《数字图像处理》实验报告

时间: 2021 年 11 月 16 日（第 12 周，星期 二 ）

地点： 测绘工程实验室（B 区）-- B 实验大楼 202

学生姓名： 魏子继 学号： 20194947

一、 实验名称：

PyQt 图形化 GUI 界面设计

二、 实验目的：

设计一个 GUI 界面，要求这个界面简洁、美好、功能齐全、显示齐全，能够实现数字图像处理的全部功能。

界面预期实现打开和保存图像；能够在显示框内放大缩小或全屏显示图像；根据菜单选项实现灰度化全部功能；根据菜单选项实现二值化全部功能，其中点击二值化相关功能的菜单栏时会弹出一个对话框，对话框中有滑动条和显示框可以控制阈值大小，从而动态调节阈值查看二值化变化的过程；点击大津算法菜单栏时能够在显示图像的同时显示最适阈值的大小和可分离测度；点击 Canny 算法菜单栏时弹出对话框，对话框内有调节双阈值大小的滑动条和显示框；实现输出图像灰度图像直方统计图和彩色图像三通道统计图；未尽功能仍待完善。

同时希望该界面在设计完成后能够保存为.exe 为后缀的应用程序，使得没有安装 PyQt 的应用者同样能够使用这个数字图

像处理应用界面实现相关的数字图像处理功能。在做到这一步的同时，要求在界面功能完善的同时能够添加很多界面显示美化的代码，尽力做到界面的设计和当今时代潮流相契合，供使用者使用时不会感到使用的是一个上个世纪的应用程序。

目前界面能够实现图像的打开和保存、根据菜单选项灰度化、根据菜单选项二值化和大津算法、滑动条调节阈值大小动态查看二值化效果、滑动条旁有一显示框随滑动条数值改变并且可以一同修改阈值大小的功能，其中滑动条在界面上显示，还未呈现在对话框中。

三、 实验步骤：

1. 使用 PyQt 设计图像化界面，QtDesign 设计得到 ui 格式文件，PyUIC 解码转为 py 格式文件
2. 设置菜单完成数字图像处理功能的选项；设置按钮实现图像的放大缩小；设置 GraphicsViewer 显示图像；设置滑动条实现横向控制阈值大小并观察不同阈值对图像处理产生的影响；设置对话框弹出能够实现 Canny 算法的双阈值调节。
3. 总的目的是使图像处理简洁到不用看使用说明便能够理解每个控件或是按钮的含义，简单到非常方便地得到图像处理的结果。
4. 功能完善后，实现界面美化设计代码，要求界面美观、简洁。
5. 将界面转为.exe 应用程序，供使用者能够安装使用。

四、 实验中的关键点分析（包括关键算法与代码实现）：

1. 主程序（main.py）

```
"""
数字图像处理 GUI 界面主入口
"""

import sys
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
import cv2.cv2 as cv2

import interface
import function


class MainCode(QtWidgets.QMainWindow, interface.Ui_MainWindow):
    def __init__(self):
        QtWidgets.QMainWindow.__init__(self)
        interface.Ui_MainWindow.__init__(self)
        self.setupUi(self)

        # 按钮触发事件
        self.actionOpen.triggered.connect(self.openImage)
        self.actionSave.triggered.connect(self.saveImage)
        self.actionOpencvGray.triggered.connect(self.grayOpencv)
        self.actionMax.triggered.connect(self.grayMax)
        self.actionAverage.triggered.connect(self.grayAverage)
        self.actionGamma.triggered.connect(self.grayGamma)

        self.actionSelf.triggered.connect(self.binarySelf)
        self.actionSelfInv.triggered.connect(self.binarySelfInv)
        self.actionOTSU.triggered.connect(self.binaryOTSU)

        # binary 图像二值化的滑动条与按钮相接
        if self.actionOpencvBinary.triggered.connect(self.binaryOpencv):

            self.horizontalSliderBinary.valueChanged.connect(self.binaryO
pencv)
        elif self.actionSelf.triggered.connect(self.binarySelf):
            self.horizontalSliderBinary.valueChanged.connect(self.binaryS
elf)
```

```
39.         elif self.actionSelfInv.triggered.connect(self.binarySelfInv):
40.             self.horizontalSliderBinary.valueChanged.connect(self.binaryS
    elfInv)
41.
42.         # binary 图像二值化的滑动条与显示框相连
43.         self.spinBoxBinary.valueChanged.connect(self.spinBoxChange)
44.         self.horizontalSliderBinary.valueChanged.connect(self.sliderChang
    e)
45.
46.         # 按钮连接
47.         self.pushButtonLargeOriginal.clicked.connect(self.graphicsLarge)


48.
49.     # 打开图像--open
50.     def openImage(self):
51.         global imgName  # 设置为全局变量
52.         imgName, imgType = QtWidgets.QFileDialog.getOpenFileName(self.cen
    tralwidget, "打开图片", "",
53.                                                                  "All Fil
    es(*);;*.jpg;;*.png")
54.         img = QtGui.QPixmap(imgName)  # 通过文件路径获取图片文件
55.         item = QtWidgets.QGraphicsPixmapItem(img)
56.         scene = QtWidgets.QGraphicsScene()
57.         scene.addItem(item)
58.         self.graphicsViewOriginal.setScene(scene)  # 在 graphicsView 控件上
    显示选择的图片
59.
60.     # 保存图像--save
61.     def saveImage(self):
62.         screen = QtWidgets.QApplication.primaryScreen()
63.         pix = screen.grabWindow(self.graphicsViewResult.winId())
64.         fd, filetype = QtWidgets.QFileDialog.getSaveFileName(self.central
    widget, "保存图片", "",
65.                                                              "*.png;;*.jp
    g;;All Files(*)")
66.         pix.save(fd)
67.
68.     # 使图像显示在 graphicsViewResult 中
69.     def graphicsView(self, img):
70.         img = QtGui.QImage(img.data, img.shape[1], img.shape[0],
71.                            img.shape[1] * img.shape[2],
72.                            QtGui.QImage.Format_RGB888)
73.         img = QtGui.QPixmap(img)
74.         item = QtWidgets.QGraphicsPixmapItem(img)
```

```
75.         scene = QtWidgets.QGraphicsScene()
76.         scene.addItem(item)
77.         self.graphicsViewResult.setScene(scene)
78.
79.     # 灰度化 opencv--opencvGray
80.     def grayOpencv(self):
81.         gray = function.gray_function(imgName)
82.         self.graphicsView(gray)
83.
84.     # 灰度化最大值--Max
85.     def grayMax(self):
86.         gray = function.gray_Max(imgName)
87.         self.graphicsView(gray)
88.
89.     # 灰度化平均值--Average
90.     def grayAverage(self):
91.         gray = function.gray_Average(imgName)
92.         self.graphicsView(gray)
93.
94.     # 灰度化伽马校正--Gamma
95.     def grayGamma(self):
96.         gray = function.gray_Gamma(imgName)
97.         self.graphicsView(gray)
98.
99.     # 滑动条改变时，显示框数值也随之改变
100.     def sliderChange(self):
101.         self.spinBoxBinary.setValue(self.horizontalSliderBinary.value())
102.
103.     # 显示框改变时，滑动条数值也随之改变
104.     def spinBoxChange(self):
105.         self.horizontalSliderBinary.setValue(self.spinBoxBinary.value())
106.
107.     # 二值化 opencv--opencvBinary
108.     def binaryOpencv(self):
109.         threshold = int(self.horizontalSliderBinary.value())
110.         binary = function.threshold_Ref(imgName, threshold)
111.         self.graphicsView(binary)
112.
113.     # 二值化 self--self
114.     def binarySelf(self):
115.         threshold = int(self.horizontalSliderBinary.value())
116.         binary = function.threshold_self(imgName, threshold)
```

```
117.         self.graphicsView(binary)
118.
119.     # 反二值化 self--selfInv
120.     def binarySelfInv(self):
121.         threshold = int(self.horizontalSliderBinary.value())
122.         binary = function.threshold_self_inv(imgName, threshold)
123.         self.graphicsView(binary)
124.
125.     # 二值化大津算法--OTSU
126.     def binaryOTSU(self):
127.         otsu = function.otsu(imgName)
128.         self.graphicsView(otsu)
129.
130.     # 边界提取 Canny--Canny
131.     def edgeDetectionCanny(self):
132.         return 0
133.
134.     # 边界提取 CannyOpencv
135.
136.     # 图像的放大
137.     def graphicsLarge(self):
138.         img = cv2.imread(imgName)
139.         img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
140.
141.         # 图像放大
142.         imgScale = 1
143.         imgScale = imgScale + 0.05
144.         if imgScale >= 1.2:
145.             imgScale = 1.2
146.
147.         # 修改图像规模并展示
148.         Large = QtGui.QImage(img, img.shape[1], img.shape[0],
149.                              QtGui.QImage.Format_RGB888)
150.         Large = QtGui.QPixmap(Large)
151.         item = QtWidgets.QGraphicsPixmapItem(Large)
152.         item.setScale(imgScale)
153.         scene = QtWidgets.QGraphicsScene()
154.         scene.addItem(item)
155.         self.graphicsViewOriginal.setScene(scene)
156.
157.
158. if __name__ == '__main__':
159.     app = QApplication(sys.argv)
160.     MainWindow = MainCode()
```

```
    MainWindow.show()
    sys.exit(app.exec_())
```

2. 界面设计文件（interface.py）

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'interface.ui'
#
# Created by: PyQt5 UI code generator 5.15.4
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1126, 860)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.pushButtonLargeOriginal = QtWidgets.QPushButton(self.centralwidget)
        self.pushButtonLargeOriginal.setGeometry(QtCore.QRect(80, 580, 141, 61))
        self.pushButtonLargeOriginal.setObjectName("pushButtonLargeOriginal")
        self.pushButtonShrinkOriginal = QtWidgets.QPushButton(self.centralwidget)
        self.pushButtonShrinkOriginal.setGeometry(QtCore.QRect(330, 580, 141, 61))
        self.pushButtonShrinkOriginal.setObjectName("pushButtonShrinkOriginal")
        self.graphicsViewOriginal = QtWidgets.QGraphicsView(self.centralwidget)
        self.graphicsViewOriginal.setGeometry(QtCore.QRect(20, 80, 521, 461))
        self.graphicsViewOriginal.setObjectName("graphicsViewOriginal")
```

```
29.         self.graphicsViewResult = QtWidgets.QGraphicsView(self.centralwid
    get)
30.         self.graphicsViewResult.setGeometry(QtCore.QRect(580, 80, 521, 46
    1))
31.         self.graphicsViewResult.setObjectName("graphicsViewResult")
32.         self.pushButtonLargeResult = QtWidgets.QPushButton(self.centralwi
    dget)
33.         self.pushButtonLargeResult.setGeometry(QtCore.QRect(650, 580, 131
    , 61))
34.         self.pushButtonLargeResult.setObjectName("pushButtonLargeResult")

35.         self.pushButtonShrinkResult = QtWidgets.QPushButton(self.centralw
    idget)
36.         self.pushButtonShrinkResult.setGeometry(QtCore.QRect(880, 580, 14
    1, 61))
37.         self.pushButtonShrinkResult.setObjectName("pushButtonShrinkResult
    ")
38.         self.label = QtWidgets.QLabel(self.centralwidget)
39.         self.label.setGeometry(QtCore.QRect(190, 20, 121, 41))
40.         font = QtGui.QFont()
41.         font.setPointSize(16)
42.         self.label.setFont(font)
43.         self.label.setAlignment(QtCore.Qt.AlignCenter)
44.         self.label.setObjectName("label")
45.         self.label_2 = QtWidgets.QLabel(self.centralwidget)
46.         self.label_2.setGeometry(QtCore.QRect(760, 20, 121, 41))
47.         font = QtGui.QFont()
48.         font.setPointSize(16)
49.         self.label_2.setFont(font)
50.         self.label_2.setAlignment(QtCore.Qt.AlignCenter)
51.         self.label_2.setObjectName("label_2")
52.         self.label_3 = QtWidgets.QLabel(self.centralwidget)
53.         self.label_3.setGeometry(QtCore.QRect(260, 690, 101, 31))
54.         font = QtGui.QFont()
55.         font.setPointSize(12)
56.         self.label_3.setFont(font)
57.         self.label_3.setAlignment(QtCore.Qt.AlignCenter)
58.         self.label_3.setObjectName("label_3")
59.         self.horizontalSliderBinary = QtWidgets.QSlider(self.centralwidge
    t)
60.         self.horizontalSliderBinary.setGeometry(QtCore.QRect(490, 680, 32
    1, 51))
61.         self.horizontalSliderBinary.setMaximum(255)
62.         self.horizontalSliderBinary.setProperty("value", 127)
```

```
63.         self.horizontalSliderBinary.setOrientation(QtCore.Qt.Horizontal)
64.         self.horizontalSliderBinary.setTickPosition(QtWidgets.QSlider.TicksBelow)
65.         self.horizontalSliderBinary.setTickInterval(32)
66.         self.horizontalSliderBinary.setObjectName("horizontalSliderBinary")
67.         self.spinBoxBinary = QtWidgets.QSpinBox(self.centralwidget)
68.         self.spinBoxBinary.setGeometry(QtCore.QRect(380, 690, 81, 31))
69.         self.spinBoxBinary.setMaximum(255)
70.         self.spinBoxBinary.setProperty("value", 127)
71.         self.spinBoxBinary.setObjectName("spinBoxBinary")
72.         self.labelBinary = QtWidgets.QLabel(self.centralwidget)
73.         self.labelBinary.setGeometry(QtCore.QRect(410, 750, 281, 51))
74.         font = QtGui.QFont()
75.         font.setPointSize(12)
76.         self.labelBinary.setFont(font)
77.         self.labelBinary.setObjectName("labelBinary")
78.         MainWindow.setCentralWidget(self.centralwidget)
79.         self.menubar = QtWidgets.QMenuBar(MainWindow)
80.         self.menubar.setEnabled(True)
81.         self.menubar.setGeometry(QtCore.QRect(0, 0, 1126, 26))
82.         self.menubar.setDefaultUp(False)
83.         self.menubar.setNativeMenuBar(True)
84.         self.menubar.setObjectName("menubar")
85.         self.menu = QtWidgets.QMenu(self.menubar)
86.         self.menu.setObjectName("menu")
87.         self.menu_2 = QtWidgets.QMenu(self.menubar)
88.         self.menu_2.setObjectName("menu_2")
89.         self.menu_3 = QtWidgets.QMenu(self.menubar)
90.         self.menu_3.setObjectName("menu_3")
91.         self.menu_4 = QtWidgets.QMenu(self.menubar)
92.         self.menu_4.setObjectName("menu_4")
93.         MainWindow.setMenuBar(self.menubar)
94.         self.statusbar = QtWidgets.QStatusBar(MainWindow)
95.         self.statusbar.setObjectName("statusbar")
96.         MainWindow.setStatusBar(self.statusbar)
97.         self.actionOpen = QtWidgets.QAction(MainWindow)
98.         self.actionOpen.setObjectName("actionOpen")
99.         self.actionSave = QtWidgets.QAction(MainWindow)
100.            self.actionSave.setObjectName("actionSave")
101.            self.actionOpencvGray = QtWidgets.QAction(MainWindow)
102.            self.actionOpencvGray.setObjectName("actionOpencvGray")
103.            self.actionMax = QtWidgets.QAction(MainWindow)
```

```
104.             self.actionMax.setObjectName("actionMax")
105.             self.actionAverage = QtWidgets.QAction(MainWindow)
106.             self.actionAverage.setObjectName("actionAverage")
107.             self.actionGamma = QtWidgets.QAction(MainWindow)
108.             self.actionGamma.setObjectName("actionGamma")
109.             self.actionOpencvBinary = QtWidgets.QAction(MainWindow)
110.             self.actionOpencvBinary.setObjectName("actionOpencvBinary")
111.             self.actionSelf = QtWidgets.QAction(MainWindow)
112.             self.actionSelf.setObjectName("actionSelf")
113.             self.actionSelfInv = QtWidgets.QAction(MainWindow)
114.             self.actionSelfInv.setObjectName("actionSelfInv")
115.             self.actionOTSU = QtWidgets.QAction(MainWindow)
116.             self.actionOTSU.setObjectName("actionOTSU")
117.             self.actionCanny = QtWidgets.QAction(MainWindow)
118.             self.actionCanny.setObjectName("actionCanny")
119.             self.menu.addAction(self.actionOpen)
120.             self.menu.addAction(self.actionSave)
121.             self.menu_2.addAction(self.actionOpencvGray)
122.             self.menu_2.addAction(self.actionMax)
123.             self.menu_2.addAction(self.actionAverage)
124.             self.menu_2.addAction(self.actionGamma)
125.             self.menu_3.addAction(self.actionOpencvBinary)
126.             self.menu_3.addAction(self.actionSelf)
127.             self.menu_3.addAction(self.actionSelfInv)
128.             self.menu_3.addSeparator()
129.             self.menu_3.addAction(self.actionOTSU)
130.             self.menu_4.addAction(self.actionCanny)
131.             self.menubar.addAction(self.menu.menuAction())
132.             self.menubar.addAction(self.menu_2.menuAction())
133.             self.menubar.addAction(self.menu_3.menuAction())
134.             self.menubar.addAction(self.menu_4.menuAction())
135. 
136.             self.retranslateUi(MainWindow)
137.             QtCore.QMetaObject.connectSlotsByName(MainWindow)
138. 
139.         def retranslateUi(self, MainWindow):
140.             _translate = QtCore.QCoreApplication.translate
141.             MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow
    "))
142.             self.pushButtonLargeOriginal.setText(_translate("MainWindow",
    "放大"))
143.             self.pushButtonShrinkOriginal.setText(_translate("MainWindow",
     "缩小"))
```

```
144.             self.pushButtonLargeResult.setText(_translate("MainWindow", "
    放大"))
145.             self.pushButtonShrinkResult.setText(_translate("MainWindow", "
    缩小"))
146.             self.label.setText(_translate("MainWindow", "原图"))
147.             self.label_2.setText(_translate("MainWindow", "结果图"))
148.             self.label_3.setText(_translate("MainWindow", "调整阈值："))
149.             self.labelBinary.setText(_translate("MainWindow", "当前阈值：
    "))
150.             self.menu.setTitle(_translate("MainWindow", "文件"))
151.             self.menu_2.setTitle(_translate("MainWindow", "灰度化"))
152.             self.menu_3.setTitle(_translate("MainWindow", "二值化"))
153.             self.menu_4.setTitle(_translate("MainWindow", "边缘提取"))
154.             self.actionOpen.setText(_translate("MainWindow", "open"))
155.             self.actionSave.setText(_translate("MainWindow", "save"))
156.             self.actionOpencvGray.setText(_translate("MainWindow", "opencv
    "))
157.             self.actionMax.setText(_translate("MainWindow", "Max"))
158.             self.actionAverage.setText(_translate("MainWindow", "Average")
    )
159.             self.actionGamma.setText(_translate("MainWindow", "Gamma"))
160.             self.actionOpencvBinary.setText(_translate("MainWindow", "open
    cv"))
161.             self.actionSelf.setText(_translate("MainWindow", "self"))
162.             self.actionSelfInv.setText(_translate("MainWindow", "selfinv")
    )
163.             self.actionOTSU.setText(_translate("MainWindow", "otsu"))
164.             self.actionCanny.setText(_translate("MainWindow", "Canny"))
```

3. 图像界面控件涉及相关数字图像处理函数（function.py）

```
1.  """
2.  方便起见，将其他函数全部导入其中
3.  下列函数与原 py 函数做出了简要修改，主要为 cv2 可以读取，qt 可以输出
4.  """
5.
6.  import cv2.cv2 as cv2
7.  import numpy as np
8.
9.  """
10. 灰度化
11. """
12.
```

```
13.
14. # opencv 内置函数灰度化，即加权平均
15. def gray_function(img):
16.     img = cv2.imread(img)
17.     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  # COLOR_BGR2GRAY 参数表示
    BGR 图像转化为灰度图像
18.     gray = cv2.cvtColor(gray, cv2.COLOR_BGR2RGB)  # COLOR_BGR2RGB 参数表示
    BGR 图像转化为 RGB 图像以在 plt 中显示
19.     return gray
20.
21.
22. # 使用最大值灰度化
23. def gray_Max(img):
24.     img = cv2.imread(img)
25.     height, width = img.shape[0:2]
26.     gray = np.zeros((height, width), img.dtype)  # 设置一个新的图像存放灰度
    图像，防止对原图进行修改
27.     for i in range(height):
28.         for j in range(width):
29.             gray[i, j] = max(img[i, j, 0], img[i, j, 1], img[i, j, 2])  #
     最大值设置为灰度
30.     gray = cv2.cvtColor(gray, cv2.COLOR_BGR2RGB)
31.     return gray
32.
33.
34. # 使用平均值灰度化
35. def gray_Average(img):
36.     img = cv2.imread(img)
37.     height, width = img.shape[0:2]
38.     gray = np.zeros((height, width), img.dtype)
39.     for i in range(height):
40.         for j in range(width):
41.             gray[i, j] = (int(img[i, j, 0]) + int(img[i, j, 1]) + int(img
    [i, j, 2])) / 3  # 平均值作为灰度
42.     gray = cv2.cvtColor(gray, cv2.COLOR_BGR2RGB)
43.     return gray
44.
45.
46. # 使用 Gamma 校正进行灰度处理
47. def gray_Gamma(img):
48.     img = cv2.imread(img)
49.     height, width = img.shape[0:2]
50.     gray = np.zeros((height, width), img.dtype)
51.     for i in range(height):
```

```
        for j in range(width):
            a = img[i, j, 2] ** 2.2 + 1.5 * img[i, j, 1] ** 2.2 + 0.6 * img[i, j, 0] ** 2.2  # 分子
            b = 1 + 1.5 ** 2.2 + 0.6 ** 2.2  # 分母
            gray[i, j] = pow(a / b, 1.0 / 2.2)  # 开 2.2 次方根，即 Gamma 矫正灰度处理
    gray = cv2.cvtColor(gray, cv2.COLOR_BGR2RGB)
    return gray


"""
二值化
"""


# 使用 opencv 算法求解全局阈值，作为参照
def threshold_Ref(img, threshold):
    img = cv2.imread(img)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    threshold, binary = cv2.threshold(gray, threshold, 255, cv2.THRESH_BINARY)
    binary = cv2.cvtColor(binary, cv2.COLOR_BGR2RGB)
    return binary


# 自定义二进制阈值，生成二值图像，大于阈值图像设置为 255
def threshold_self(img, threshold):  # threshold 为输入的阈值
    img = cv2.imread(img)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    height, width = gray.shape[0:2]
    binary = np.zeros((height, width), gray.dtype)  # 空白图像存放二值图像，避免对原图修改
    for x in range(height):
        for y in range(width):
            # 判断是否阈值，大于阈值设置为 255(全白)
            binary[x, y] = 255 if gray[x, y] >= threshold else 0
    binary = cv2.cvtColor(binary, cv2.COLOR_BGR2RGB)
    return binary


# 自定义反二进制阈值，生成二值图像，大于阈值图像设置为 0
def threshold_self_inv(img, threshold):
    img = cv2.imread(img)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
92.     height, width = gray.shape[0:2]
93.     binary = np.zeros((height, width), gray.dtype)
94.     for x in range(height):
95.         for y in range(width):
96.             if gray[x, y] >= threshold:
97.                 binary[x, y] = 0
98.             else:
99.                 binary[x, y] = 255
100.        binary = cv2.cvtColor(binary, cv2.COLOR_BGR2RGB)
101.        return binary
102.
103.
104.    """
105.    大津算法
106.    """
107.
108.
109.    # 大津二值化，返回二值化图像
110.    def otsu(image):
111.        image = cv2.imread(image)
112.        # 灰度化
113.        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
114.        height, width = gray.shape[0:2]
115.
116.        max_gray = 0  # 存储灰度类间方差
117.        suitable_th = 0  # 存储灰度类间方差对应的阈值
118.        ave, total = cv2.meanStdDev(gray)
119.
120.        # 遍历每一个灰度值
121.        for threshold in range(255):
122.            binary = gray > threshold  # 输出为 true/false
123.            binary_inv = gray <= threshold  # 输出为 true/false
124.            fore_pix = np.sum(binary)  # 前景像素总数(c1)
125.            back_pix = np.sum(binary_inv)  # 后景像素总数(c2)
126.            if fore_pix == 0:
127.                break
128.            if back_pix == 0:
129.                continue
130.
131.            w0 = float(fore_pix) / gray.size  # gray.size 获得灰度图像总长
    度，即像素个数
132.            u0 = float(np.sum(gray * binary)) / fore_pix  # c1 灰度值总和，
    u0 为 c1 平均灰度
133.            w1 = float(back_pix) / gray.size
```

```
134.             u1 = float(np.sum(gray * binary_inv)) / back_pix
135.
136.             class_Var = w0 * w1 * (u0 - u1) * (u0 - u1)  # class_Var 为类间
    方差
137.             # 取类间方差最大的对应阈值为最终阈值，并计算可分离测度
138.             if class_Var > max_gray:
139.                 max_gray = class_Var
140.                 suitable_th = threshold
141.             #     yita = float(class_Var / (total**2))
142.         # print("otsu 计算最适阈值为: %d, 可分离测度
    为: %.20f" % (suitable_th, yita))
143.
144.         # 生成 otsu 二值化图像
145.         binary_otsu = np.zeros((height, width), gray.dtype)
146.         for i in range(height):
147.             for j in range(width):
148.                 if gray[i, j] >= suitable_th:
149.                     binary_otsu[i, j] = 255
150.                 else:
151.                     binary_otsu[i, j] = 0
152.         binary_otsu = cv2.cvtColor(binary_otsu, cv2.COLOR_BGR2RGB)
153.         return binary_otsu
154.
155.
156.     """
157.     Canny 算法进行边界提取
158.     """
159.
160.
161.     # Canny 算法边缘提取
162.     def Canny(img, minT, maxT):
163.         """
164.         Canny 算法边缘提取
165.         :param img: 原图像，不为灰度
166.         :param minT: 低阈值
167.         :param maxT: 高阈值
168.         :return: Canny 算法结果图像
169.         """
170.         img = cv2.imread(img)
171.         # 获取灰度图像
172.         gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
173.         # 定义空白图像，用于存放 LOG 算法提取出的轮廓图
174.         # canny_img = np.zeros((height, width), dtype=gray.dtype)
175.         # 定义空白图像，用于存放高斯滤波处理后的图像
```

```
        height, width = gray.shape[0:2]
        gauss_img = np.zeros((height, width), dtype=gray.dtype)
        # 定义高斯核卷积模板
        a = np.array([[1, 4, 7, 4, 1],
                      [4, 16, 26, 16, 4],
                      [7, 26, 41, 26, 7],
                      [4, 16, 26, 16, 4],
                      [1, 4, 7, 4, 1]])
        kernel = a * (1 / 273)

        '''''
        使用高斯平滑去除噪声
        '''
        # 对图像进行高斯滤波处理
        for i in range(2, height - 2):
            for j in range(2, width - 2):
                sum_num = np.sum(gray[i - 2:i + 2 + 1, j - 2:j + 2 + 1] * kernel)
                gauss_img[i, j] = sum_num

        '''''
        按照 Sobel 滤波器步骤计算梯度幅值与方向，寻找图像强度梯度
        '''
        sobel_img = np.zeros((height, width), dtype=gray.dtype)
        # 对阈值化图像遍历，进行 Sobel 求解梯度值
        for i in range(height - 1):
            for j in range(width - 1):
                dx = (int(gauss_img[i - 1, j - 1]) + 2 * int(gauss_img[i - 1, j]) + int(gauss_img[i - 1, j + 1])) - (
                        int(gauss_img[i + 1, j - 1]) + 2 * int(gauss_img[i + 1, j]) + int(gauss_img[i + 1, j + 1]))
                dy = (int(gauss_img[i - 1, j + 1]) + 2 * int(gauss_img[i, j + 1]) + int(gauss_img[i + 1, j + 1])) - (
                        int(gauss_img[i - 1, j - 1]) + 2 * int(gauss_img[i, j - 1]) + int(gauss_img[i + 1, j - 1]))
                sobel_img[i, j] = np.sqrt(dx ** 2 + dy ** 2)

        '''''
        通过 Non-maximum Suppression 过滤非边缘元素
        '''
        suppression_img = np.zeros((height, width), dtype=gray.dtype)
        # 对阈值化图像遍历，进行 non-maximum suppression
        for i in range(height - 1):
            for j in range(width - 1):
```

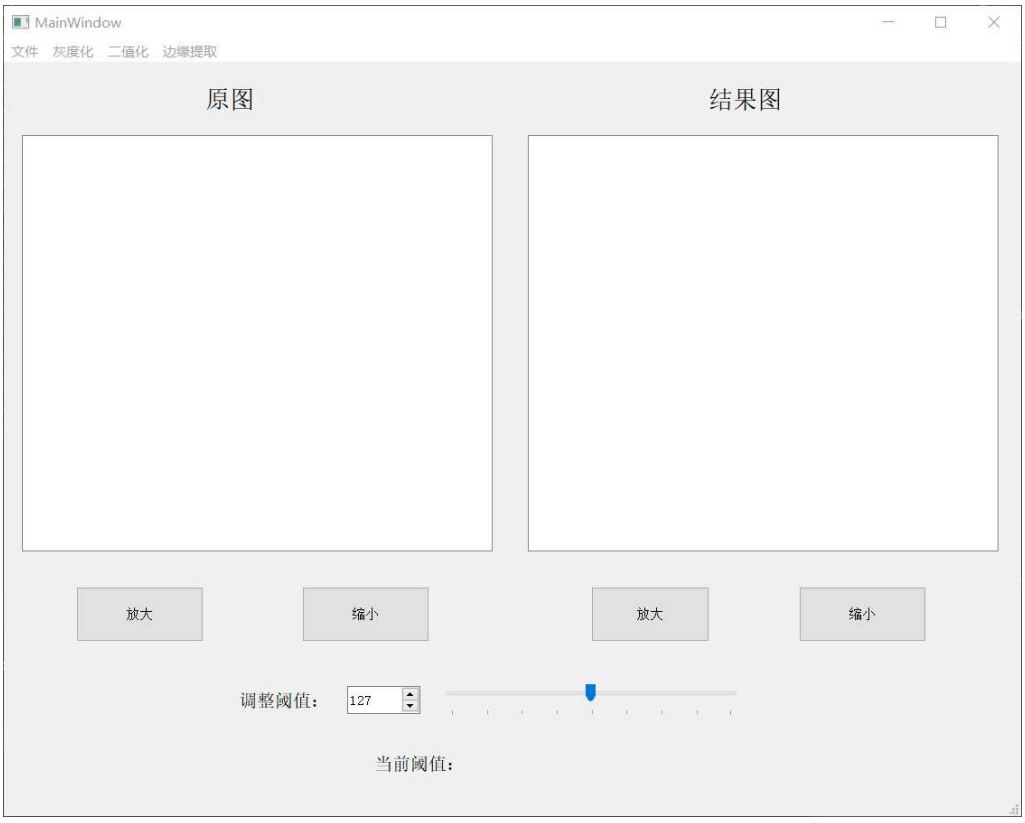
```
                dx = (int(gauss_img[i - 1, j - 1]) + 2 * int(gauss_img[i - 1, j]) + int(gauss_img[i - 1, j + 1])) - (
                        int(gauss_img[i + 1, j - 1]) + 2 * int(gauss_img[i + 1, j]) + int(gauss_img[i + 1, j + 1]))
                dy = (int(gauss_img[i - 1, j + 1]) + 2 * int(gauss_img[i, j + 1]) + int(gauss_img[i + 1, j + 1])) - (
                        int(gauss_img[i - 1, j - 1]) + 2 * int(gauss_img[i, j - 1]) + int(gauss_img[i + 1, j - 1]))

                # 确保分母不为0
                dx = np.maximum(dx, 1e-10)
                theta = np.arctan(dy / dx)

                '''''
                确定梯度角度
                '''
                if -0.4142 < theta < 0.4142:
                    angle = 0
                elif 0.4142 < theta < 2.4142:
                    angle = 45
                elif abs(theta) > 2.4142:
                    angle = 90
                elif -2.4142 < theta < -0.4142:
                    angle = 135

                '''''
                根据梯度角度方向，求对应的非极大值抑制
                '''
                if angle == 0:
                    if max(sobel_img[i, j], sobel_img[i, j - 1], sobel_img[i, j + 1]) == sobel_img[i, j]:
                        # 比较x方向梯度三个值中的最大值，如果Sobel_img[i,j]最大则保留，否则设置为0
                        suppression_img[i, j] = sobel_img[i, j]
                    else:
                        suppression_img[i, j] = 0
                elif angle == 45:
                    if max(sobel_img[i, j], sobel_img[i - 1, j + 1], sobel_img[i + 1, j - 1]) == sobel_img[i, j]:
                        # 比较正对角线方向梯度三个值中的最大值，如果sobel_img[i,j]最大则保留，否则设置为0
                        suppression_img[i, j] = sobel_img[i, j]
                    else:
                        suppression_img[i, j] = 0
```

```
251.                 elif angle == 90:
252.                     if max(sobel_img[i, j], sobel_img[i - 1, j], sobel_img
    [i + 1.j]) == sobel_img[i, j]:
253.                         # 比较 y 方向梯度三个值中的最大值，如果 sobel_img[i,j]
    最大则保留，否则设置为 0
254.                         suppression_img[i, j] = sobel_img[i, j]
255.                     else:
256.                         suppression_img[i, j] = 0
257.                 elif angle == 135:
258.                     if max(sobel_img[i, j], sobel_img[i - 1, j - 1], sobel
    _img[i + 1, j - 1]) == sobel_img[i, j]:
259.                         # 比较反对角线方向梯度三个值中的最大值，如果
    sobel_img[i,j]最大则保留，否则设置为 0
260.                         suppression_img[i, j] = sobel_img[i, j]
261.                     else:
262.                         suppression_img[i, j] = 0
263.
264.         '''''
265.         利用双阈值方法确定潜在的边界
266.         '''
267.         canny_img = np.zeros((height, width), dtype=gray.dtype)
268.         # 对阈值化图像遍历，进行双阈值处理
269.         for i in range(height):
270.             for j in range(width):
271.                 if suppression_img[i, j] >= maxT:  # 大于高阈值，设置为
    255
272.                     canny_img[i, j] = 255
273.                 elif suppression_img[i, j] <= minT:  # 小于低阈值，设置为
    0
274.                     canny_img[i, j] = 0
275.                 else:
276.                     '''''
277.                     利用滞后技术跟踪边界，若某一像素位置和 strong edge 相连的
    weak edge 认定是边界，其余的若边界删除(设为 0)
278.                     '''
279.                     # 周围 8 邻域内有比该像素值更大的像素，则设置为 255，否则设
    置为 0
280.                     if max(suppression_img[i - 1, j - 1], suppression_img[
    i - 1, j], suppression_img[i - 1, j + 1],
281.                            suppression_img[i, j - 1], suppression_img[i, j
     + 1], suppression_img[i + 1, j - 1],
282.                            suppression_img[i + 1, j], suppression_img[i +
    1, j + 1]) >= suppression_img[i, j]:
283.                         canny_img[i, j] = 255
```

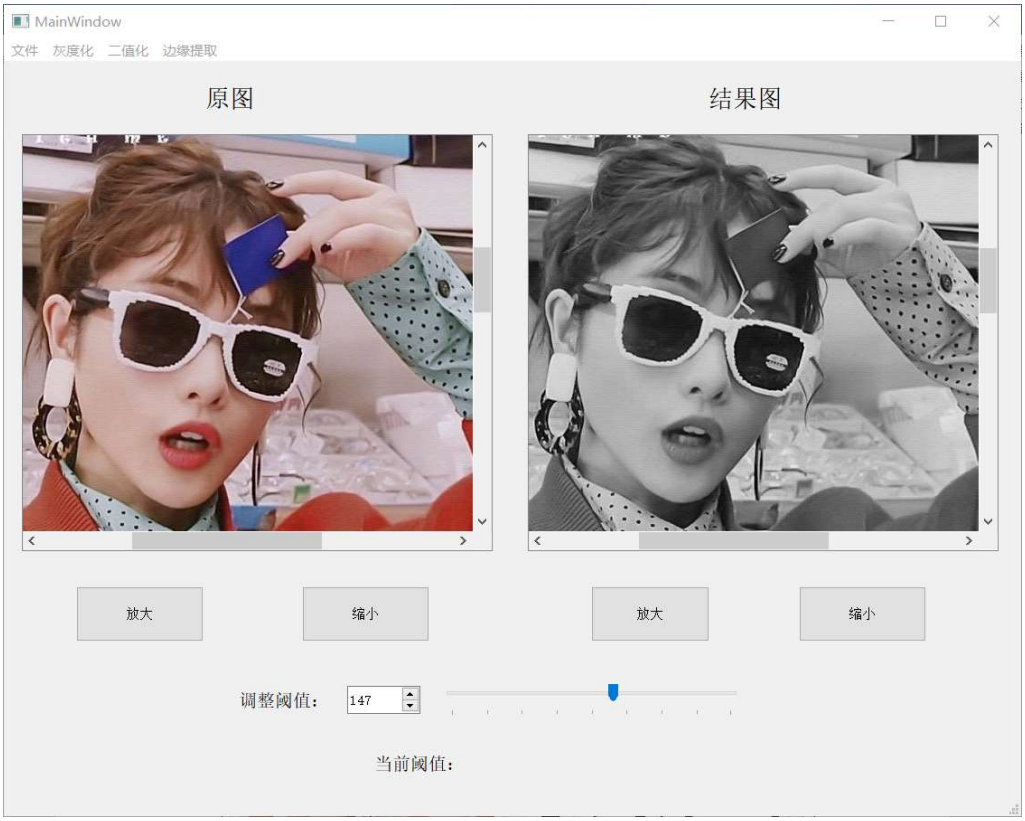
```
                    else:
                        canny_img[i, j] = 0

        # 转为可输出格式
        canny_img = cv2.cvtColor(canny_img, cv2.COLOR_BGR2RGB)
        return canny_img


    # opencv 进行 Canny 算法
    def Canny_opencv(img, minT, maxT):
        img = cv2.imread(img)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        blur = cv2.GaussianBlur(gray, (5, 5), 0)  # 高斯滤波
        Canny_op = cv2.Canny(blur, minT, maxT)  # 80 为低阈值，255 为高阈值
        return Canny_op
```

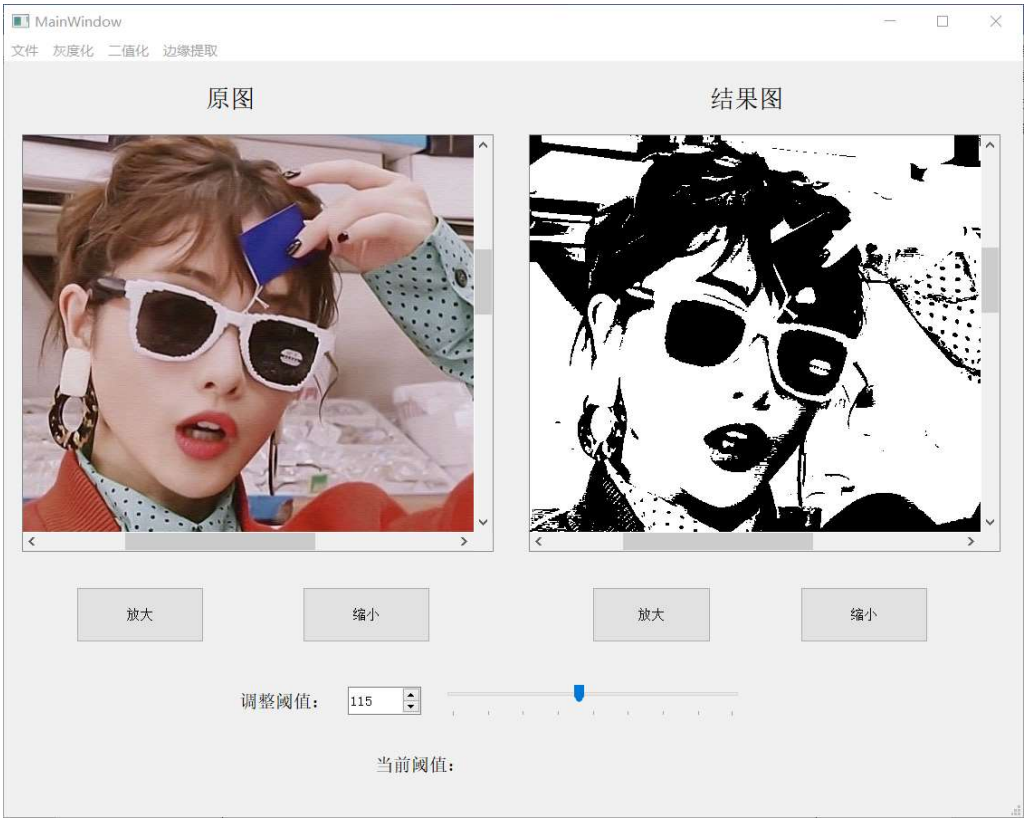
五、 实验原始数据与实验结果：

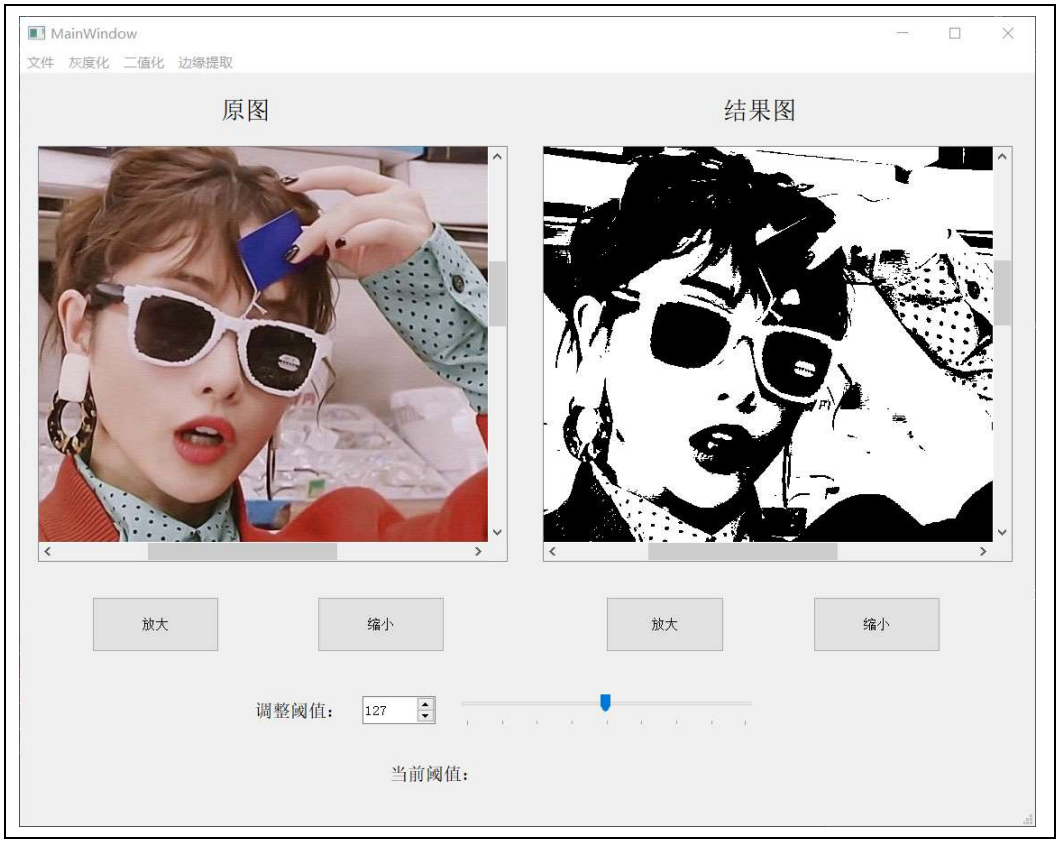
源图像：



结果图像：（分别展示灰度化、二值化、大津算法）







六、 问题分析与心得体会：

在实验过程中，主要遇到了以下问题并做出总结：

1. 界面拟采用 PyQt 设计并实现，由于之前没有做界面设计的经验，因此此次尽最大努力，在满足界面满足基本功能的基础上，再考虑界面的美观优化。
2. 界面的调动和重新设计非常麻烦。因此将主程序（main.py）和界面设计（interface.py）的代码相分离，main.py 中为界面控件之间的逻辑关系和功能实现。写在两个不同的文件中，并通过 class 连接。这样做成功将界面设计和界面的逻辑关系功能实现代码相分离，实现了即使界面的控件、菜单等随意调动，甚至添加或删除控件都不再会影响其背后的逻辑关系。

3. 直方图统计的功能还未添加至界面代码中。
4. 界面设计显示与保存图像无法进行很好的取舍，仍留有较大的遗憾。这主要表现在两个方面，一方面，在使用 Label 显示图像时，图像的显示是压缩后的固定大小，使得图像大小改变，存在一定程度的失真；另一方面，在使用 GraphicalView 显示图像时，图像的显示和原图相同（通过显示框旁的滑动条滑动查看图像的不同区域），但是只能保存显示框及其以内的图像大小。二者的保存图像效果如下：

Label 保存

GraphicalView 保存

遗憾的是，由于时间原因，对于此点还未有相应的处理方法，因此预想是添加一个方法或控件，能够全屏显示图像。

5. 由于时间紧迫的原因，在界面设计已临近考试周，学习任务急剧增长，因此该界面还有许多功能未能完成、背后的逻辑关系仍需要重新设计。同时，界面的要求和实现目标需要随需求改变，需要有更加优质的想法衬托。本实验所有代码已上传 GitHub(https://github.com/Chloewz/DigitalImageProcessing)