Experiment No: 5

Aim: Write a Python program to read an image from a file system and perform the bit plane 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 a value for which bit position you want to display the image
4. Get the RGB value of the pixel
5. Now convert the decimal value to binary and extract the value at the position earlier inputted.
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:

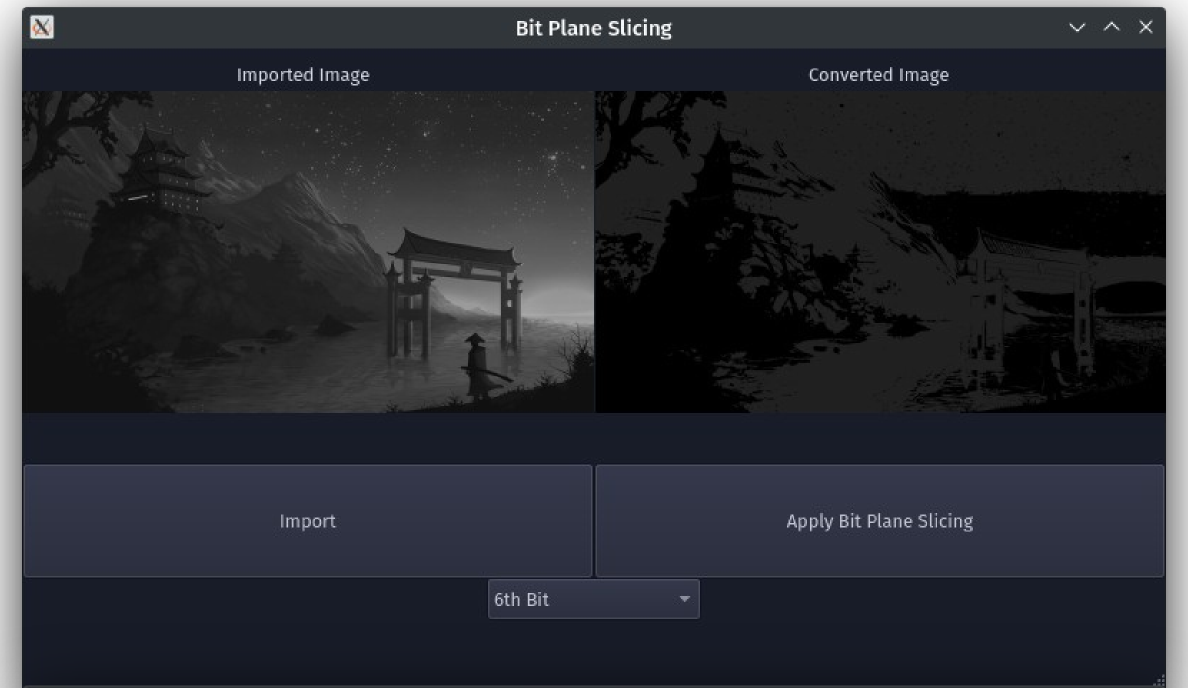
Python Code:

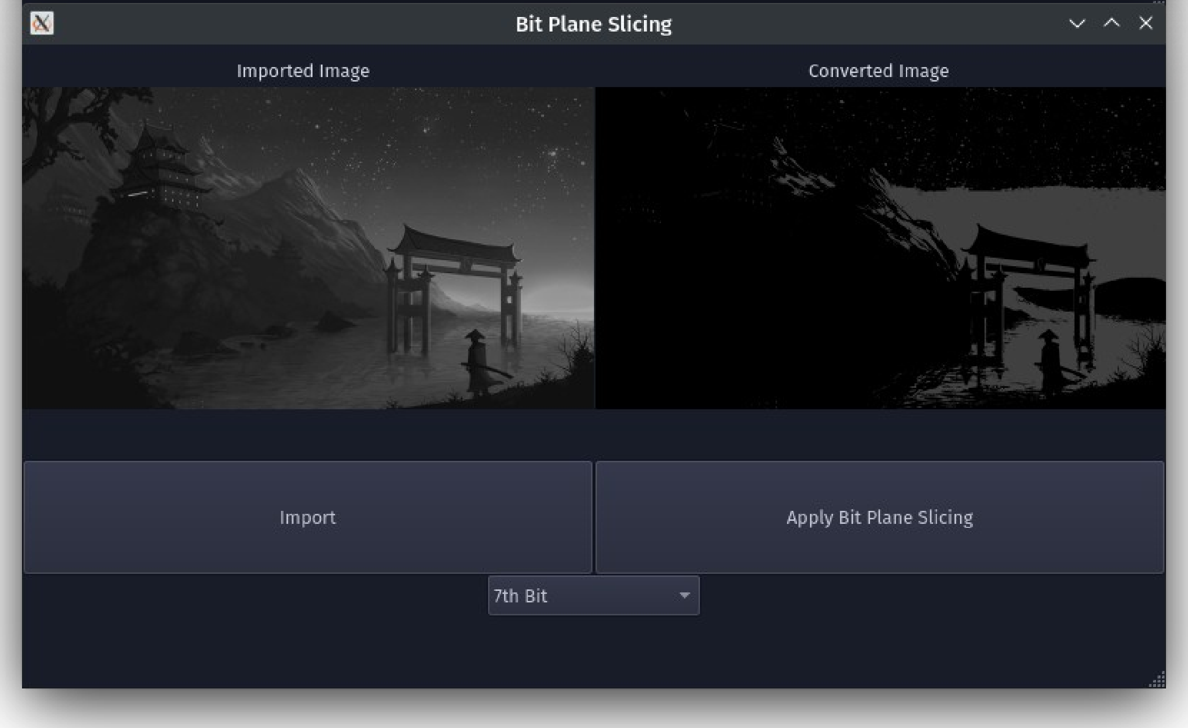
```
from PyQt5 import QtCore, QtGui, QtWidgets
import numpy as np
import cv2

class Ui_MainWindow(object):
def setupUi(self, MainWindow):
MainWindow.setObjectName("MainWindow")
MainWindow.resize(800, 450)
self.centralwidget = QtWidgets.QWidget(MainWindow)
self.centralwidget.setObjectName("centralwidget")
self.imageinput = QtWidgets.QLabel(self.centralwidget)
self.imageinput.setGeometry(QtCore.QRect(0, 30, 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, 290, 400, 81))
self.import_image.setObjectName("import_image")
self.output = QtWidgets.QPushButton(self.centralwidget)
self.output.setGeometry(QtCore.QRect(400, 290, 400, 81))
self.output.setObjectName("output")
self.imageoutput = QtWidgets.QLabel(self.centralwidget)
self.imageoutput.setGeometry(QtCore.QRect(401, 30, 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, 10, 101, 17))
self.label.setObjectName("label")
self.label_2 = QtWidgets.QLabel(self.centralwidget)
self.label_2.setGeometry(QtCore.QRect(550, 10, 101, 17))
self.label_2.setObjectName("label_2")
self.comboBox = QtWidgets.QComboBox(self.centralwidget)
self.comboBox.setGeometry(QtCore.QRect(325, 370, 150, 30))
self.comboBox.setObjectName("comboBox")
self.comboBox.addItems(["1st Bit", "2nd Bit", "3rd Bit", "4th Bit", "5th Bit", "6th Bit", "7th Bit", "8th Bit"])
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.bit_plane_slicing)

def retranslateUi(self, MainWindow):
_translate = QtCore.QCoreApplication.translate
MainWindow.setWindowTitle(_translate("MainWindow", "Bit Plane Slicing"))
self.import_image.setText(_translate("MainWindow", "Import"))
self.output.setText(_translate("MainWindow", "Apply Bit Plane Slicing"))
self.label.setText(_translate("MainWindow", "Imported Image"))
self.label_2.setText(_translate("MainWindow", "Converted Image"))

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 DecimalToBinary(self,num,width):
bnr = bin(num).replace('0b','')
x = bnr[::-1]
while len(x) < width:
x += '0'
bnr = x[::-1]
return bnr
def bit_to_img(self,array,bitselect):
img_con=[]
for i in array:
img_con.append(i[bitselect])
return img_con

def bit_plane_slicing(self):
(row, col) = self.img.shape[0:2]
lst = []
for i in range(row):
for j in range(col):
lst.append(np.binary_repr(self.img[i][j] ,width=8))
new_img=(np.array([int(i[7-self.comboBox.currentIndex()]) for i in lst],dtype = np.uint8) *
(2**(self.comboBox.currentIndex()))).reshape(self.img.shape[0],self.img.shape[1])
cv2.imwrite('bps_output.jpg',new_img)
self.imageoutput.setPixmap(QtGui.Qpixmap("bps_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 perform bit plane slicing on it was written and executed successfully.

Deepraj Bhosale
Batch-A
181105016