Національний авіаційний університет

Факультет кібербезпеки, комп’ютерної та програмної інженерії

**Лабораторна робота №2**

Дисципліна: «Високорівневі мови програмування»

Кафедра: прикладної математики

ОС: бакалавр

Спеціальність: 113 «Прикладна математика»

ОПП: «Прикладне програмне забезпечення»

Виконав: здобувач вищої освіти 3 курсу. 351 групи

Архіпов О.Т.

Перевірив: Сорокопуд В.І.

Київ 2023

**Зміст**

1. Завдання
2. Теоретична частина
3. Результат роботи програми
4. Висновок
5. Додаток

**Завдання**

1. Створити графічний застосунок для роботи з даними з використанням технологій PyQt5, та OpenCV.
2. Розробити макет графічного застосунку в редакторі та зберегти його в ui файл. Згенерувати py файл з ui файлу.
3. Застосунок повинен мати функціонал для завантаження зображення та його відображення.
4. Застосунок повинен демонструвати графіки розподілу кольорів та базові показники.
5. Застосунок повен мати функції зміни розміру зображення.
6. Застосунок повинен мати функції перетворення зображення в gray scale.
7. Застосунок повинен мати функцію накладання будь-якого фільтру розмірності 3 на 3.
8. Застосунок повинен мати функцію для збереження отриманого зображення.

**Теоретична частина**

OpenCV - це open source бібліотека комп'ютерного зору, яка призначена для аналізу, класифікації та обробки зображень. Широко використовується у таких мовах як C, C++, Python та Java.

Кожне зображення складається із набору пікселів. Піксель – це будівельний блок зображення. Якщо уявити зображення як сітки, кожен квадрат у сітці містить один піксель, де точці з координатою ( 0, 0 ) відповідає верхній лівий кут зображення. Наприклад, уявимо, що у нас є зображення з роздільною здатністю 400x300 пікселів. Це означає, що наша сітка складається з 400 рядків та 300 стовпців. У сукупності у зображенні є 400\*300 = 120000 пікселів.

У більшості зображень пікселі представлені двома способами: у відтінках сірого та у колірному просторі RGB. У зображеннях у відтінках сірого кожен піксель має значення між 0 і 255 де 0 відповідає чорному, а 255 відповідає білому. А значення між 0 і 255 приймають різні відтінки сірого, де значення ближче до 0 темніші, а значення ближче до 255 світліші:

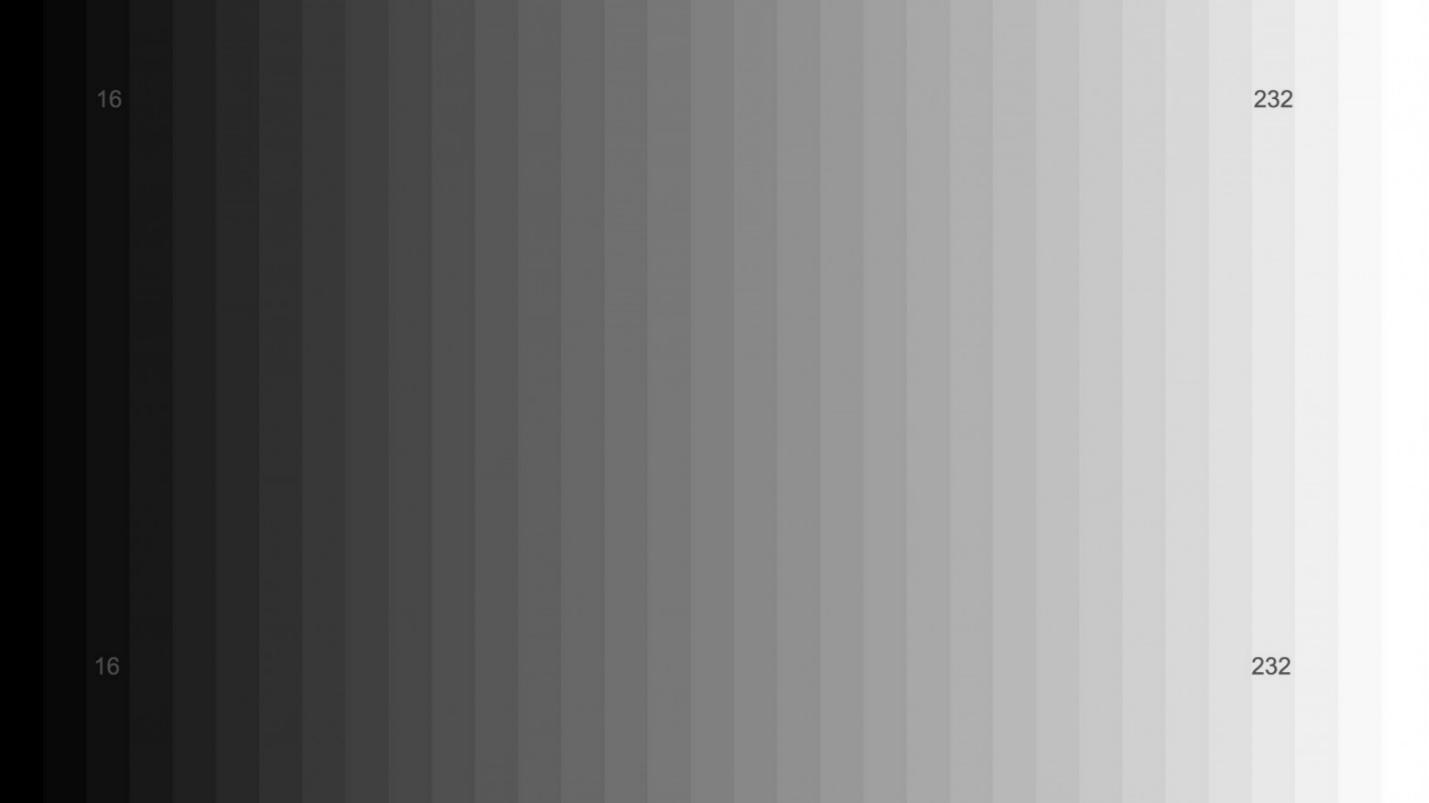


Рисунок 1. Відтінки сірого

Кольорові пікселі зазвичай представлені в колірному просторі RGB(red, green, blue — червоний, зелений, синій), де одне значення для червоної компоненти, одне для зеленої та синій. Кожна з трьох компонентів представлена ​​цілим числом в діапазоні від 0 до 255 включно, яке вказує як «багато» кольору міститься. Виходячи з того, що кожна компонента представлена ​​в діапазоні [0,255], для того, щоб уявити насиченість кожного кольору, нам буде достатньо 8-бітного цілого беззнакового числа. Потім ми об'єднуємо значення всіх трьох компонентів у кортеж виду (червоний, зелений, синій). Наприклад, щоб отримати білий колір, кожна з компонентів повинна дорівнювати 255: (255, 255, 255). Тоді, щоб отримати чорний колір, кожна з компонентів повинна бути рівною 0: (0, 0, 0). Нижче наведені поширені кольори,

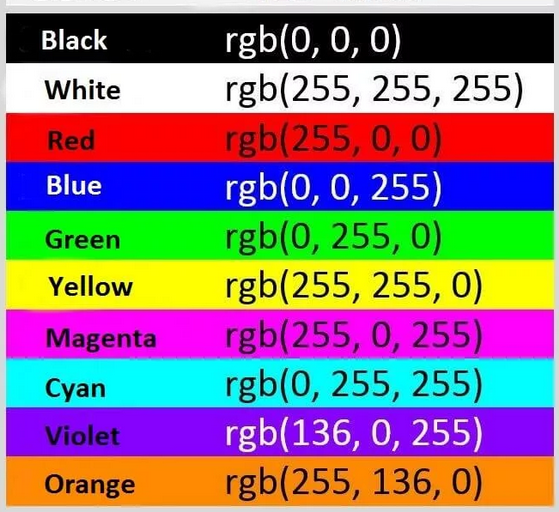


Рисунок 2. Поширені кольори

**Результат роботи програми**

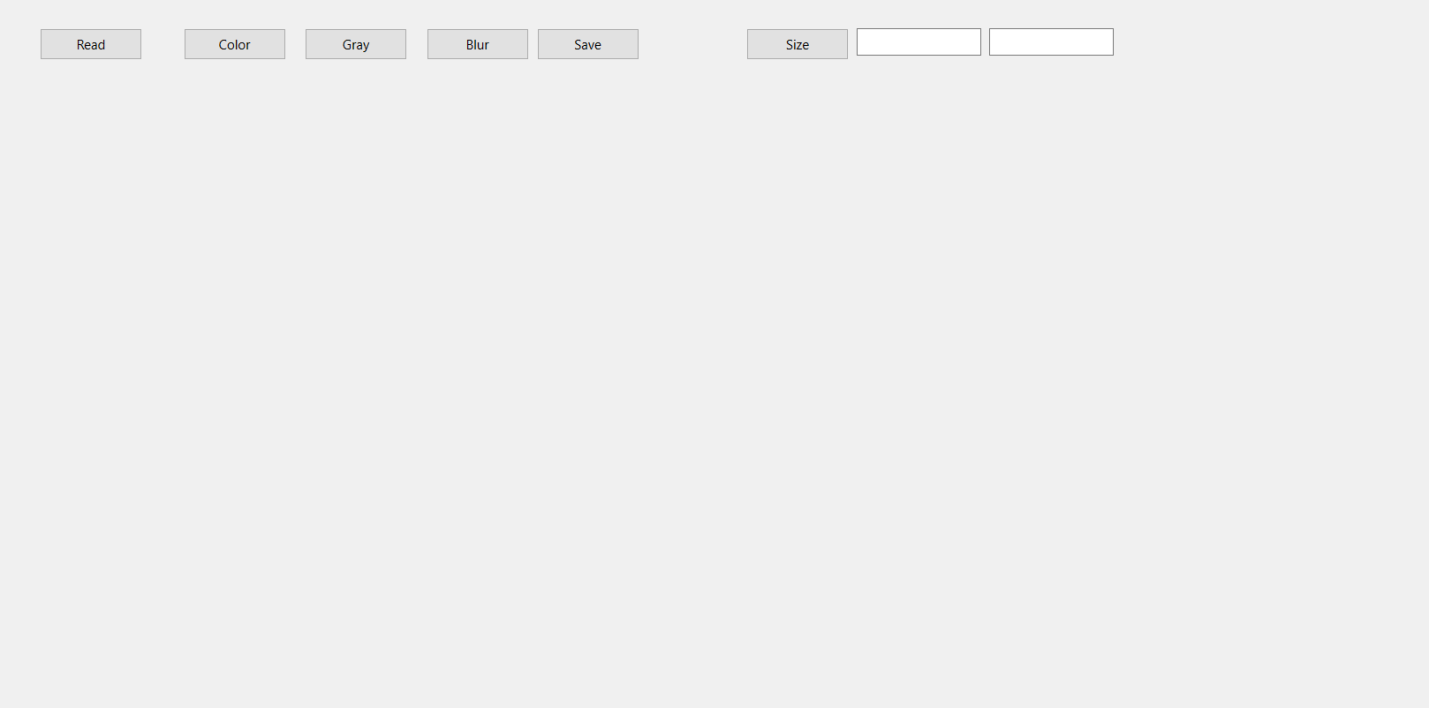


Рисунок 3. Вікно програми

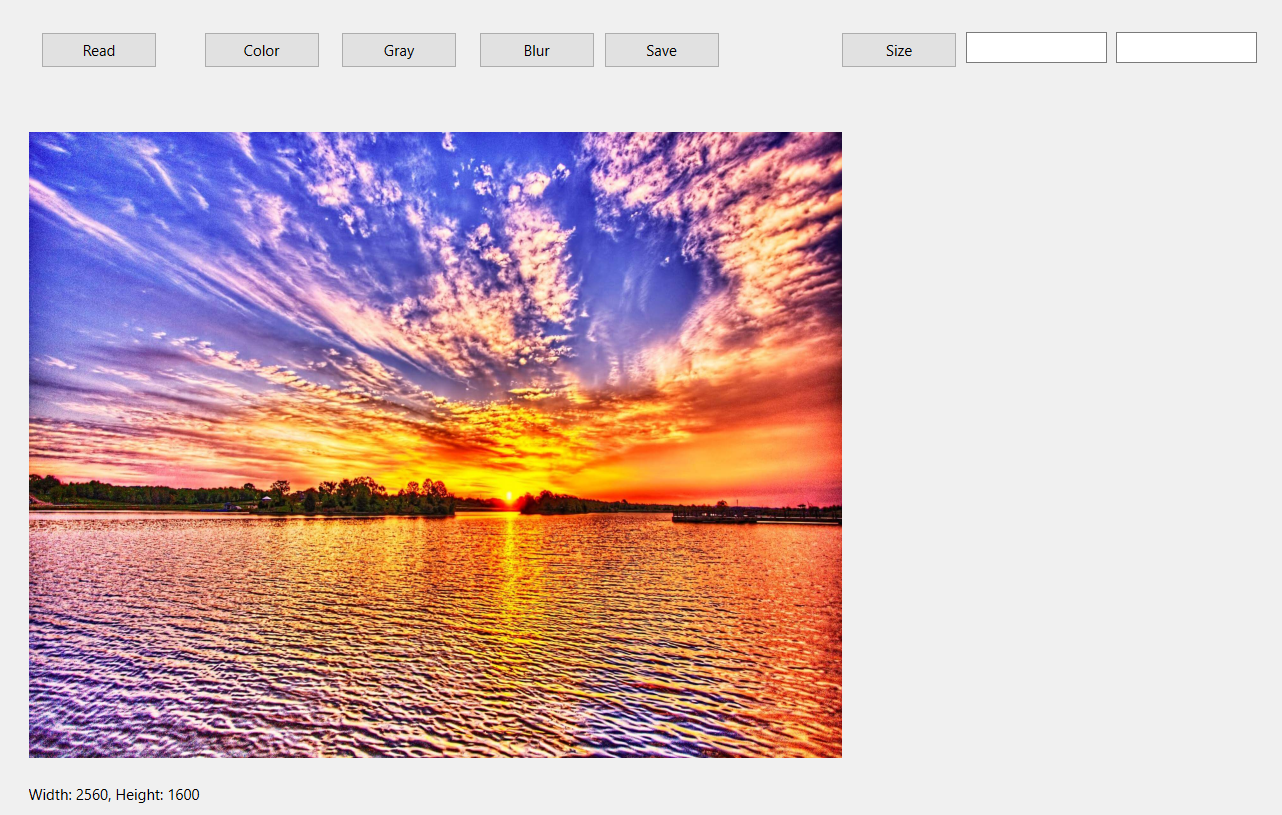


Рисунок 4. Зчитане зображення

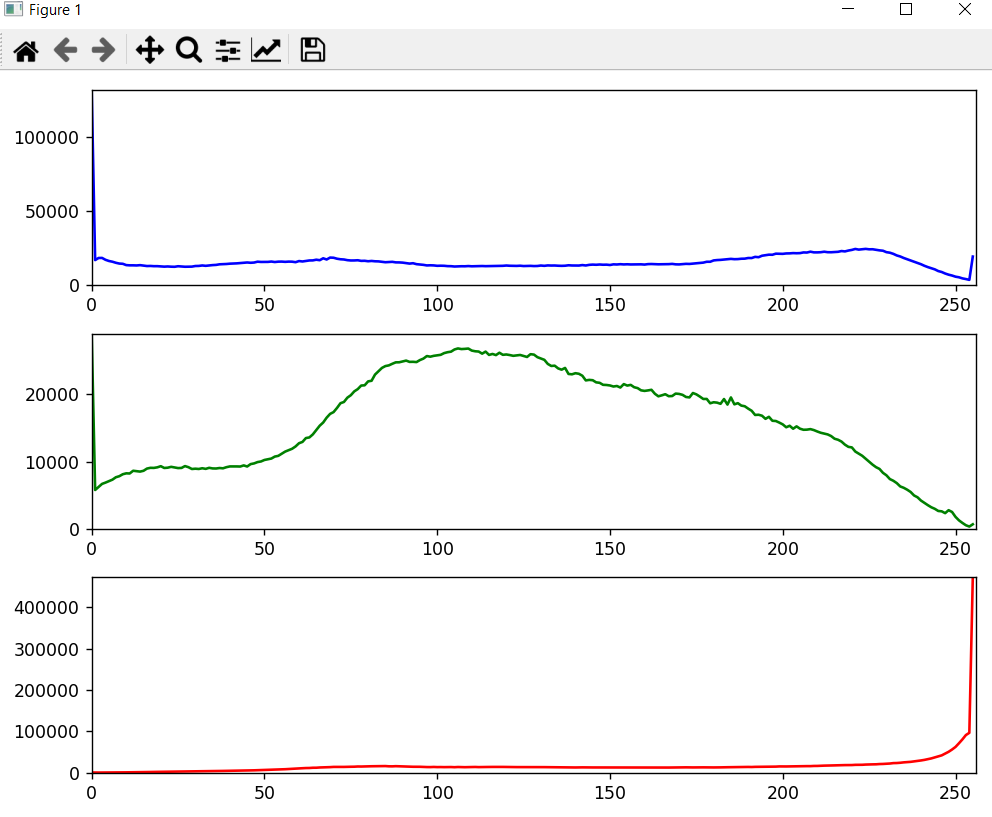


Рисунок 5. Графіки розподілів кожного з кольорів

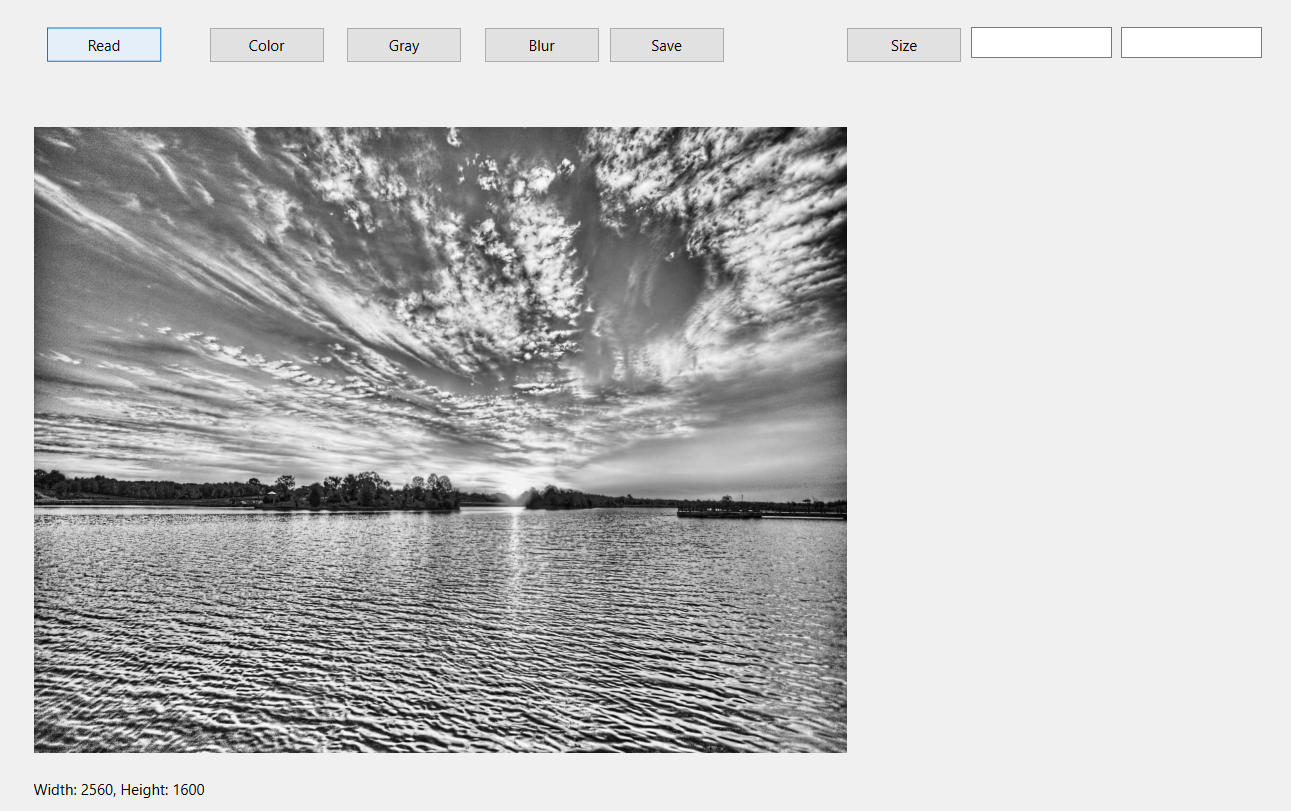


Рисунок 6. Переведення у gray scale

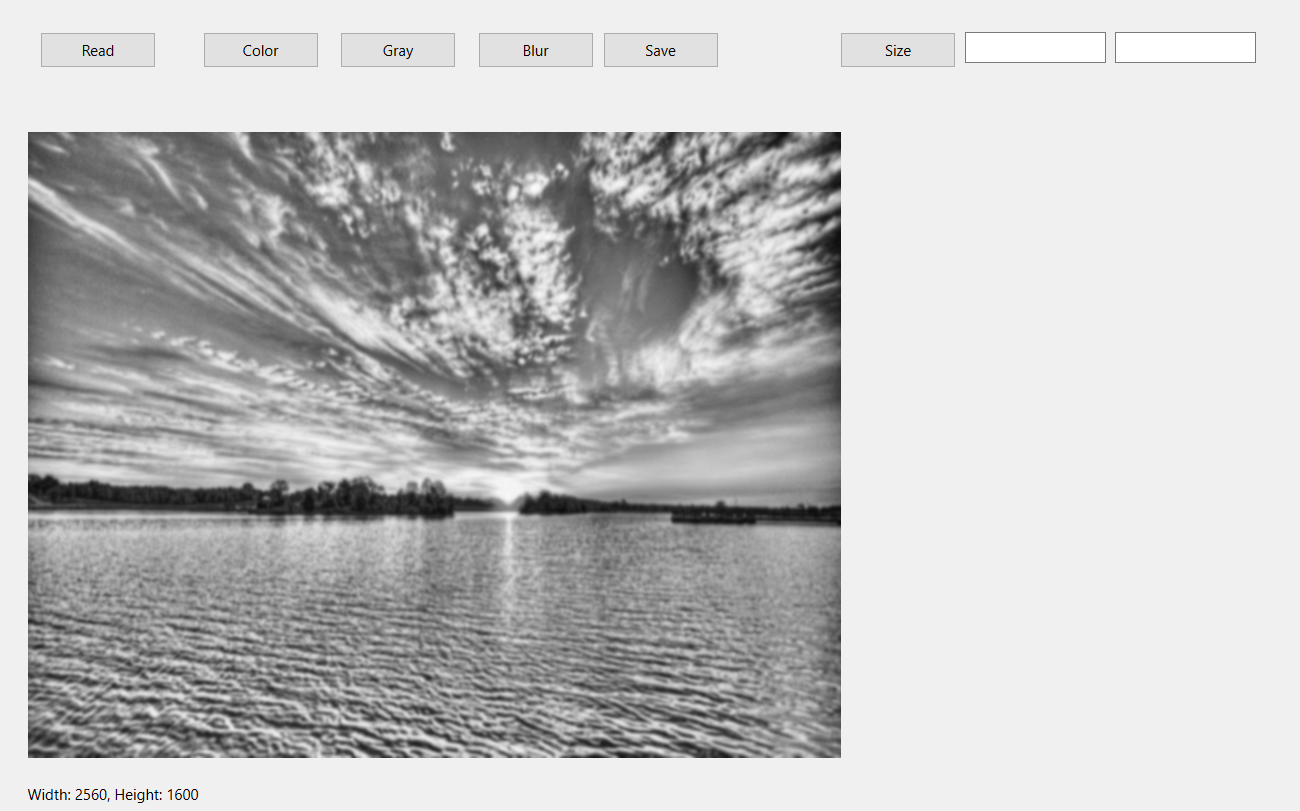


Рисунок 7. Заблюрення

**Висновок**

1. Створив графічний застосунок для роботи з даними з використанням технологій PyQt5, та OpenCV.
2. Розробив макет графічного застосунку в редакторі та зберегти його в ui файл. Згенерував py файл з ui файлу.
3. Застосунок має функціонал для завантаження зображення та його відображення.
4. Застосунок демонструє графіки розподілу кольорів та базові показники.
5. Застосунок має функції зміни розміру зображення.
6. Застосунок має функції перетворення зображення в gray scale.
7. Застосунок має функцію замилення.
8. Застосунок має функцію для збереження отриманого зображення.

**Додаток**

Widget.py

import sys

import cv2

from PySide6.QtWidgets import QApplication, QWidget, QFileDialog, QLabel

from PySide6.QtGui import QPixmap, Qt, QImage, QColor, qRgb

from matplotlib import pyplot

import numpy as np

from ui\_form import Ui\_Widget

class Widget(QWidget):

    def \_\_init\_\_(self, parent=None):

        super().\_\_init\_\_(parent)

        self.ui = Ui\_Widget()

        self.ui.setupUi(self)

        self.ui.pushButton.clicked.connect(self.Read)

        self.ui.btn\_color.clicked.connect(self.Color\_RGB)

        self.ui.btn\_gray.clicked.connect(self.Gray)

        self.ui.btn\_blur.clicked.connect(self.Blur)

        self.ui.btn\_size.clicked.connect(self.Size)

        self.ui.btn\_save.clicked.connect(self.Save)

        self.picture = QPixmap

        self.file = QFileDialog

    def Read(self):

         file, \_ = QFileDialog.getOpenFileName(None, "Select image file", "", "Image Files (\*.jpg);;Image Files (\*.png);;All Files (\*)")

         if file:

             pixmap = QPixmap(file)

             self.picture = pixmap

             self.file = file

             width = pixmap.width()

             height = pixmap.height()

             self.ui.label\_2.setText(f'Width: {width}, Height: {height}')

             self.ui.label.setPixmap(pixmap)

             self.ui.label.setScaledContents(True)

             self.ui.label.setAspectRatioMode(Qt.IgnoreAspectRatio)

    def Color\_RGB(self):

        img = cv2.imread(self.file)

        if img is not None:

            channels = cv2.split(img)

            colors = ('b', 'g', 'r')

            fig, axes = pyplot.subplots(3, 1, figsize=(8, 6))

            for i, channel in enumerate(channels):

                hist = cv2.calcHist([channel], [0], None, [256], [0, 256])

                axes[i].plot(hist, color=colors[i])

                axes[i].set\_xlim([0, 256])

                axes[i].set\_ylim([0, hist.max()])

            pyplot.tight\_layout()

            pyplot.show()

    def Gray(self):

        image = self.picture.toImage()

        gray\_image = image.convertToFormat(QImage.Format\_Grayscale8)

        grayscale\_pixmap = QPixmap.fromImage(gray\_image)

        self.ui.label.setPixmap(grayscale\_pixmap)

        self.picture = grayscale\_pixmap

    def Blur(self):

        image = self.picture.toImage()

        width = image.width()

        height = image.height()

        buffer = image.bits().tobytes()

        image\_np = np.frombuffer(buffer, dtype=np.uint8).reshape((height, width, 4))

        image\_bgr = cv2.cvtColor(image\_np, cv2.COLOR\_RGBA2RGB)

        blue\_channel, green\_channel, red\_channel = cv2.split(image\_bgr)

        filter\_matrix = np.array([[1, 0, 1], [0, 1, 0], [1, 0, 1]], dtype=np.float32) / 5

        filtered\_blue\_channel = cv2.filter2D(blue\_channel, -1, kernel=filter\_matrix)

        filtered\_green\_channel = cv2.filter2D(green\_channel, -1, kernel=filter\_matrix)

        filtered\_red\_channel = cv2.filter2D(red\_channel, -1, kernel=filter\_matrix)

        filtered\_image\_bgr = cv2.merge([filtered\_blue\_channel, filtered\_green\_channel, filtered\_red\_channel])

        filtered\_image = QImage(filtered\_image\_bgr.data, filtered\_image\_bgr.shape[1], filtered\_image\_bgr.shape[0], QImage.Format\_BGR888)

        filtered\_pixmap = QPixmap.fromImage(filtered\_image)

        self.ui.label.setPixmap(filtered\_pixmap)

        self.picture = filtered\_pixmap

    def Size(self):

        user\_width = int(self.ui.lineEdit.text())

        user\_height = int(self.ui.lineEdit\_2.text())

        resized\_image = self.picture.scaled(user\_width, user\_height)

        width = resized\_image.width()

        height = resized\_image.height()

        self.ui.label\_2.setText(f'Width: {width}, Height: {height}')

        self.ui.label.setPixmap(resized\_image)

        self.picture = resized\_image

    def Save(self):

        file\_path = "E:\p.jpg"

        self.picture.save(file\_path, "JPG")

if \_\_name\_\_ == "\_\_main\_\_":

    app = QApplication(sys.argv)

    widget = Widget()

    widget.show()

    sys.exit(app.exec())

ui\_form.py

from PySide6.QtCore import (QCoreApplication, QDate, QDateTime, QLocale,

    QMetaObject, QObject, QPoint, QRect,

    QSize, QTime, QUrl, Qt)

from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,

    QFont, QFontDatabase, QGradient, QIcon,

    QImage, QKeySequence, QLinearGradient, QPainter,

    QPalette, QPixmap, QRadialGradient, QTransform)

from PySide6.QtWidgets import (QApplication, QLabel, QLineEdit, QPushButton,

    QSizePolicy, QWidget)

class Ui\_Widget(object):

    def setupUi(self, Widget):

        if not Widget.objectName():

            Widget.setObjectName(u"Widget")

        Widget.resize(1310, 658)

        self.pushButton = QPushButton(Widget)

        self.pushButton.setObjectName(u"pushButton")

        self.pushButton.setGeometry(QRect(40, 30, 93, 29))

        self.label = QLabel(Widget)

        self.label.setObjectName(u"label")

        self.label.setGeometry(QRect(30, 110, 651, 501))

        self.label\_2 = QLabel(Widget)

        self.label\_2.setObjectName(u"label\_2")

        self.label\_2.setGeometry(QRect(30, 630, 691, 20))

        self.btn\_color = QPushButton(Widget)

        self.btn\_color.setObjectName(u"btn\_color")

        self.btn\_color.setGeometry(QRect(170, 30, 93, 29))

        self.btn\_gray = QPushButton(Widget)

        self.btn\_gray.setObjectName(u"btn\_gray")

        self.btn\_gray.setGeometry(QRect(280, 30, 93, 29))

        self.btn\_blur = QPushButton(Widget)

        self.btn\_blur.setObjectName(u"btn\_blur")

        self.btn\_blur.setGeometry(QRect(390, 30, 93, 29))

        self.lineEdit = QLineEdit(Widget)

        self.lineEdit.setObjectName(u"lineEdit")

        self.lineEdit.setGeometry(QRect(780, 30, 113, 25))

        self.lineEdit\_2 = QLineEdit(Widget)

        self.lineEdit\_2.setObjectName(u"lineEdit\_2")

        self.lineEdit\_2.setGeometry(QRect(900, 30, 113, 25))

        self.btn\_size = QPushButton(Widget)

        self.btn\_size.setObjectName(u"btn\_size")

        self.btn\_size.setGeometry(QRect(680, 30, 93, 29))

        self.btn\_save = QPushButton(Widget)

        self.btn\_save.setObjectName(u"btn\_save")

        self.btn\_save.setGeometry(QRect(490, 30, 93, 29))

        self.retranslateUi(Widget)

        QMetaObject.connectSlotsByName(Widget)

    # setupUi

    def retranslateUi(self, Widget):

        Widget.setWindowTitle(QCoreApplication.translate("Widget", u"Widget", None))

        self.pushButton.setText(QCoreApplication.translate("Widget", u"Read", None))

        self.label.setText("")

        self.label\_2.setText("")

        self.btn\_color.setText(QCoreApplication.translate("Widget", u"Color", None))

        self.btn\_gray.setText(QCoreApplication.translate("Widget", u"Gray", None))

        self.btn\_blur.setText(QCoreApplication.translate("Widget", u"Blur", None))

        self.btn\_size.setText(QCoreApplication.translate("Widget", u"Size", None))

        self.btn\_save.setText(QCoreApplication.translate("Widget", u"Save", None))