Praktikum II  
(Pengolahan Citra Digital)

(A3 – A7)

A3



A computer screen shot of a program code

Description automatically generated

A screen shot of a computer code

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a cartoon

Description automatically generated

A4

A computer screen shot of a program code

Description automatically generated

A screen shot of a computer code

Description automatically generated

A computer screen shot of a program code

Description automatically generated

A computer screen shot of a program code

Description automatically generated

A computer screen with text

Description automatically generated

**A screenshot of a cartoon

Description automatically generated**

**A screenshot of a cartoon

Description automatically generated**

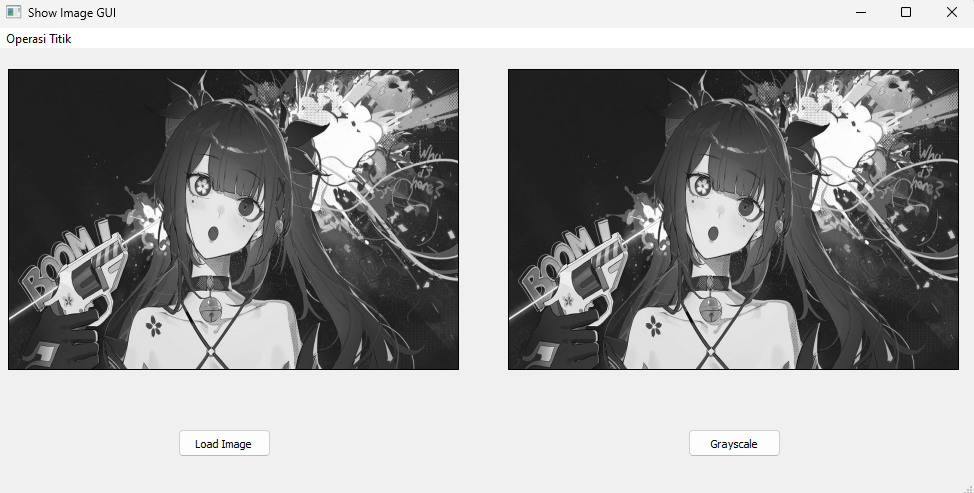
A5

import cv2  
import sys  
import numpy as np  
from PyQt5 import QtCore, QtWidgets  
from PyQt5.QtCore import pyqtSlot, Qt  
from PyQt5.QtGui import QImage, QPixmap  
from PyQt5.QtWidgets import QMainWindow, QMessageBox  
from PyQt5.uic import loadUi  
  
class ShowImage(QMainWindow):  
 def \_\_init\_\_(self):  
 super(ShowImage, self).\_\_init\_\_()  
 loadUi('showgui.ui', self)  
 self.image = None  
 self.loadButton.clicked.connect(self.loadClicked)  
 self.grayButton.clicked.connect(self.grayClicked)  
 self.actionOperasi\_Pencerahan.triggered.connect(self.brightness)  
 self.actionSimple\_Contrast.triggered.connect(self.contrast)  
  
 @pyqtSlot()  
 def loadClicked(self):  
 self.loadImage('BANG.jpeg')  
  
 def loadImage(self, flname):  
 self.image = cv2.imread(flname)  
 self.displayImage()  
  
 def displayImage(self):  
 qformat = QImage.Format\_Indexed8  
  
 if len(self.image.shape) == 3: # row[0], col[1], channel[2]  
 if self.image.shape[2] == 4:  
 qformat = QImage.Format\_RGBA8888  
 else:  
 qformat = QImage.Format\_RGB888  
  
 img = QImage(self.image.data, self.image.shape[1], self.image.shape[0], self.image.strides[0], qformat)  
 img = img.rgbSwapped()  
  
 self.imgLabel.setPixmap(QPixmap.fromImage(img))  
 self.imgLabel.setAlignment(Qt.AlignHCenter | Qt.AlignVCenter)  
  
 def grayClicked(self):  
 H, W = self.image.shape[:2]  
 gray = np.zeros((H, W), np.uint8)  
 for i in range(H):  
 for j in range(W):  
 gray[i, j] = np.clip(  
 0.299 \* self.image[i, j, 0] + 0.587 \* self.image[i, j, 1] + 0.114 \* self.image[i, j, 2], 0,  
 255)  
 self.image = gray  
 self.displayImage(2)  
  
 def brightness(self):  
 try:  
 self.image = cv2.cvtColor(self.image, cv2.COLOR\_BGR2GRAY)  
 except:  
 pass  
  
 H, W = self.image.shape[:2]  
 brightness = 50  
 for i in range(H):  
 for j in range(W):  
 a = self.image.item(i, j)  
 b = np.clip(a + brightness, 0, 255)  
 self.image.itemset((i, j), b)  
  
 self.displayImage(1)  
  
 def contrast(self):  
 try:  
 self.image = cv2.cvtColor(self.image, cv2.COLOR\_BGR2GRAY)  
 except:  
 pass  
  
 H, W = self.image.shape[:2]  
 contrast = 1.6  
 for i in range(H):  
 for j in range(W):  
 a = self.image.item(i, j)  
 b = np.clip(a \* contrast, 0, 255)  
 self.image.itemset((i, j), b)  
  
 self.displayImage(1)  
 def displayImage(self, window=1):  
 qformat = QImage.Format\_Indexed8  
 if len(self.image.shape) == 3:  
 if self.image.shape[2] == 4:  
 qformat = QImage.Format\_RGBA8888  
 else:  
 qformat = QImage.Format\_RGB888  
 img = QImage(self.image, self.image.shape[1], self.image.shape[0], self.image.strides[0], qformat)  
 img = img.rgbSwapped()  
  
 if window == 1:  
 self.imgLabel.setPixmap(QPixmap.fromImage(img))  
 self.imgLabel.setAlignment(QtCore.Qt.AlignHCenter | QtCore.Qt.AlignVCenter)  
 self.imgLabel.setScaledContents(True)  
  
 if window == 2:  
 self.hasilLabel.setPixmap(QPixmap.fromImage(img))  
 self.hasilLabel.setAlignment(QtCore.Qt.AlignHCenter | QtCore.Qt.AlignVCenter)  
 self.hasilLabel.setScaledContents(True)  
  
app = QtWidgets.QApplication(sys.argv)  
window = ShowImage()  
window.setWindowTitle('Show Image GUI')  
window.show()  
sys.exit(app.exec\_())

A screenshot of a cartoon

Description automatically generated

A6import cv2  
import sys  
import numpy as np  
from PyQt5 import QtCore, QtWidgets  
from PyQt5.QtCore import pyqtSlot, Qt  
from PyQt5.QtGui import QImage, QPixmap  
from PyQt5.QtWidgets import QMainWindow, QMessageBox  
from PyQt5.uic import loadUi  
  
class ShowImage(QMainWindow):  
 def \_\_init\_\_(self):  
 super(ShowImage, self).\_\_init\_\_()  
 loadUi('showgui.ui', self)  
 self.image = None  
 self.loadButton.clicked.connect(self.loadClicked)  
 self.grayButton.clicked.connect(self.grayClicked)  
 self.actionOperasi\_Pencerahan.triggered.connect(self.brightness)  
 self.actionSimple\_Contrast.triggered.connect(self.contrast)  
 self.actionContrast\_Streching.triggered.connect(self.contrastStreching)  
  
 @pyqtSlot()  
 def loadClicked(self):  
 self.loadImage('BANG.jpeg')  
  
 def loadImage(self, flname):  
 self.image = cv2.imread(flname)  
 self.displayImage()  
  
 def displayImage(self):  
 qformat = QImage.Format\_Indexed8  
  
 if len(self.image.shape) == 3: # row[0], col[1], channel[2]  
 if self.image.shape[2] == 4:  
 qformat = QImage.Format\_RGBA8888  
 else:  
 qformat = QImage.Format\_RGB888  
  
 img = QImage(self.image.data, self.image.shape[1], self.image.shape[0], self.image.strides[0], qformat)  
 img = img.rgbSwapped()  
  
 self.imgLabel.setPixmap(QPixmap.fromImage(img))  
 self.imgLabel.setAlignment(Qt.AlignHCenter | Qt.AlignVCenter)  
  
 def grayClicked(self):  
 H, W = self.image.shape[:2]  
 gray = np.zeros((H, W), np.uint8)  
 for i in range(H):  
 for j in range(W):  
 gray[i, j] = np.clip(  
 0.299 \* self.image[i, j, 0] + 0.587 \* self.image[i, j, 1] + 0.114 \* self.image[i, j, 2], 0,  
 255)  
 self.image = gray  
 self.displayImage(2)  
  
 def brightness(self):  
 try:  
 self.image = cv2.cvtColor(self.image, cv2.COLOR\_BGR2GRAY)  
 except:  
 pass  
  
 H, W = self.image.shape[:2]  
 brightness = 50  
 for i in range(H):  
 for j in range(W):  
 a = self.image.item(i, j)  
 b = np.clip(a + brightness, 0, 255)  
 self.image.itemset((i, j), b)  
  
 self.displayImage(1)  
  
 def contrast(self):  
 try:  
 self.image = cv2.cvtColor(self.image, cv2.COLOR\_BGR2GRAY)  
 except:  
 pass  
  
 H, W = self.image.shape[:2]  
 contrast = 1.6  
 for i in range(H):  
 for j in range(W):  
 a = self.image.item(i, j)  
 b = np.clip(a \* contrast, 0, 255)  
 self.image.itemset((i, j), b)  
  
 self.displayImage(1)  
  
 def contrastStreching(self):  
 try:  
 self.image = cv2.cvtColor(self.image, cv2.COLOR\_BGR2GRAY)  
 except:  
 pass  
  
 H, W = self.image.shape[:2]  
 minV = np.min(self.image)  
 maxV = np.max(self.image)  
  
 for i in range(H):  
 for j in range(W):  
 a = self.image.item(i, j)  
 b = float(a - minV) / (maxV - minV) \* 255  
 self.image.itemset((i, j), b)  
  
 self.displayImage(1)  
  
 def displayImage(self, window=1):  
 qformat = QImage.Format\_Indexed8  
 if len(self.image.shape) == 3:  
 if self.image.shape[2] == 4:  
 qformat = QImage.Format\_RGBA8888  
 else:  
 qformat = QImage.Format\_RGB888  
 img = QImage(self.image, self.image.shape[1], self.image.shape[0], self.image.strides[0], qformat)  
 img = img.rgbSwapped()  
  
 if window == 1:  
 self.imgLabel.setPixmap(QPixmap.fromImage(img))  
 self.imgLabel.setAlignment(QtCore.Qt.AlignHCenter | QtCore.Qt.AlignVCenter)  
 self.imgLabel.setScaledContents(True)  
  
 if window == 2:  
 self.hasilLabel.setPixmap(QPixmap.fromImage(img))  
 self.hasilLabel.setAlignment(QtCore.Qt.AlignHCenter | QtCore.Qt.AlignVCenter)  
 self.hasilLabel.setScaledContents(True)  
  
app = QtWidgets.QApplication(sys.argv)  
window = ShowImage()  
window.setWindowTitle('Show Image GUI')  
window.show()  
sys.exit(app.exec\_())



A7

import cv2  
import sys  
import numpy as np  
from PyQt5 import QtCore, QtWidgets  
from PyQt5.QtCore import pyqtSlot, Qt  
from PyQt5.QtGui import QImage, QPixmap  
from PyQt5.QtWidgets import QMainWindow, QMessageBox  
from PyQt5.uic import loadUi  
  
class ShowImage(QMainWindow):  
 def \_\_init\_\_(self):  
 super(ShowImage, self).\_\_init\_\_()  
 loadUi('showgui.ui', self)  
 self.image = None  
 self.loadButton.clicked.connect(self.loadClicked)  
 self.grayButton.clicked.connect(self.grayClicked)  
 self.actionOperasi\_Pencerahan.triggered.connect(self.brightness)  
 self.actionSimple\_Contrast.triggered.connect(self.contrast)  
 self.actionContrast\_Streching.triggered.connect(self.contrastStreching)  
 self.actionNegative\_Image.triggered.connect(self.negativeImage)  
  
 @pyqtSlot()  
 def loadClicked(self):  
 self.loadImage('BANG.JPEG')  
  
 def loadImage(self, flname):  
 self.image = cv2.imread(flname)  
 self.displayImage()  
  
 def displayImage(self):  
 qformat = QImage.Format\_Indexed8  
  
 if len(self.image.shape) == 3: # row[0], col[1], channel[2]  
 if self.image.shape[2] == 4:  
 qformat = QImage.Format\_RGBA8888  
 else:  
 qformat = QImage.Format\_RGB888  
  
 img = QImage(self.image.data, self.image.shape[1], self.image.shape[0], self.image.strides[0], qformat)  
 img = img.rgbSwapped()  
  
 self.imgLabel.setPixmap(QPixmap.fromImage(img))  
 self.imgLabel.setAlignment(Qt.AlignHCenter | Qt.AlignVCenter)  
  
 def grayClicked(self):  
 H, W = self.image.shape[:2]  
 gray = np.zeros((H, W), np.uint8)  
 for i in range(H):  
 for j in range(W):  
 gray[i, j] = np.clip(  
 0.299 \* self.image[i, j, 0] + 0.587 \* self.image[i, j, 1] + 0.114 \* self.image[i, j, 2], 0,  
 255)  
 self.image = gray  
 self.displayImage(2)  
  
 def brightness(self):  
 try:  
 self.image = cv2.cvtColor(self.image, cv2.COLOR\_BGR2GRAY)  
 except:  
 pass  
  
 H, W = self.image.shape[:2]  
 brightness = 50  
 for i in range(H):  
 for j in range(W):  
 a = self.image.item(i, j)  
 b = np.clip(a + brightness, 0, 255)  
 self.image.itemset((i, j), b)  
  
 self.displayImage(1)  
  
 def contrast(self):  
 try:  
 self.image = cv2.cvtColor(self.image, cv2.COLOR\_BGR2GRAY)  
 except:  
 pass  
  
 H, W = self.image.shape[:2]  
 contrast = 1.6  
 for i in range(H):  
 for j in range(W):  
 a = self.image.item(i, j)  
 b = np.clip(a \* contrast, 0, 255)  
 self.image.itemset((i, j), b)  
  
 self.displayImage(1)  
  
 def contrastStreching(self):  
 try:  
 self.image = cv2.cvtColor(self.image, cv2.COLOR\_BGR2GRAY)  
 except:  
 pass  
  
 H, W = self.image.shape[:2]  
 minV = np.min(self.image)  
 maxV = np.max(self.image)  
  
 for i in range(H):  
 for j in range(W):  
 a = self.image.item(i, j)  
 b = float(a - minV) / (maxV - minV) \* 255  
 self.image.itemset((i, j), b)  
  
 self.displayImage(1)  
  
 def negativeImage(self):  
 if self.image is not None:  
 negative\_img = 255 - self.image  
 self.image = negative\_img  
 self.displayImage()  
  
 def displayImage(self, window=1):  
 qformat = QImage.Format\_Indexed8  
 if len(self.image.shape) == 3:  
 if self.image.shape[2] == 4:  
 qformat = QImage.Format\_RGBA8888  
 else:  
 qformat = QImage.Format\_RGB888  
 img = QImage(self.image, self.image.shape[1], self.image.shape[0], self.image.strides[0], qformat)  
 img = img.rgbSwapped()  
  
 if window == 1:  
 self.imgLabel.setPixmap(QPixmap.fromImage(img))  
 self.imgLabel.setAlignment(QtCore.Qt.AlignHCenter | QtCore.Qt.AlignVCenter)  
 self.imgLabel.setScaledContents(True)  
  
 if window == 2:  
 self.hasilLabel.setPixmap(QPixmap.fromImage(img))  
 self.hasilLabel.setAlignment(QtCore.Qt.AlignHCenter | QtCore.Qt.AlignVCenter)  
 self.hasilLabel.setScaledContents(True)  
  
app = QtWidgets.QApplication(sys.argv)  
window = ShowImage()  
window.setWindowTitle('Show Image GUI')  
window.show()  
sys.exit(app.exec\_())

A screenshot of a cartoon

Description automatically generated