GOA COLLEGE OF ENGINEERING

"Bhausaheb Bandodkar Technical Education Complex"

Experiment No: 10 **Date:**

Aim: Write a program to implement Canny edge detector.

Theory: The basic steps involved in this algorithm are:

1. Noise reduction using Gaussian filter
This step is of utmost importance in the Canny edge detection. It uses a Gaussian filter for the removal of noise from the image, it is because this noise can be assumed as edges due to sudden intensity change by the edge detector. The sum of the elements in the Gaussian kernel is 1, so, the kernel should be normalized before applying as convolution to the image.

2. Gradient calculation
When the image is smoothed, the derivatives Ix and Iy are calculated w.r.t x and y axis. It can be implemented by using the Sobel-Feldman kernels convolution after applying these kernel we can use the gradient magnitudes and the angle to further process this step.

3. Non-Maximum Suppression
This step aims at reducing the duplicate merging pixels along the edges to make them uneven. For each pixel find two neighbors in the positive and negative gradient directions, supposing that each neighbor occupies the angle of pi /4, and 0 is the direction straight to the right. If the magnitude of the current pixel is greater than the magnitude of the neighbors, nothing changes, otherwise, the magnitude of the current pixel is set to zero.

4. Double Thresholding
The gradient magnitudes are compared with two specified threshold values, the first one is lower than the second. The gradients that are smaller than the low threshold value are suppressed, the gradients higher than the high threshold value are marked as strong ones and the corresponding pixels are included in the final edge map. All the rest gradients are marked as weak ones and pixels corresponding to these gradients are considered in the next step.

5. Edge Tracking using Hysteresis
Since a weak edge pixel caused by true edges will be connected to a strong edge pixel, pixel W with weak gradient is marked as edge and included in the final edge map if and only if it is involved in the same connected component as some pixel S with strong gradient. In other words, there should be a chain of neighbor weak pixels connecting W and S (the neighbors are 8 pixels around the considered one). We will make up and implement an algorithm that finds all the connected components of the gradient map considering each pixel only once. After that, you can decide which pixels will be included in the final edge map.

Program:

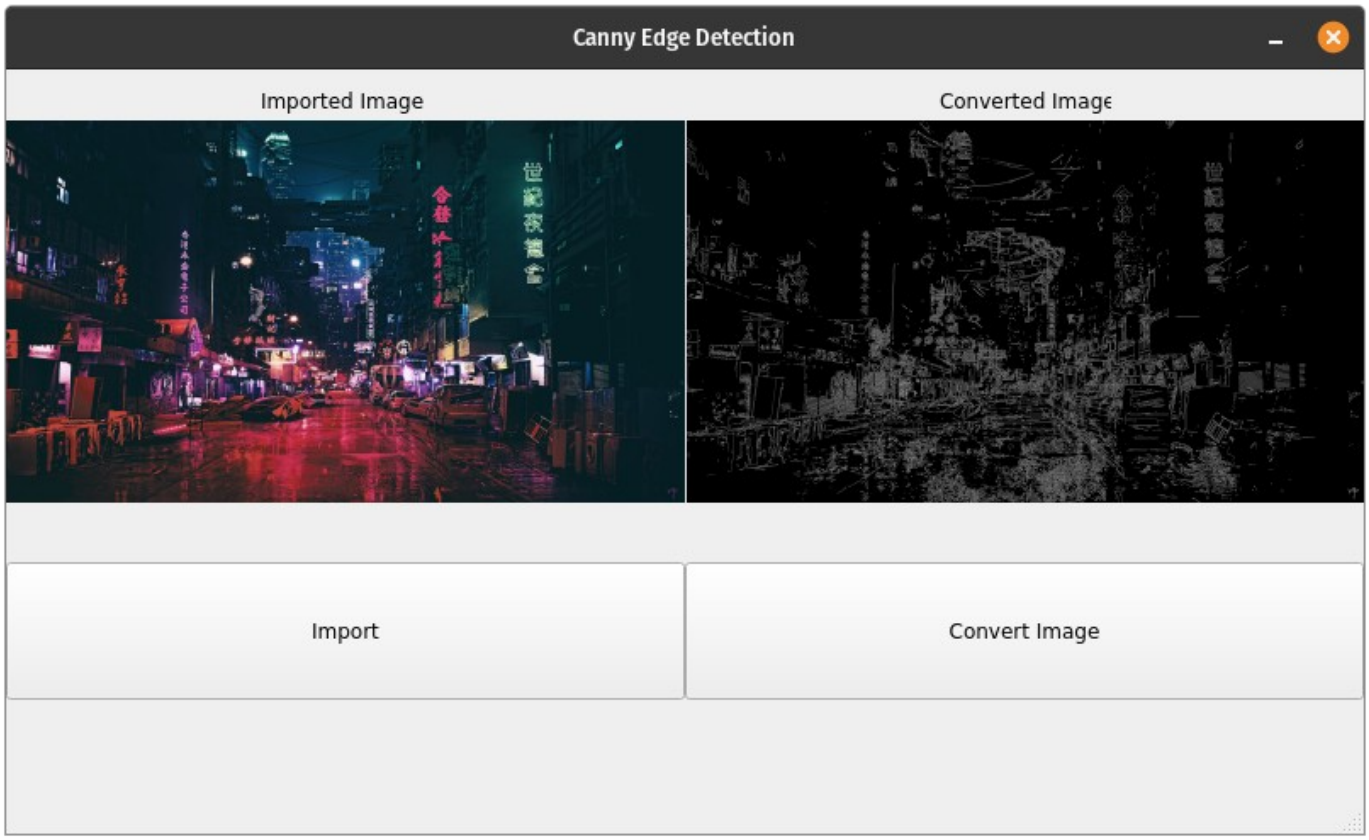
Python Code:

```
def show_image(self):
file_filter = 'Image File (*.jpg *.png)'
fname = QtWidgets.QFileDialog.getOpenFileName(parent=self.centralwidget,
caption='Select an Image',
directory="/run/media/deeprajb/HDD/Important Photos/Wallpapers",
filter=file_filter)
self.img = cv2.imread(fname[0])
self.img1 = QtGui.QImage(self.img.data, self.img.shape[1], self.img.shape[0],
QtGui.QImage.Format_RGB888).rgbSwapped()
self.imageinput.setPixmap(QtGui.QPixmap.fromImage(self.img1))
def canny_edge(self):
```

```
canny = cv2.Canny(self.img, 150, 200)
cv2.imwrite('canny_output.jpg',canny)
self.imageoutput.setPixmap(QtGui.QPixmap("canny_output.jpg"))
```

Output:

Python GUI Output:



Conclusion: Program to read an image and perform Canny Edge Detection was studied and the code was implemented successfully.