

Praktikum II

(Pengolahan Citra Digital)

(A3 – A7)

A3

```
1  ~ import cv2
2  import sys
3  import numpy as np
4  from PyQt5 import QtCore, QtWidgets
5  from PyQt5.QtCore import pyqtSlot, Qt
6  from PyQt5.QtGui import QImage, QPixmap
7  from PyQt5.QtWidgets import QMainWindow, QMessageBox
8  from PyQt5.uic import loadUi
9
10  2 usages
11  ~ class ShowImage(QMainWindow):
12  ~     def __init__(self):
13  ~         super(ShowImage, self).__init__()
14  ~         loadUi('showgui.ui', self)
15  ~         self.image = None
16  ~         self.loadButton.clicked.connect(self.loadClicked)
17  ~         self.grayButton.clicked.connect(self.grayClicked)
18
19  1 usage
20  @pyqtSlot()
21  def loadClicked(self):
22     self.loadImage('BANG.jpeg')
```

```
1 usage
22  def loadImage(self, filename):
23     self.image = cv2.imread(filename)
24     self.displayImage()
25
26  2 usages
27  def displayImage(self):
28     qformat = QImage.Format_Indexed8
29
30     if len(self.image.shape) == 3: # row[0], col[1], channel[2]
31         if self.image.shape[2] == 4:
32             qformat = QImage.Format_RGBA8888
33         else:
34             qformat = QImage.Format_RGB888
35
36     img = QImage(self.image.data, self.image.shape[1], self.image.shape[0], self.image.strides[0], qformat)
37     img = img.rgbSwapped()
38
39     self.imgLabel.setPixmap(QPixmap.fromImage(img))
40     self.imgLabel.setAlignment(Qt.AlignHCenter | Qt.AlignVCenter)
```

```

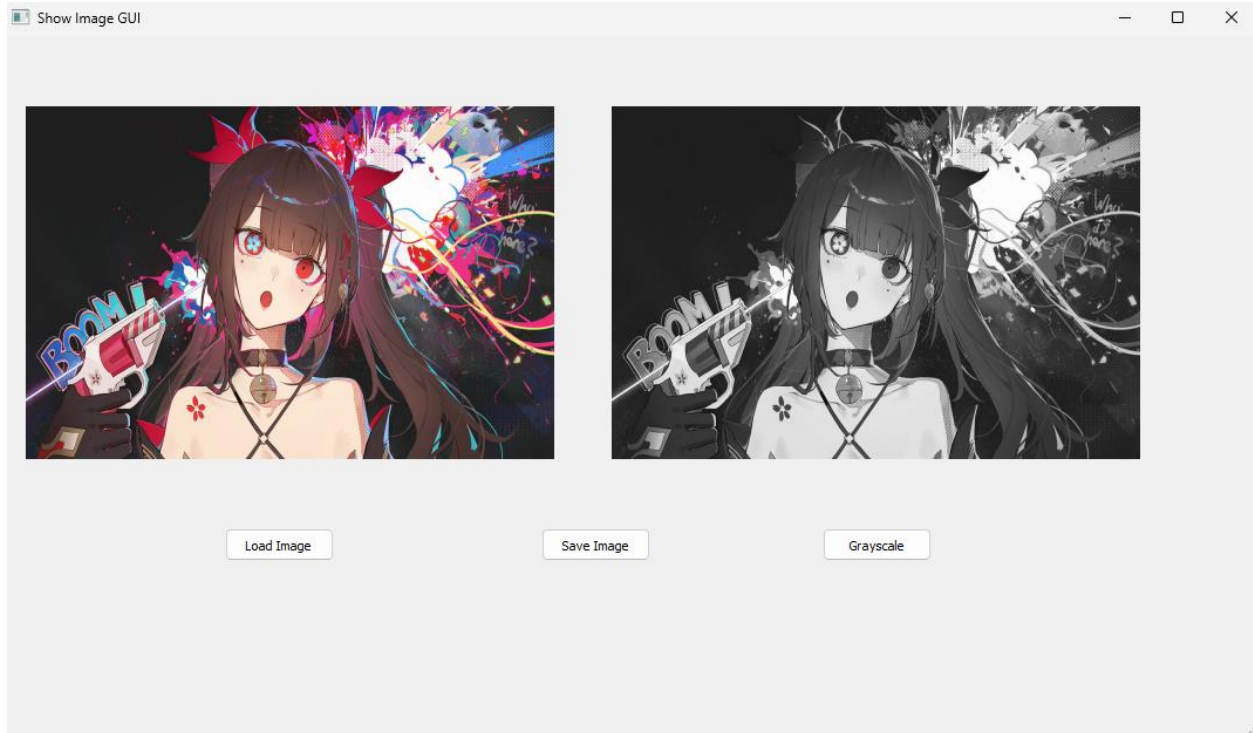
2 usages
41 def displayImage(self, window=1):
42     qformat = QImage.Format_Indexed8
43     if len(self.image.shape) == 3:
44         if self.image.shape[2] == 4:
45             qformat = QImage.Format_RGBA8888
46         else:
47             qformat = QImage.Format_RGB888
48     img = QImage(self.image, self.image.shape[1], self.image.shape[0], self.image.strides[0], qformat)
49     img = img.rgbSwapped()
50
51     if window == 1:
52         self.imgLabel.setPixmap(QPixmap.fromImage(img))
53         self.imgLabel.setAlignment(Qt.AlignHCenter | Qt.AlignVCenter)
54         self.imgLabel.setScaledContents(True)
55
56     if window == 2:
57         self.hasilLabel.setPixmap(QPixmap.fromImage(img))
58         self.hasilLabel.setAlignment(Qt.AlignHCenter | Qt.AlignVCenter)
59         self.hasilLabel.setScaledContents(True)
60

```

```

1 usage
61 @pyqtSlot()
62 def grayClicked(self):
63     try:
64         if self.image is not None:
65             H, W = self.image.shape[:2]
66             gray = np.zeros(shape=(H, W), np.uint8)
67             for i in range(H):
68                 for j in range(W):
69                     gray[i, j] = np.clip(
70                         0.299 * self.image[i, j, 0] + 0.587 * self.image[i, j, 1] + 0.114 * self.image[i, j, 2], a_min=0,
71                         a_max=255)
72             self.image = gray
73             self.displayImage(2)
74     except Exception as e:
75         QMessageBox.critical(self, "Error", str(e))
76
77
78
79 app = QtWidgets.QApplication(sys.argv)
80 window = ShowImage()
81 window.setWindowTitle('Show Image GUI')
82 window.show()
83 sys.exit(app.exec_())

```



A4

```
1 import cv2
2 import sys
3 import numpy as np
4 from PyQt5 import QtCore, QtWidgets
5 from PyQt5.QtCore import pyqtSlot, Qt
6 from PyQt5.QtGui import QImage, QPixmap
7 from PyQt5.QtWidgets import QMainWindow, QMessageBox
8 from PyQt5.uic import loadUi
9
10 2 usages
11 class ShowImage(QMainWindow):
12     def __init__(self):
13         super(ShowImage, self).__init__()
14         loadUi('uifile: 'showgui.ui', self)
15         self.image = None
16         self.loadButton.clicked.connect(self.loadClicked)
17         self.grayButton.clicked.connect(self.grayClicked)
18         self.actionOperasi_Pencerahan.triggered.connect(self.brightness)
19
20 1 usage
21 @pyqtSlot()
22 def loadClicked(self):
23     self.loadImage('BANG.jpeg')
```

```

1 usage
23  def loadImage(self, fname):
24      self.image = cv2.imread(fname)
25      self.displayImage()
26
27  3 usages
28  def displayImage(self):
29      qformat = QImage.Format_Indexed8
30
31      if len(self.image.shape) == 3: # row[0], col[1], channel[2]
32          if self.image.shape[2] == 4:
33              qformat = QImage.Format_RGBA8888
34          else:
35              qformat = QImage.Format_RGB888
36
37      img = QImage(self.image.data, self.image.shape[1], self.image.shape[0], self.image.strides[0], qformat)
38      img = img.rgbSwapped()
39
40      self.imgLabel.setPixmap(QPixmap.fromImage(img))
41      self.imgLabel.setAlignment(Qt.AlignHCenter | Qt.AlignVCenter)

```

```

1 usage
42  def grayClicked(self):
43      H, W = self.image.shape[:2]
44      gray = np.zeros(shape=(H, W), np.uint8)
45      for i in range(H):
46          for j in range(W):
47              gray[i, j] = np.clip(
48                  0.299 * self.image[i, j, 0] + 0.587 * self.image[i, j, 1] + 0.114 * self.image[i, j, 2], a_min=0,
49                  a_max=255)
50      self.image = gray
51      self.displayImage(2)
52
53  1 usage
54  def brightness(self):
55      try:
56          self.image = cv2.cvtColor(self.image, cv2.COLOR_BGR2GRAY)
57      except:
58          pass

```

```

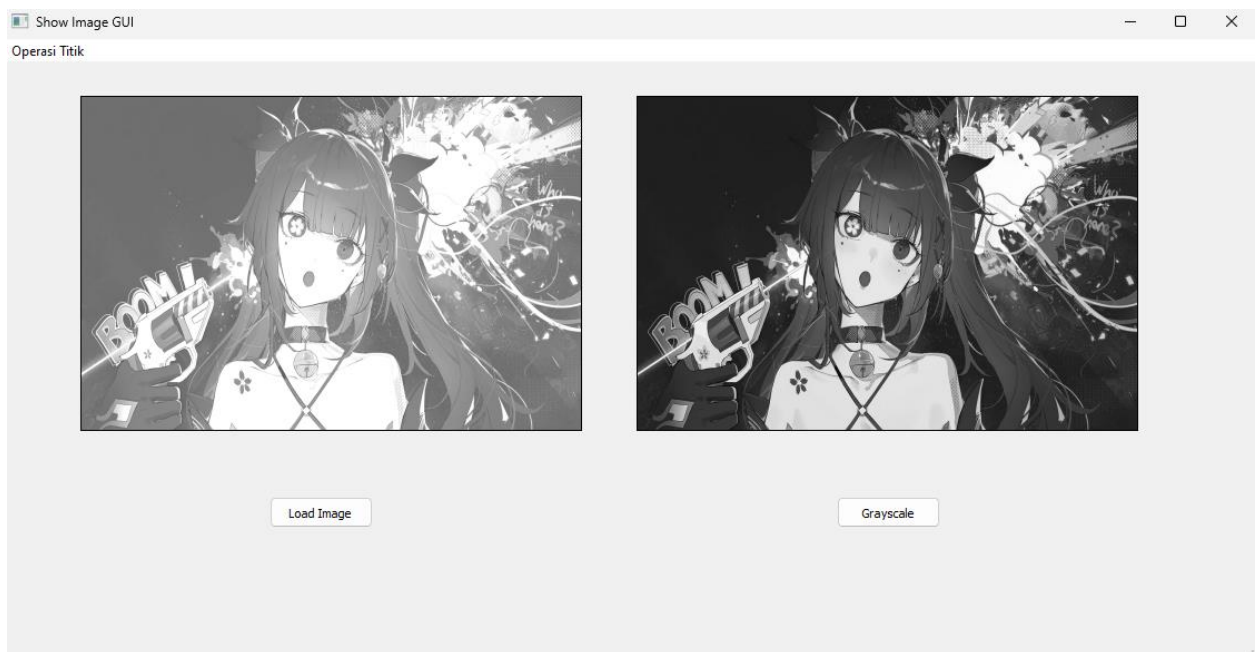
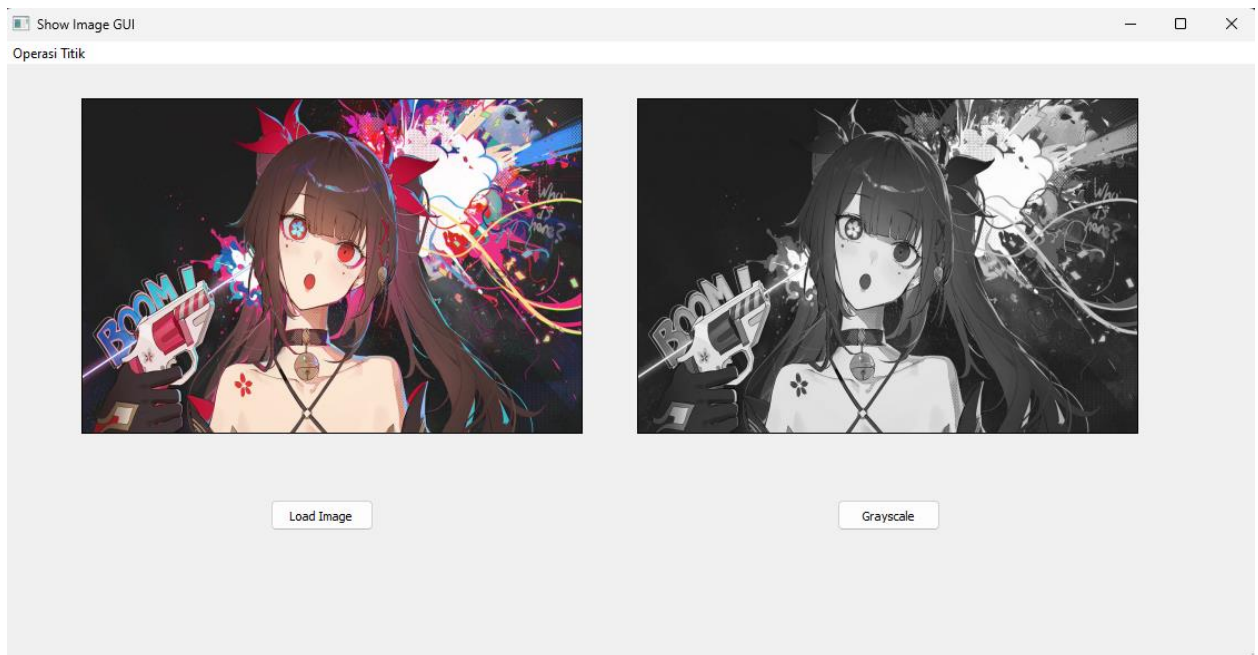
58
59     H, W = self.image.shape[:2]
60     brightness = 80
61     for i in range(H):
62         for j in range(W):
63             a = self.image.item(i, j)
64             b = np.clip(a + brightness, a_min: 0, a_max: 255)
65             self.image.itemset((i, j), b)
66
67     self.displayImage(1)
68
69     3 usages
70     def displayImage(self, window=1):
71         qformat = QImage.Format_Indexed8
72         if len(self.image.shape) == 3:
73             if self.image.shape[2] == 4:
74                 qformat = QImage.Format_RGBA8888
75             else:
76                 qformat = QImage.Format_RGB888
77         img = QImage(self.image, self.image.shape[1], self.image.shape[0], self.image.strides[0], qformat)
78         img = img.rgbSwapped()

```

```

78
79     if window == 1:
80         self.imgLabel.setPixmap(QPixmap.fromImage(img))
81         self.imgLabel.setAlignment(Qt.AlignHCenter | Qt.AlignVCenter)
82         self.imgLabel.setScaledContents(True)
83
84     if window == 2:
85         self.hasilLabel.setPixmap(QPixmap.fromImage(img))
86         self.hasilLabel.setAlignment(Qt.AlignHCenter | Qt.AlignVCenter)
87         self.hasilLabel.setScaledContents(True)
88
89     app = QtWidgets.QApplication(sys.argv)
90     window = ShowImage()
91     window.setWindowTitle('Show Image GUI')
92     window.show()
93     sys.exit(app.exec_())

```



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()

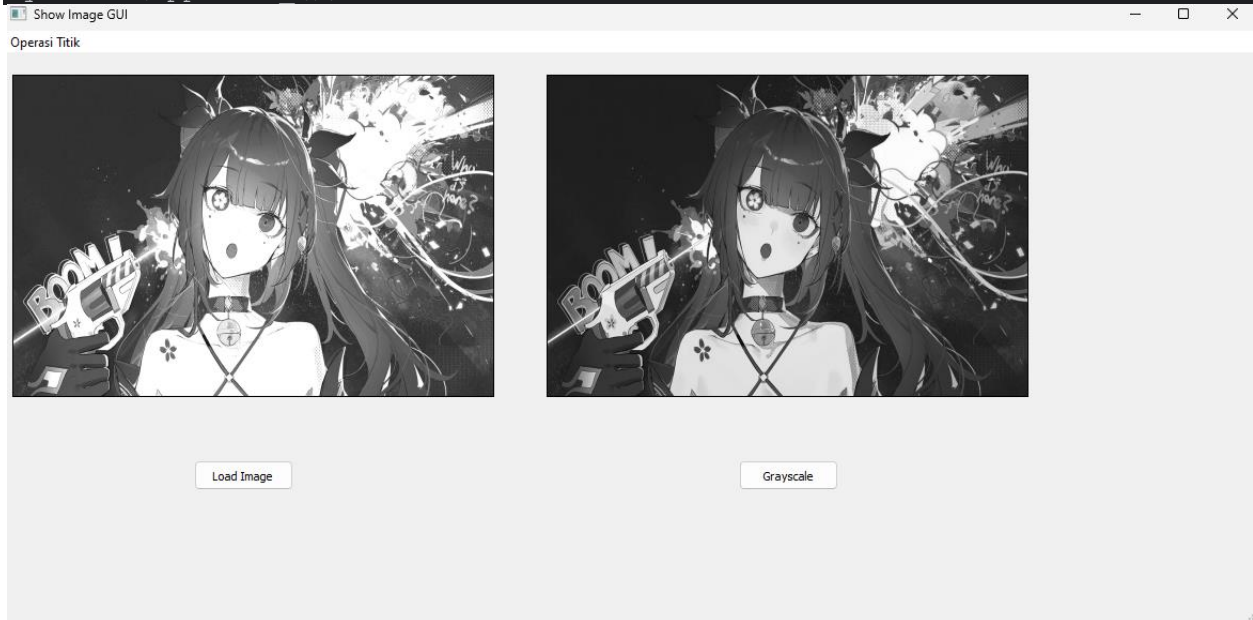
```



```

window.setWindowTitle('Show Image GUI')
window.show()
sys.exit(app.exec_())

```



```

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_Strechng.triggered.connect(self.contrastStrechng)

    @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)

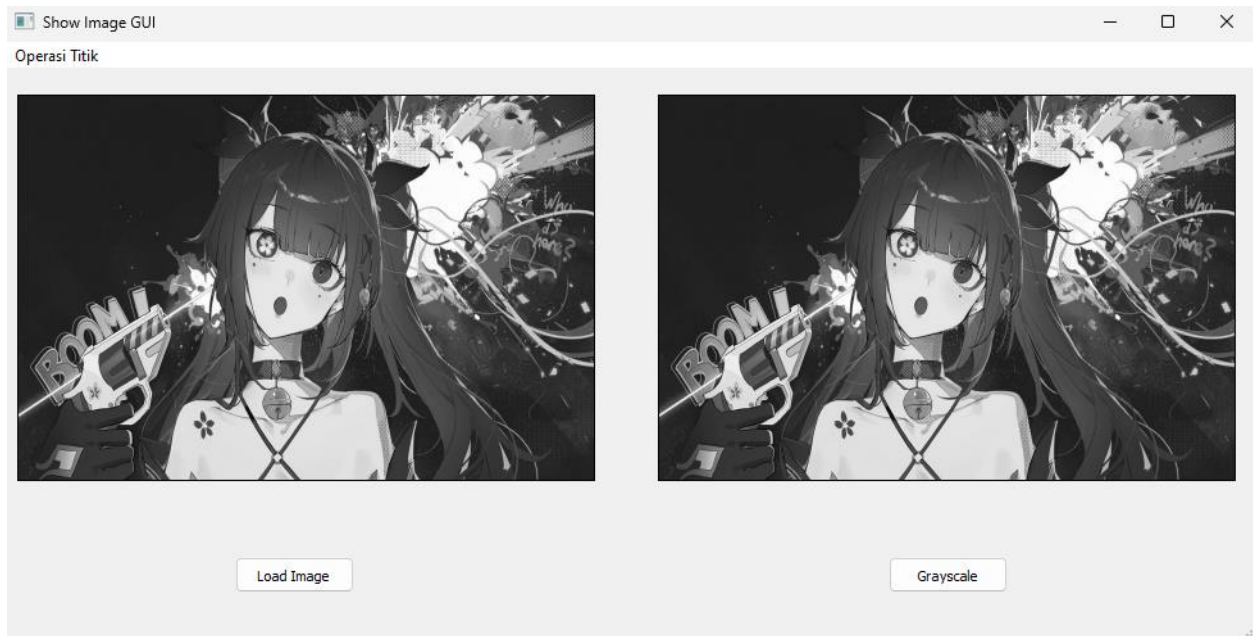
    def displayImage(self, window=1):
        qformat = QImage.Format_Indexed8
        if len(self.image.shape) == 3:
            if self.image.shape[2] == 4:
                qformat = QImage.Format_RGBA8888
            else:
                qformat = QImage.Format_RGB888
        img = QImage(self.image, 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_Strechging.triggered.connect(self.contrastStrechging)
        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_())

```

