Segmenting Towers and Transmission Lines

(Using OpenCV in Python)

IRTIQA MALIK

December, 2019

Contents

INTRODUCTION

Segmentation of Transmission Lines and Towers involves processing of digital media to extract the position of a tower transmission line.Edge detection can be used to extract the outline of a tower/transmission line from a digital image or a video frame.
The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. The basic steps of Canny edge detection are as follows:

- First a Gaussian blur is applied to the image to reduce noise.
- Thresholding can also be used if the objects of interest are significantly contrasted from the background and detailed textures are irrelevant.
- Next the edge direction and gradient strength of each pixel in the image are found using Sobel masks.Then edges are traced using that information.
- Finally non maximum edges are suppressed by finding parallel edges and eliminating those with weaker gradient strengths.

SOFTWARE IMPLEMENTATION

I have used the cv2 library which is available in Python and within this library we have the following functions that are used:

1. imread() : this reads the given image into a matrix of rgb values of individual pixels.
2. cvtColor (): is used to transform the rgb space into grey.
3. blur(): is used to blur the given image .
4. Canny(): is used to extract the edges from the given image with the threshold margin specified.
5. VideoCapture(): is used to obtain a video capture object which is later used to read the video frame by frame
6. read method of video capture is used to read video frame by frame
7. Imshow() is used to display images
8. Imwrite() is used to save images

The Algorithmic Approach followed is:

1. Obtain frame: which in video is by reading individual frames and in images is simply reading the image
2. Convert to Grey Colorspace: which is simply to convert to grey color space from rgb.
3. Blur: Use the blur function to introduce blur into the grayscale image
4. Use the canny edge detector to extract edge
5. Fine tune the result by changing parameters of canny edge function
6. Save image / video frame

RESULTS

The code for implementation of the Canny edge detection algorithm used is shown in the picture below.

```
#Canny edge Function
def cannyedge(img_grey):
    low_threshold=cv2.getTrackbarPos("Min Threshold","Edge_View")
    high_threshold=cv2.getTrackbarPos("Maximum Threshold","Edge_View")
    detected_edges=cv2.blur(img_grey,(3,3))
    cv2.Canny(img_grey,low_threshold,high_threshold,detected_edges,3,True)
    return detected_edges
```

Figure 1: Code Snipet

It uses Canny Edge Detector whose threshold values can be varied using two trackbars. On application of this canny edge detector to the input image as shown below, we segment only transmission lines and Towers and get the output with minimum noise when we set the Min Threshold and Max Threshold at values 118 and 256 respectively using the trackbars.

Figure 2: Input

The output image is shown below:

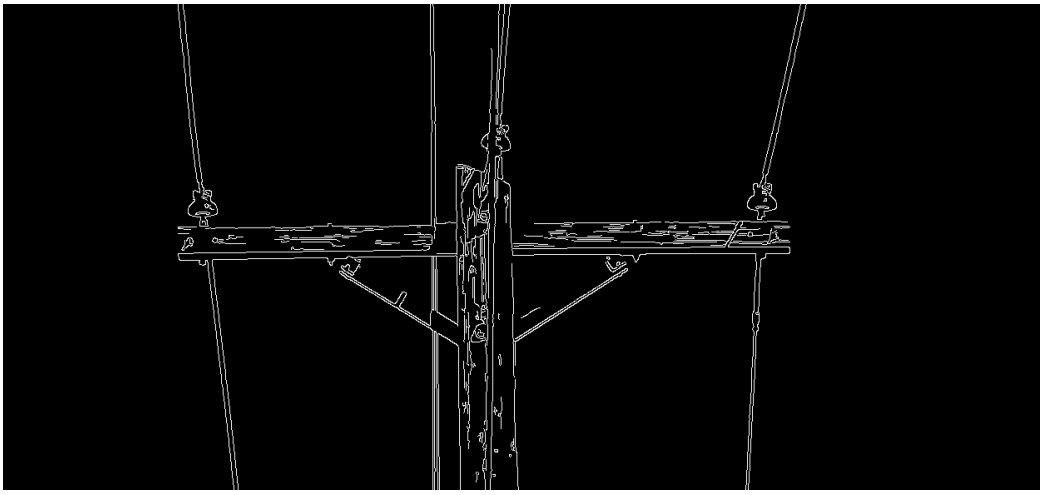

Figure 3: Output

REFERENCES USED

In completing this project I have used the python documentation and openCV library documentation .