Filters and edge detection

1.Sobel Filter:

This filter is a basic tool in image processing that helps to find the edges in an image. It works by looking at how the brightness of an image changes from one pixel to the next. By focusing on these changes, the Sobel Filter can highlight areas where there are sharp transitions, which usually correspond to edges. It uses two small matrices (called kernels) to detect changes in both horizontal and vertical directions.

Mainly it is used for tasks like identifying the outline of objects in an image, which is essential in many computer vision applications.

Python example:

```
 image = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)
sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0)
sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1) # Vertical edges
sobel_combined = cv2.magnitude(sobel_x, sobel_y)
cv2.imshow('Sobel Edge Detection', sobel_combined)
 cv2.waitKey(0)
 cv2.destroyAllWindows()
```

2.The Laplacian Filter:

This filter is edge detection tool, but it takes a different approach. Instead of just looking at the first changes in brightness, it looks at how these changes themselves are changing. This makes the Laplacian Filter particularly good at finding edges that change rapidly in all directions.

Main uses for the Laplacian Filter is when we want to sharpen an image or make the edges stand out more clearly.

Python Example:

```
image = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)
laplacian = cv2.Laplacian(image, cv2.CV_64F)


cv2.imshow('Laplacian Edge Detection', laplacian)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

3.Canny Edge Detector:

The Canny Edge Detector is more advanced method for finding edges in images. It is like a step-by-step process: first, it smooths the image to reduce noise, then finds the intensity changes, and finally identifies the most important edges by filtering out less significant ones. The result is a clean set of edges that are easy to work with.

Canny is widely used in image processing because it provides very clear and accurate edge detection. It's often used in applications like facial recognition, object detection, and image segmentation.

Python Example:

```
import cv2


image = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)
edges = cv2.Canny(image, 100, 200)


cv2.imshow('Canny Edge Detection', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

<u>4.Contours</u>

Contours are basically the outlines or boundaries of objects in an image. In image processing, finding contours means identifying the shapes in an image by tracing the edges. This is useful for tasks like identifying and measuring objects or recognizing shapes.

Contours are used in a variety of tasks such as detecting objects in images, analyzing shapes, and even recognizing hand gestures.

Python Example:

```
import cv2
image = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)
binary = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)
```

```
contours,_=cv2.findContours(binary, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

contour_image = cv2.cvtColor(image, cv2.COLOR_GRAY2BGR)

cv2.drawContours(contour_image, contours, -1, (0, 255, 0), 2)

cv2.imshow('Contours', contour_image)

cv2.waitKey(0)

cv2.destroyAllWindows()
```