STB600 Lab 4:Image segmentation

Atieh Sahraeidolatkhaneh: atieh.sahraeidolatkhaneh@hv.se

Yongcui Mi: yongcui.mi@hv.se

David Stjern: david.stjern@hv.se

1. Objectives

The objective of this lab is to learn different image segmentation methods. Six different images are provided to work with.

2. Tasks

- Prewitt edge detector
- Soble edge detector
- Canny edge detector
- Hough Transform
- Intensity Otsu Thresholding
- Adaptive (variable) thresholding

2.1. Prewitt edge detector

The Prewitt operator is used for edge detection in an image. Prewitt operator detects edges along the x-axis and edges along the y-axis (apply it on all images). Prewitt operator provides two masks one for detecting edges along x-axis and another for edges along y-axis:

[x-axis] = ([[-1,0,1],[-1,0,1],[-1,0,1]])

[y-axis] = ([[1,1,1],[0,0,0],[-1,-1,-1]])

```
import cv2
import numpy as np

# Read image

# Convert image to grayscale if it is not

#image filtering
```

```
#using your preferred filtering method

# detect edges along x-axis and y-axis using Prewitt

# cv2.filter2D(src, ddepth, input kernel)
# hint: ddepth is the desirable depth of destination image. Value -1
    represents that the result image will have same depth as the src
    image.

# Combine the edges detected along the X and Y axes

# Show image
```

2.2. Soble edge detector

Sobel Operator is a specific type of 2D derivative mask which is efficient in detecting the edges in an image. Sobel kernel:

[x-axis] = ([[-1,0,1],[-2,0,2],[-1,0,1]])

[y-axis] = ([[1,2,1],[0,0,0],[-1,-2,-1]])

```
import cv2
import numpy as np

# Read image

# Convert image to grayscale if it is not

#image filtering
# Blur the image using your preferred filtering method

# Apply Sobel edge detection on the X and Y axis
# cv2.sobel(src, ddepth, dx, dy, ksize)

# Show image
```

2.3. Canny edge detector

Canny edge detection is a multi-stage algorithm which first remove noise in the image using a 5x5 Gaussian filter, secondly finds the intensity gradient of the image, thirdly it convert the image to a binary one and in the last stage it applies hysteresis thresholding

to detect the edges with maximum intensity. However, you can use Canny edge detection using a single function in OpenCV. Canny edge detector decides which are all edges are really edges and which are not. For this, we need two threshold values, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded.

```
import cv2
import numpy as np

# Read image

# Convert image to grayscale if it is not

#image filtering
# Blur the image using your preferred filtering method

# Apply Canny edge detection on the X and Y axis
# cv.Canny(src, threshold1, threshold2)

# Show image
```

2.4. Hough Transform

Hough Transform is a popular technique to detect any shape, if you can represent that shape in mathematical form. It can detect the shape even if it is broken or distorted a little bit. Use the built-in Hough Transform function in OpenCV cv2.HoughLines(). First parameter, Input image should be a binary image, so apply threshold or use canny edge detection before finding applying hough transform. Second and third parameters are ρ and θ accuracies respectively.

Fourth argument is the threshold, which means minimum vote it should get for it to be considered as a line. Remember, number of votes depend upon number of points on the line. So it represents the minimum length of line that should be detected. Compaere its result with Probabilistic Hough Transform (cv2.HoughLinesP()) It has two new arguments:

minLineLength - Minimum length of line. Line segments shorter than this are rejected.

maxLineGap - Maximum allowed gap between line segments to treat them as single line.

```
import cv2
import numpy as np

# Read image

# Convert image to grayscale if it is not

# Detect edges

# Apply Hough Transform

# Draw the lines on the image

# Save image

# Show image
```

2.5. Otsu Thresholding

In Otsu Thresholding, a value of the threshold isn't chosen but is determined automatically.

cv2.threshold(src, thresholdValue, maxVal, thresholdingTechnique)

src: Input img (must be in Grayscale).

thresholdValue: Value of Threshold below and above which pixel values will change accordingly.

maxVal: Maximum value that can be assigned to a pixel.

thresholdingTechnique: The type of thresholding to be applied.

Hint: use the Traditional cv2.threshold function and use `cv2.THRESH_OTSU` as an extra flag

```
import cv2
import numpy as np

# Read image

# Convert image to grayscale if it is not


```

```
# Apply Adaptive thresholding

# Show thresholded image in a new window
```

2.6. Adaptive Thresholding

Adaptive thresholding is the method where the threshold value is calculated for smaller regions. This leads to different threshold values for different regions with respect to the change in lighting. Use cv2.adaptiveThreshold(src, maxVal, adaptiveMethod) as function.

src: Input Img

maxVal: Maximum value that can be assigned to a pixel.

adaptiveMethod: Adaptive method decides how threshold value is calculated.

`cv.ADAPTIVE_THRESH_MEAN_C`: The threshold value is the mean of the neighbourhood area minus the constant C.

`cv.ADAPTIVE_THRESH_GAUSSIAN_C`: The threshold value is a gaussian-weighted sum of the neighbourhood values minus the constant C.

Compares global thresholding and adaptive thresholding for an image with varying illumination.

```
import cv2
import numpy as np


# Read image

# Convert image to grayscale & blur it, if it is needed

# Apply global (ex;Binary) thresholding

# Apply Adaptive thresholding

# Show thresholded images in one or multiple windows
```

3. Additional resource

References