny = cv.Canny(lena, 100, 200)

plt.subplot(2, 1, 1), plt.imshow(lena, cmap="gray")
plt.title("Original"), plt.xticks([]), plt.yticks([])
plt.subplot(2, 2, 3), plt.imshow(sobelx, cmap = 'gray')
plt.title('Sobel X'), plt.xticks([]), plt.yticks([])
plt.subplot(2, 2, 4), plt.imshow(sobely, cmap = 'gray')
plt.title('Sobel Y'), plt.xticks([]), plt.yticks([])

plt.show()

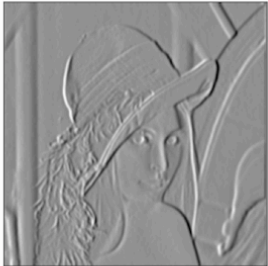

plt.subplot(121),plt.imshow(lena,cmap = 'gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(canny,cmap = 'gray')
plt.title('Edge Image'), plt.xticks([]), plt.yticks([])
plt.show()

key = cv.waitKey(0)
if key == ord("q"):
    sys.exit()
```

Result

Original Image:



Method	Outcomes
Sobel operator	<div>Sobel X</div>  <div>Sobel Y</div> 
Canny edge detector	

Conclusion

Edge detection is a valuable method used in image processing, computer vision and machine vision for many reasons. It is clear that we need to keep developing and utilize these methods to the maximum value.

