6. Computer Vision – Use Cases

Contents

6. Computer Vision – Use Cases

1. Edge & Contour Detection

- ✓ CV applications detect edges first and then collect other information.
- ✓ There are many edge detection algorithms, and the most popular is the Canny edge detector because it's pretty effective compared to others.
- ✓ It's also a complex edge-detection technique.
- ✓ Below are the steps for Canny edge detection:
  - Reduce noise and smoothen image
  - Calculate the gradient
  - Non-maximum suppression
  - Double the threshold
  - Linking and edge detecting

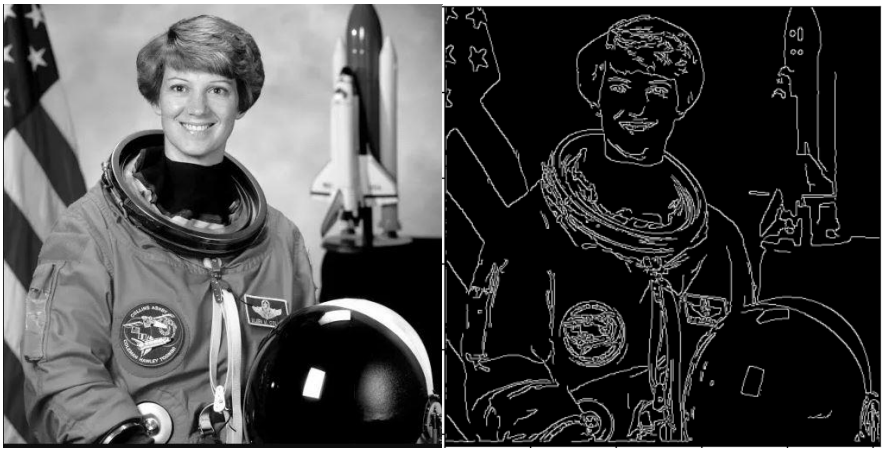
Program Name: Canny edge detection
demo1.py

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('1.jpg')
edges = cv2.Canny(img, 100, 200, 3, L2gradient = True)

plt.figure()
plt.title('Spider')
plt.imsave('dancing-spider-canny.png', edges, cmap = 'gray',
format='png')
plt.imshow(edges, cmap='gray')
plt.show()
```

Input



2. Contours

- ✓ Contours are lines joining all the continuous objects or points (along the boundary), having the same color or intensity.
- ✓ For example, it detects the shape of a leaf based on its parameters or border.
- ✓ Contours are an important tool for shape and object detection.
- ✓ The contours of an object are the boundary lines that make up the shape of an object as it is.
- ✓ Contours are also called outline, edges, or structure, for a very good reason.

Program Name: Contours
demo2.py

```
import cv2
import numpy as np

image = cv2.imread('2.jpg')
cv2.waitKey(0)

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

edged = cv2.Canny(gray, 30, 200)
cv2.waitKey(0)

contours, hierarchy = cv2.findContours(edged,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

cv2.imshow('Canny Edges After Contouring', edged)
cv2.waitKey(0)

cv2.drawContours(image, contours, -1, (0, 255, 0), 3)

cv2.imshow('Contours', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input