Experiment No: 4

Aim: Write a Python program to read an image from a file system and perform the grayscale slicing of the image and write it back with a different name.

Algorithm:

1. Read the image and store it in a container to perform operations on it.
2. Convert this image into grayscale image
3. Input 2 values for upper limit and lower limit
4. Get the RGB value of the pixel
5. Calculate new RGB value based on : If the value of pixel :
    1. Within limits of upper & lower limits → new RGB value = 255, 255, 255 resp.
    2. Out of limits → new RGB values = 0, 0, 0
6. Save the new RGB value in the pixel
7. Repeat step 4 -6 for each pixel of the image
8. Choose a directory to store the new image
9. Store the sliced image into the selected directory

Program:

Python Code:

```
from PyQt5 import QtCore, QtGui, QtWidgets
import cv2

class Ui_MainWindow(object):
def setupUi(self, MainWindow):
MainWindow.setObjectName("MainWindow")
MainWindow.resize(800, 700)
self.centralwidget = QtWidgets.QWidget(MainWindow)
self.centralwidget.setObjectName("centralwidget")
self.imageinput = QtWidgets.QLabel(self.centralwidget)
self.imageinput.setGeometry(QtCore.QRect(0, 20, 400, 225))
self.imageinput.setText("")
self.imageinput.setPixmap(QtGui.QPixmap(""))
self.imageinput.setScaledContents(True)
self.imageinput.setObjectName("imageinput")
self.import_image = QtWidgets.QPushButton(self.centralwidget)
self.import_image.setGeometry(QtCore.QRect(0, 520, 400, 81))
self.import_image.setObjectName("import_image")
self.output = QtWidgets.QPushButton(self.centralwidget)
self.output.setGeometry(QtCore.QRect(400, 520, 400, 81))
self.output.setObjectName("output")
self.imageoutput = QtWidgets.QLabel(self.centralwidget)
```

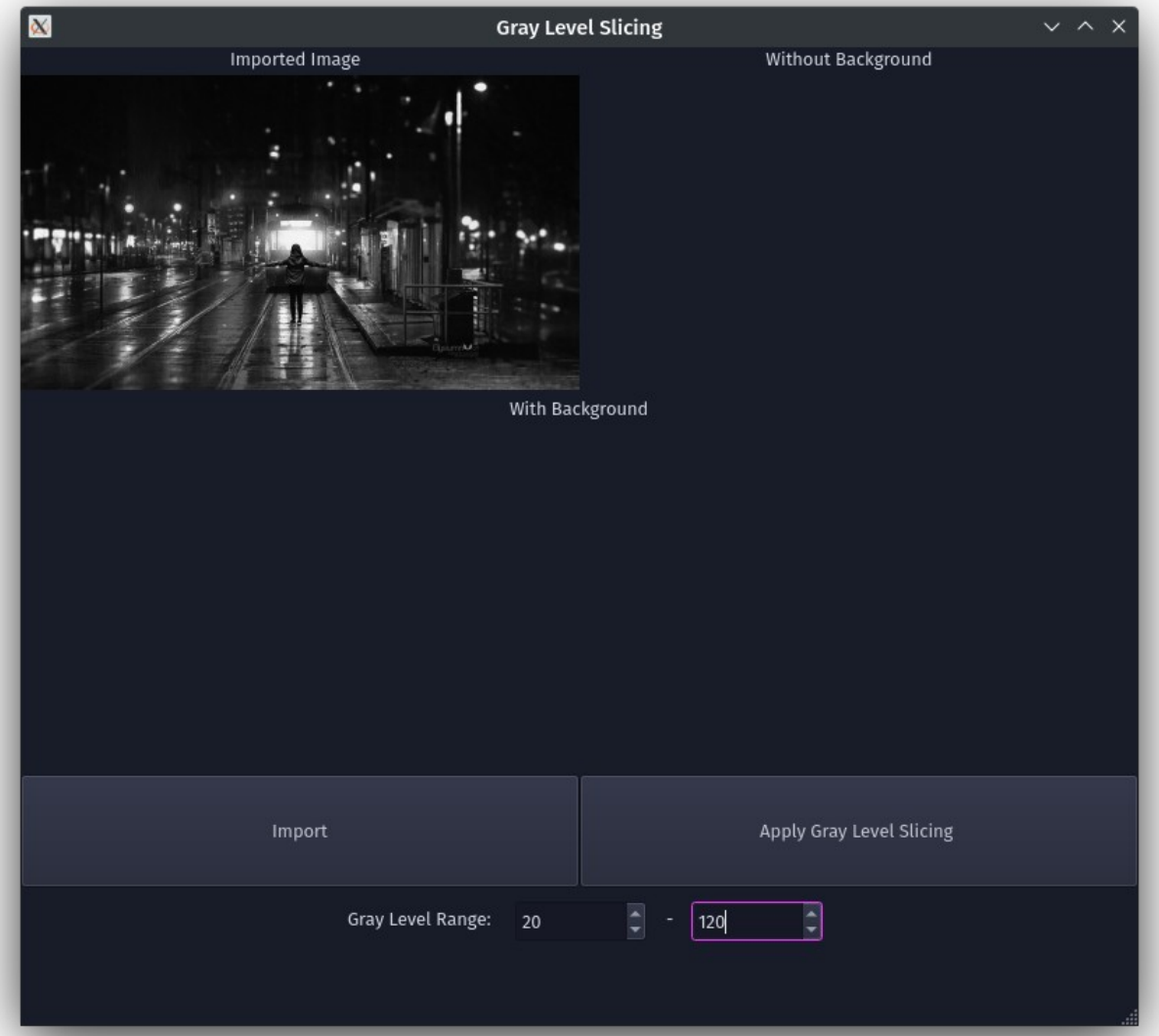
```
self.imageoutput.setGeometry(QtCore.QRect(401, 20, 400, 225))
self.imageoutput.setText("")
self.imageoutput.setPixmap(QtGui.QPixmap(""))
self.imageoutput.setScaledContents(True)
self.imageoutput.setObjectName("imageoutput")
self.label = QtWidgets.QLabel(self.centralwidget)
self.label.setGeometry(QtCore.QRect(150, 0, 101, 17))
self.label.setObjectName("label")
self.label_2 = QtWidgets.QLabel(self.centralwidget)
self.label_2.setGeometry(QtCore.QRect(530, 0, 131, 17))
self.label_2.setObjectName("label_2")
self.label_3 = QtWidgets.QLabel(self.centralwidget)
self.label_3.setGeometry(QtCore.QRect(350, 250, 101, 17))
self.label_3.setObjectName("label_3")
self.imageoutput_2 = QtWidgets.QLabel(self.centralwidget)
self.imageoutput_2.setGeometry(QtCore.QRect(200, 270, 400, 225))
self.imageoutput_2.setText("")
self.imageoutput_2.setPixmap(QtGui.QPixmap(""))
self.imageoutput_2.setScaledContents(True)
self.imageoutput_2.setObjectName("imageoutput_2")
self.label_4 = QtWidgets.QLabel(self.centralwidget)
self.label_4.setGeometry(QtCore.QRect(234, 615, 111, 16))
self.label_4.setObjectName("label_4")
self.label_5 = QtWidgets.QLabel(self.centralwidget)
self.label_5.setGeometry(QtCore.QRect(462, 615, 56, 17))
self.label_5.setObjectName("label_5")
self.lowerbound = QtWidgets.QSpinBox(self.centralwidget)
self.lowerbound.setGeometry(QtCore.QRect(354, 610, 94, 30))
self.lowerbound.setMaximum(254)
self.lowerbound.setObjectName("lowerbound")
self.upperbound = QtWidgets.QSpinBox(self.centralwidget)
self.upperbound.setGeometry(QtCore.QRect(480, 610, 94, 30))
self.upperbound.setMaximum(254)
self.upperbound.setObjectName("upperbound")
MainWindow.setCentralWidget(self.centralwidget)
self.menubar = QtWidgets.QMenuBar(MainWindow)
self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 27))
self.menubar.setObjectName("menubar")
MainWindow.setMenuBar(self.menubar)
self.statusbar = QtWidgets.QStatusBar(MainWindow)
self.statusbar.setObjectName("statusbar")
MainWindow.setStatusBar(self.statusbar)

self.retranslateUi(MainWindow)
QtCore.QMetaObject.connectSlotsByName(MainWindow)

#Importing Image
```

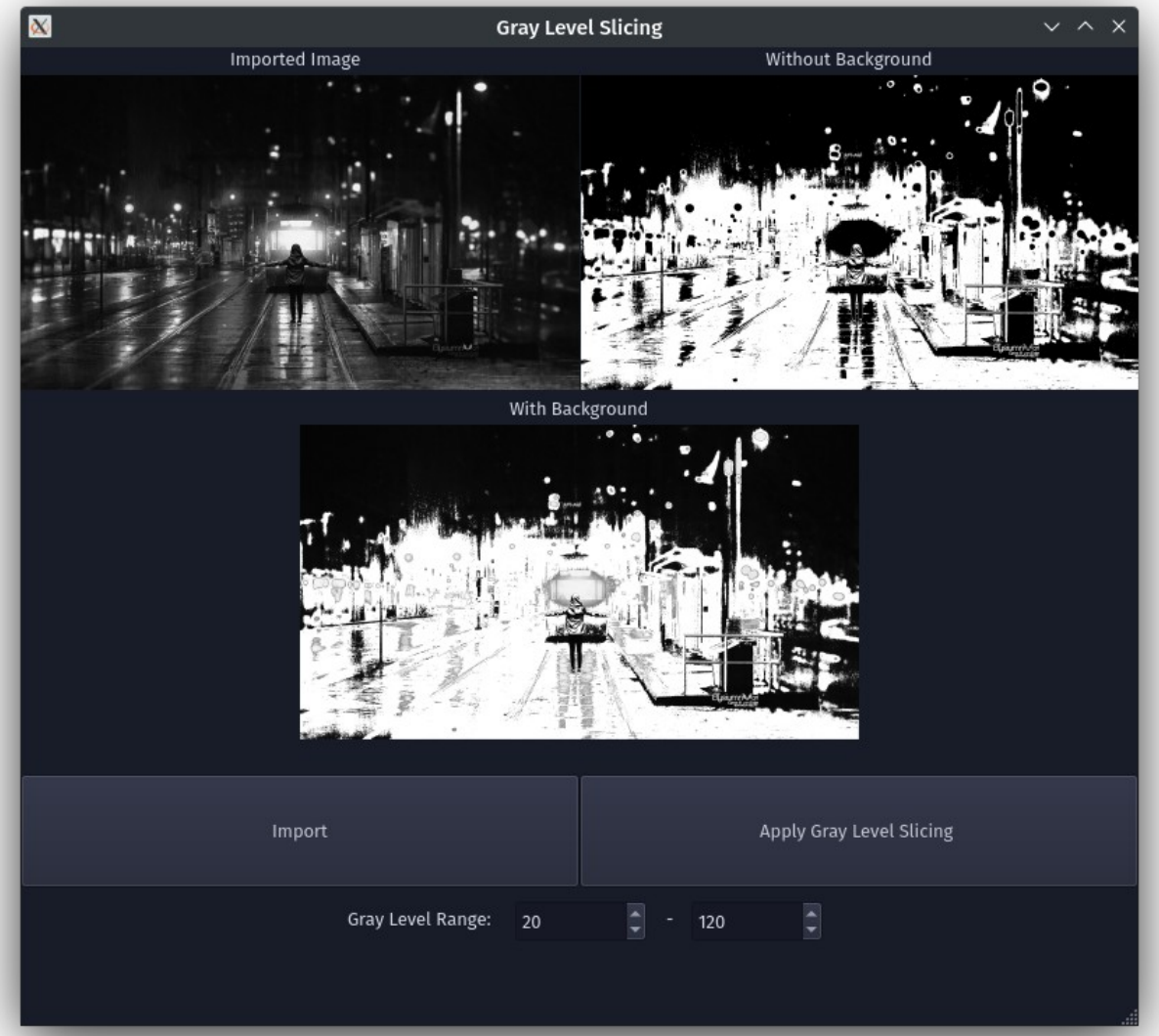
```
self.import_image.clicked.connect(self.show_image)

#Checking Output Image
self.output.clicked.connect(self.gray_level_slicing)

def retranslateUi(self, MainWindow):
_translate = QtCore.QCoreApplication.translate
MainWindow.setWindowTitle(_translate("MainWindow", "Gray Level Slicing"))
self.import_image.setText(_translate("MainWindow", "Import"))
self.output.setText(_translate("MainWindow", "Apply Gray Level Slicing "))
self.label.setText(_translate("MainWindow", "Imported Image"))
self.label_2.setText(_translate("MainWindow", " Without Background"))
self.label_3.setText(_translate("MainWindow", "With Background"))
self.label_4.setText(_translate("MainWindow", "Gray Level Range: "))
self.label_5.setText(_translate("MainWindow", "-"))

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], cv2.IMREAD_GRAYSCALE)
self.img1 = QtGui.QImage(self.img.data, self.img.shape[1], self.img.shape[0],
QtGui.QImage.Format_Grayscale8)
self.imageinput.setPixmap(QtGui.QPixmap.fromImage(self.img1))
def gray_level_slicing(self):
(row, col) = self.img.shape[0:2]
min_range = self.lowerbound.value()
max_range = self.upperbound.value()
wbg = self.img.copy()
wobg = self.img.copy()
for i in range(0,row-1):
for j in range(0,col-1):
if self.img[i,j]>min_range and self.img[i,j]<max_range:
wbg[i,j] = 255
wobg[i,j] = 255

else:
wbg[i,j] = self.img[i,j]
wobg[i,j] = 0
cv2.imwrite('gls_withbg_output.jpg',wbg)
cv2.imwrite('gls_withoutbg_output.jpg',wobg)
self.imageoutput_2.setPixmap(QtGui.QPixmap("gls_withbg_output.jpg"))
self.imageoutput.setPixmap(QtGui.QPixmap("gls_withoutbg_output.jpg"))
```

```
if __name__ == "__main__":
import sys
app = QtWidgets.QApplication(sys.argv)
MainWindow = QtWidgets.QMainWindow()
ui = Ui_MainWindow()
ui.setupUi(MainWindow)
MainWindow.show()
sys.exit(app.exec_())
```

Output:

Python GUI Output:





Conclusion: Program to read an image and convert it into negative was written and executed successfully.

Deepraj Bhosale
Batch-A
181105016