Filters and Edge Detection

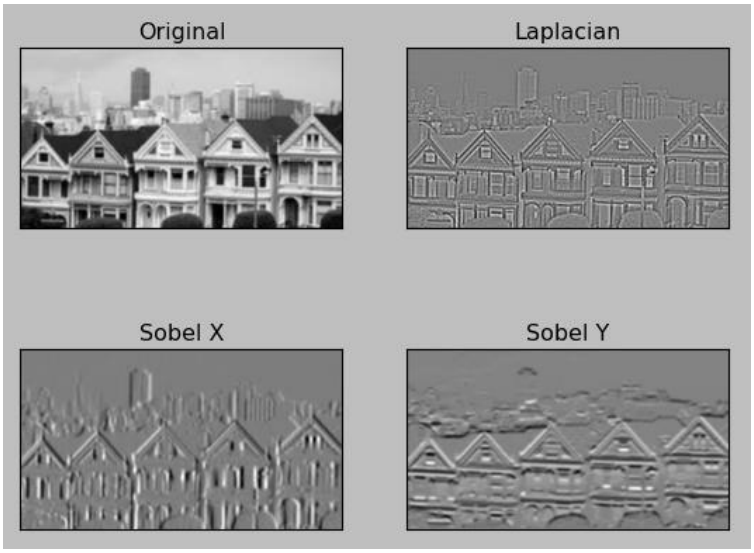
By: Yahia Ahmed Saber

Sobel filter

A pair of convolutional filters are applied to an image. These two kernels are used to detect edges in an image, with Gx focusing on horizontal edges and Gy on vertical edges. Each kernel is applied on its own and the results are used together to compute the pixel value at each position in the output image. Sobel filter is often used to work out the approximate magnitude of absolute gradient at each point in an image.

Code:

```
import cv2
import numpy as np
img = cv2.imread('image.bmp', cv2.IMREAD_GRAYSCALE)
rows, cols = img.shape
sobel_horizontal = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=5)
sobel_vertical = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=5)
cv2.imshow('Original', img)
cv2.imshow('Sobel horizontal', sobel_horizontal)
cv2.imshow('Sobel vertical', sobel_vertical)
cv2.waitKey(0)
```



Laplacian filter

Mostly, laplacian filters are used in edge detection as they can highlight the edges in an image. Its importance lies in how well it can draw out image features. It works by applying a kernel of weights to each grid cell and its neighbouring cells in an image.

Code:

```
import cv2
import matplotlib.pyplot as plt
image = cv2.imread(r"E:\eye.png", cv2.IMREAD_COLOR)
image = cv2.GaussianBlur(image, (3, 3), 0)
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
filtered_image = cv2.Laplacian(image_gray, cv2.CV_16S, ksize=3)
```

Canny Edge Detector

Canny edge detection is another great method of edge detection which extracts edges in an image. An instant change in pixel intensity represents an edge and these edges show the boundaries of different objects. The algorithm works by first turning the image into grayscale then a gaussian filter is applied for noise reduction then the gradient of the smoothed image is calculated using the sobel filter.

Code:

```
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('Brand.jpeg')
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
edges = cv.Canny(gray,100,200)
im_rgb = cv.cvtColor(img, cv.COLOR_BGR2RGB)
plt.subplot(121),plt.imshow(im_rgb,cmap = 'Blues')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(edges,cmap = 'bone')
plt.title('Edge Image'), plt.xticks([]), plt.yticks([])
plt.show()
```

Contours in Image processing

Contours are used to define boundaries of objects and set them apart from the background and they are often shaped as curves. They join consecutive points which share the same intensity. With the help of edge detection techniques, contours are detected then they can be processed using filtering and feature extraction.

Code:

```
import cv2
import matplotlib.pyplot as plt
image = cv2.imread("thumbs_up_down.jpg")
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
_, binary = cv2.threshold(gray, 225, 255, cv2.THRESH_BINARY_INV)
plt.imshow(binary, cmap="gray")
plt.show()
contours, hierarchy = cv2.findContours(binary, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
image = cv2.drawContours(image, contours, -1, (0, 255, 0), 2)
plt.imshow(image)
plt.show()
```