```
# Hollow 图片（视频）信息隐藏加解密软件 V1.0
# 编程语言：Python 3
from PIL import Image
import sys
import os
import time
import numpy as np
import math
import random
import threading
import cv2
from PyQt5.QtWidgets import *
from PyQt5.QtGui import QIcon
from PyQt5 import QtCore, QtGui, QtWidgets
import res_rc # 为 res.qrc 编译生成的软件图标二进制数据

# 对图像的红绿蓝三通道合并，处理变换中的异常像素值，并且输出文件
def output_file(picname, red, green, blue, width, height):
    for i in range(width):
        for j in range(height):
            if red[i, j] > 255:
                red[i, j] = 255
            elif red[i, j] < 0:
                red[i, j] *= -1
    for i in range(width):
        for j in range(height):
            if green[i, j] > 255:
                green[i, j] = 255
            elif green[i, j] < 0:
                green[i, j] *= -1
    for i in range(width):
        for j in range(height):
            if blue[i, j] > 255:
                blue[i, j] = 255
            elif blue[i, j] < 0:
                blue[i, j] *= -1
    red = red.T
    green = green.T
    blue = blue.T
    source = Image.new("RGB", (width, height))
    img = np.array(source)
    img[:, :, 0] = red[:, :]
    img[:, :, 1] = green[:, :]
    img[:, :, 2] = blue[:, :]
    fo = Image.fromarray(img, 'RGB')
    fo.save(picname)

# 进行离散小波变换处理
def dwt(colour, width, height):
    TEMP1 = np.zeros((width, height), int)
```

```
51      TEMP2 = np.zeros((width, height), int)
52      for i in range(width):
53          for j in range(height//2):
54              tmp = (colour[i, j*2] + colour[i, j * 2 + 1])
55              if((tmp < 0 or (tmp == (-1) or tmp // 2 == (-
    1))) and ((colour[i, j * 2] + colour[i, j * 2+1]) % 2) != 0):
56                  TEMP1[i, j] = tmp // 2 - 1
57              else:
58                  TEMP1[i, j] = tmp // 2
59          ct = height // 2
60          for j in range(height//2):
61              TEMP1[i, ct] = (colour[i, j * 2] - colour[i, j * 2 + 1])
62              ct += 1
63      for j in range(height):
64          for i in range(width//2):
65              tmp = (TEMP1[i * 2, j] + TEMP1[i * 2 + 1, j])
66              if ((tmp < 0 or tmp == (-1) or tmp // 2 == (-
    1)) and ((TEMP1[i * 2, j] + TEMP1[i * 2 + 1, j]) % 2) != 0):
67                  TEMP2[i, j] = tmp // 2 - 1
68              else:
69                  TEMP2[i, j] = tmp // 2
70          ct = width // 2
71          for i in range(width//2):
72              TEMP2[ct, j] = (TEMP1[i * 2, j] - TEMP1[i * 2 + 1, j])
73              ct += 1
74      return TEMP2
75
76  # 对图片的隐藏信息进行提取
77  def extract_watermark(TEMP2, width, height, mwidth, mheight, comple
    x, passwd):
78      outputwm = np.zeros((mwidth*mheight), int)
79      order = np.zeros((mwidth*mheight), int)
80      tmp = np.zeros((mwidth*mheight), int)
81      k = 5
82      position_pixel = 0
83      for i in range(width // 2, width):
84          for j in range(height // 2, height):
85              p = math.fabs(TEMP2[i, j])
86              k = p % 10
87              outputwm[position_pixel] = p % 10
88              p = int(p//10)
89              if(p % 2 == 0):
90                  outputwm[position_pixel] = 9+outputwm[position_pixel]
91              outputwm[position_pixel] = math.pow(
92                  outputwm[position_pixel]+complex[position_pixel], 2)
93              if(outputwm[position_pixel] > 255 and(k >= 7 and k < 10)):
94                  outputwm[position_pixel] = math.pow(
95                      k+complex[position_pixel], 2)
96              else:
97                  outputwm[position_pixel] = outputwm[position_pixel]
```

```
98              position_pixel += 1
99              if position_pixel == mwidth*mheight:
100                 break
101         if position_pixel == mwidth*mheight:
102             break
103     random.seed(passwd)
104     for i in range(position_pixel):
105         order[i] = i
106     random.shuffle(order)
107     for i in range(position_pixel):
108         tmp[order[i]] = outputwm[i]
109     for i in range(position_pixel):
110         outputwm[i] = tmp[i]
111     return outputwm
112
113 # 向图片写入隐藏信息
114 def hiding_data(TEMP2, filename, flag, writedata, passwd):
115     width = TEMP2.shape[0]
116     height = TEMP2.shape[1]
117     watermark = Image.open(filename)
118     if len(watermark.split()) < 3:
119         watermark = watermark.convert("RGB")
120     mk = watermark.load()
121     wm_width, wm_height = watermark.size
122     outputwm = np.zeros((wm_width*wm_height), int)
123     wm_pixel = np.zeros((wm_width*wm_height), int)
124     complex = np.zeros((wm_width*wm_height), float)
125     k = 5
126     position_pixel = 0
127     for i in range(wm_width):
128         for j in range(wm_height):
129             wm_pixel[position_pixel] = int(mk[i, j][flag])
130             position_pixel += 1
131     random.seed(passwd)
132     random.shuffle(wm_pixel)
133     position_pixel = 0
134     for i in range(width // 2, width):
135         for j in range(height // 2, height):
136             a = math.fabs(TEMP2[i, j])
137             flag = 1
138             if(TEMP2[i, j] < 0):
139                 flag = -1
140             diwm = round(math.sqrt(wm_pixel[position_pixel]))
141             complex[position_pixel] = math.sqrt(wm_pixel[position_pix
    el])-diwm
142             if(diwm >= 10):
143                 g = a - (a % 10)
144                 p = g//10
145                 if(p % 2 == 1):
146                     g = g-10+(diwm//10) + (diwm % 10)
```

```
147                 else:
148                     g = g+(diwm//10) + (diwm % 10)
149             else:
150                 g = a - (a % 10)
151                 p = g//10
152                 if(p % 2 == 0):
153                     if(g == 240):
154                         g = g-20
155                     g = g+10+diwm
156                 else:
157                     g = g+diwm
158             g = g*flag
159             if(writedata):
160                 TEMP2[i, j] = g
161             position_pixel += 1
162             if position_pixel == wm_width*wm_height:
163                 break
164         if position_pixel == wm_width*wm_height:
165             break
166     return complex
167
168 # 离散小波反变换
169 def idwt(TEMP2, pixel, width, height):
170     for j in range(height):
171         ct = 0
172         for i in range(width//2):
173             tmp = (TEMP2[i + (width // 2), j]) + 1
174             if ((tmp < 0 or tmp == (-1) or tmp // 2 == (-
    1)) and ((TEMP2[i + (width // 2), j] + 1) % 2) != 0):
175                 pixel[ct, j] = TEMP2[i, j] + (tmp // 2 - 1)
176             else:
177                 pixel[ct, j] = TEMP2[i, j] + tmp // 2
178             pixel[ct + 1, j] = pixel[ct, j] - TEMP2[i + (width // 2),
     j]
179             ct += 2
180     for i in range(width):
181         ct = 0
182         for j in range(height//2):
183             tmp = (pixel[i, j + (height // 2)]) + 1
184             if ((tmp < 0 or tmp == (-1) or tmp // 2 == (-
    1)) and((pixel[i, j + (height // 2)] + 1) % 2) != 0):
185                 TEMP2[i, ct] = pixel[i, j] + (tmp // 2 - 1)
186             else:
187                 TEMP2[i, ct] = pixel[i, j] + tmp // 2
188             TEMP2[i, ct + 1] = TEMP2[i, ct] - pixel[i, j + (height //
     2)]
189             ct += 2
190
191 # 重写线程类，使得其能获取计算结果返回值
192 class MyThread(threading.Thread):
```

```
193     def __init__(self, func, args, name=''):
194         threading.Thread.__init__(self)
195         self.name = name
196         self.func = func
197         self.args = args
198         self.result = self.func(*self.args)
199 
200     def get_result(self):
201         try:
202             return self.result
203         except Exception:
204             return None
205 
206 # 作者信息窗口类
207 class AuInfo(object):
208     def setupUi(self, Dialog):
209         Dialog.setObjectName("Dialog")
210         Dialog.resize(408, 305)
211         self.label_2 = QtWidgets.QLabel(Dialog)
212         self.label_2.setGeometry(QtCore.QRect(90, 100, 271, 51))
213         font = QtGui.QFont()
214         font.setPointSize(18)
215         self.label_2.setFont(font)
216         self.label_2.setObjectName("label_2")
217         self.label_3 = QtWidgets.QLabel(Dialog)
218         self.label_3.setGeometry(QtCore.QRect(80, 160, 251, 16))
219         font = QtGui.QFont()
220         font.setPointSize(10)
221         self.label_3.setFont(font)
222         self.label_3.setObjectName("label_3")
223         self.label_4 = QtWidgets.QLabel(Dialog)
224         self.label_4.setGeometry(QtCore.QRect(10, 200, 391, 16))
225         self.label_4.setObjectName("label_4")
226         self.label_5 = QtWidgets.QLabel(Dialog)
227         self.label_5.setGeometry(QtCore.QRect(10, 220, 371, 20))
228         self.label_5.setObjectName("label_5")
229         self.pushButton = QtWidgets.QPushButton(Dialog)
230         self.pushButton.setGeometry(QtCore.QRect(50, 260, 93, 28))
231         self.pushButton.setObjectName("pushButton")
232         self.pushButton_2 = QtWidgets.QPushButton(Dialog)
233         self.pushButton_2.setGeometry(QtCore.QRect(250, 260, 93, 28))
234         self.pushButton_2.setObjectName("pushButton_2")
235         self.label = QtWidgets.QLabel(Dialog)
236         self.label.setGeometry(QtCore.QRect(40, -40, 291, 191))
237         self.label.setStyleSheet("image:url(:/mark.png)")
238         self.label.setText("")
239         self.label.setObjectName("label")
240 
241         self.retranslateUi(Dialog)
242         self.pushButton.clicked.connect(self.openHome)
```

```
243         self.pushButton_2.clicked.connect(self.openFund)
244         QtCore.QMetaObject.connectSlotsByName(Dialog)
245 
246     def retranslateUi(self, Dialog):
247         _translate = QtCore.QCoreApplication.translate
248         Dialog.setWindowTitle(_translate("Dialog", "关于作者"))
249         Dialog.setWindowFlags(QtCore.Qt.WindowCloseButtonHint)
250         Dialog.setWindowIcon(QIcon('icon.png'))
251         self.label_2.setText(_translate("Dialog", "作者：Hollow Man"))
252         self.label_3.setText(_translate("Dialog", "兰州大学 信息科学与
    工程学院"))
253         self.label_4.setText(_translate(
254             "Dialog", "在这里，我需要感谢蔡銘峯（parkmftsai）等人的研究成
    果，他们的"))
255         self.label_5.setText(_translate(
256             "Dialog", "论文所述算法是此软件所用图片信息隐藏加密算法的基
    础。"))
257         self.pushButton.setText(_translate("Dialog", "我的网站"))
258         self.pushButton_2.setText(_translate("Dialog", "捐助我！"))
259 
260     def openHome(self):
261         if QtGui.QDesktopServices.openUrl(QtCore.QUrl("https://hollow
    man6.github.io/")):
262             pass
263 
264     def openFund(self):
265         if QtGui.QDesktopServices.openUrl(QtCore.QUrl("https://hollow
    man6.github.io/fund.html")):
266             pass
267 
268 # 软件帮助使用说明窗口类
269 class helpw(object):
270     def setupUi(self, Dialog):
271         Dialog.setObjectName("Dialog")
272         Dialog.resize(400, 308)
273         Dialog.setFixedSize(400, 308)
274         self.textBrowser = QtWidgets.QTextBrowser(Dialog)
275         self.textBrowser.setGeometry(QtCore.QRect(10, 10, 381, 241))
276         self.textBrowser.setObjectName("textBrowser")
277         self.pushButton = QtWidgets.QPushButton(Dialog)
278         self.pushButton.setGeometry(QtCore.QRect(160, 260, 93, 28))
279         self.pushButton.setObjectName("pushButton")
280 
281         self.retranslateUi(Dialog)
282         self.pushButton.clicked.connect(Dialog.close)
283         QtCore.QMetaObject.connectSlotsByName(Dialog)
284 
285     def retranslateUi(self, Dialog):
286         _translate = QtCore.QCoreApplication.translate
287         Dialog.setWindowFlags(QtCore.Qt.WindowCloseButtonHint)
```

```
288         Dialog.setWindowIcon(QIcon('icon.png'))
289         Dialog.setWindowTitle(_translate("Dialog", "帮助"))
290         self.textBrowser.setHtml(_translate("Dialog", "<!DOCTYPE HTML
     PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-
    html40/strict.dtd\">\n"
291                                                 "<html><head><meta name=\
    "qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
292                                                 "p, li { white-
    space: pre-wrap; }\n"
293                                                 "</style></head><body sty
    le=\" font-family:\'MS Shell Dlg 2\'; font-size:7.8pt; font-
    weight:400; font-style:normal;\">\n"
294                                                 "<p style=\" margin-
    top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-
    block-indent:0; text-indent:0px;\">· 注意：要隐藏图片尺寸越小，要附加隐藏
    信息的目标图片尺寸越大，效果越好。在给视频进行隐藏信息处理时，由于视频编码的
    压缩，效果可能并不是很明显。在加密时请不要使要隐藏图片的宽和高超过要附加隐藏
    信息的图片宽和高的二分之一。如果你这样做了，虽然不会有任何错误提示，但这会导
    致在隐藏解密时图片还原的必然失败！</p>\n"
295                                                 "<p style=\"-qt-
    paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-
    left:0px; margin-right:0px; -qt-block-indent:0; text-
    indent:0px;\"><br /></p></body></html>"
296                                                 "<p style=\" margin-
    top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-
    block-indent:0; text-indent:0px;\">· 本软件基于蔡銘峯(parkmftsai)等人的
    研究成果，使用整数小波域变换实现图片信息隐藏功能，能够抵御一定各种程度的图片
    压缩以及剪切，添加噪声，旋转等降低图片质量的攻击手段，具有高健壮性，并且加密
    得到的图片与原图片画质几乎一致看不出区别，适用于版权保护或其它特殊领域需要。
    </p>\n"
297                                                 "<p style=\"-qt-
    paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-
    left:0px; margin-right:0px; -qt-block-indent:0; text-
    indent:0px;\"><br /></p>\n"
298                                                 "<p style=\" margin-
    top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-
    block-indent:0; text-indent:0px;\">· 在加密时，你需要设定一个加密密码，你
    还可以选择是否保存图片还原辅助信息，如果选择保存，该信息将会以 npy 文件扩展名
    保存，文件名为图片的文件名（不包括扩展名）。如果选择不保存，则在还原图片时，
    缺少这一信息会导致图片部分细节丢失。如果需要批量加密图片，请在\"要附加隐藏信
    息的图片（文件夹）\"处选择要处理图片所存放的文件夹。</p>\n"
299                                                 "<p style=\"-qt-
    paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-
    left:0px; margin-right:0px; -qt-block-indent:0; text-
    indent:0px;\"><br /></p>\n"
300                                                 "<p style=\" margin-
    top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-
    block-indent:0; text-indent:0px;\">· 在解密时，你需要选择被附加隐藏信息的
    图片，如果存在图片还原辅助信息，你可以选择图片还原辅助信息文件或者文件夹，如
    果你选择的是存放图片还原辅助信息文件文件夹，程序将会自动搜寻该文件夹下和你选
```

```
择的被附加隐藏信息图片同名的文件，如果不存在则不能（在批量状态下）加载图片还原辅助信息。同时，你需要正确地输入被隐藏图片的宽与高，并且输入正确的解密密码。如果你输入的信息与加密时的不符，将会导致解密的失败，即解密出来的图片为一堆杂乱的像素点。如果需要批量解密图片，请在“被附加隐藏信息的图片（文件夹）”处选择要处理图片所存放的文件夹。</p>\n"
301                                         "<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><br /></p>\n"
302                                         "<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\">· 对于视频，你可以选择软件提供的视频处理工具，将视频按帧拆解成图片，然后再对图片进行批量信息隐藏加密。视频拆解出来的图片默认保存为 png 格式,同时自动显示视频帧率 fps，方便再次合成。完成后，你可以对保存的文件夹下的图片进行信息隐藏加密（注意不要改动拆解的原始图片名称，并且在后续加密和合成视频过程中再次使用此文件夹，方便合成视频），此时需要输入原视频的视频帧率（如果软件在拆解视频时已经帮你填好则可以不变），选择图片存放的文件夹（注意不要改动图片加密后的名称）重新合成原视频。默认保存为 avi 格式，XviD 编码。对于视频的隐藏还原，同理，将视频按帧拆解成图片，然后"
303                                         "可以从中随机挑选一张或者更多的图片解密来进行视频的加密隐藏信息的获取。</p>\n"
304                                         "<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><br /></p></body></html>"))
305         self.pushButton.setText(_translate("Dialog", "关闭"))
306
307 # 设定线程最大值窗口类
308 class ThrNum(object):
309     def setupUi(self, Dialog):
310         Dialog.setObjectName("Dialog")
311         Dialog.resize(333, 208)
312         Dialog.setFixedSize(333, 208)
313         self.label = QtWidgets.QLabel(Dialog)
314         self.label.setGeometry(QtCore.QRect(120, 20, 211, 16))
315         self.label.setObjectName("label")
316         self.lineEdit = QtWidgets.QLineEdit(Dialog)
317         self.lineEdit.setGeometry(QtCore.QRect(70, 50, 201, 21))
318         self.lineEdit.setObjectName("lineEdit")
319         self.lineEdit.setText(str(threadNum))
320         self.pushButton = QtWidgets.QPushButton(Dialog)
321         self.pushButton.setGeometry(QtCore.QRect(70, 90, 93, 28))
322         self.pushButton.setObjectName("pushButton")
323         self.pushButton_2 = QtWidgets.QPushButton(Dialog)
324         self.pushButton_2.setGeometry(QtCore.QRect(180, 90, 93, 28))
325         self.pushButton_2.setObjectName("pushButton_2")
326         self.label_4 = QtWidgets.QLabel(Dialog)
327         self.label_4.setGeometry(QtCore.QRect(20, 130, 391, 16))
328         self.label_4.setObjectName("label_4")
329         self.label_5 = QtWidgets.QLabel(Dialog)
```

```
330         self.label_5.setGeometry(QtCore.QRect(90, 140, 421, 21))
331         self.label_5.setObjectName("label_5")
332         self.label_6 = QtWidgets.QLabel(Dialog)
333         self.label_6.setGeometry(QtCore.QRect(70, 170, 421, 21))
334         self.label_6.setObjectName("label_6")
335 
336         self.retranslateUi(Dialog)
337         self.pushButton.clicked.connect(lambda: self.setThr(Dialog))
338         self.pushButton_2.clicked.connect(Dialog.close)
339         QtCore.QMetaObject.connectSlotsByName(Dialog)
340 
341     def setThr(self, Dialog):
342         try:
343             temp = int(self.lineEdit.text())
344             if temp > 0:
345                 threadNum = temp
346                 threadmax = threading.BoundedSemaphore(threadNum)
347                 Dialog.close()
348             else:
349                 QMessageBox.critical(Dialog, "线程设定错误", "请输入一
    个正整数！")
350         except Exception:
351             QMessageBox.critical(Dialog, "线程设定错误", "请输入一个正整
    数！")
352 
353     def retranslateUi(self, Dialog):
354         _translate = QtCore.QCoreApplication.translate
355         Dialog.setWindowFlags(QtCore.Qt.WindowCloseButtonHint)
356         Dialog.setWindowIcon(QIcon('icon.png'))
357         Dialog.setWindowTitle(_translate("Dialog", "设定线程数"))
358         self.label.setText(_translate("Dialog", "最大线程数："))
359         self.pushButton.setText(_translate("Dialog", "确认"))
360         self.pushButton_2.setText(_translate("Dialog", "取消"))
361         self.label_4.setText(_translate("Dialog", "注意："))
362         self.label_5.setText(_translate("Dialog", "仅在批量处理图片时有
    效！"))
363         self.label_6.setText(_translate("Dialog", "请按电脑实际性能设
    置，以免死机！"))
364 
365 # 程序主界面窗口类
366 class Ui_MainWindow(QtCore.QObject):
367     signal = QtCore.pyqtSignal(str)
368     progressChanged = QtCore.pyqtSignal(int)
369     progressChanged2 = QtCore.pyqtSignal(int)
370     progressChanged3 = QtCore.pyqtSignal(int)
371 
372     def setupUi(self, MainWindow):
373         MainWindow.setObjectName("MainWindow")
374         MainWindow.resize(651, 438)
375         MainWindow.setFixedSize(651, 438)
```

```
376         self.cwd = os.getcwd()
377         self.centralwidget = QtWidgets.QWidget(MainWindow)
378         self.centralwidget.setStatusTip("已处理文件：  "+str(countw)+"个
    ")
379         self.centralwidget.setObjectName("centralwidget")
380         self.tabWidget = QtWidgets.QTabWidget(self.centralwidget)
381         self.tabWidget.setGeometry(QtCore.QRect(0, 0, 651, 411))
382         self.tabWidget.setObjectName("tabWidget")
383         self.tab = QtWidgets.QWidget()
384         self.tab.setObjectName("tab")
385         self.lineEdit = QtWidgets.QLineEdit(self.tab)
386         self.lineEdit.setGeometry(QtCore.QRect(10, 50, 491, 21))
387         self.lineEdit.setObjectName("lineEdit")
388         self.lineEdit.setClearButtonEnabled(True)
389         self.checkBox = QtWidgets.QCheckBox(self.tab)
390         self.checkBox.setGeometry(QtCore.QRect(10, 250, 161, 20))
391         self.checkBox.setChecked(True)
392         self.checkBox.setObjectName("checkBox")
393         self.pushButton = QtWidgets.QPushButton(self.tab)
394         self.pushButton.setGeometry(QtCore.QRect(540, 30, 93, 28))
395         self.pushButton.setObjectName("pushButton")
396         self.label = QtWidgets.QLabel(self.tab)
397         self.label.setGeometry(QtCore.QRect(10, 20, 211, 16))
398         self.label.setObjectName("label")
399         self.label_2 = QtWidgets.QLabel(self.tab)
400         self.label_2.setGeometry(QtCore.QRect(10, 100, 91, 16))
401         self.label_2.setObjectName("label_2")
402         self.lineEdit_2 = QtWidgets.QLineEdit(self.tab)
403         self.lineEdit_2.setGeometry(QtCore.QRect(10, 130, 491, 21))
404         self.lineEdit_2.setObjectName("lineEdit_2")
405         self.lineEdit_2.setClearButtonEnabled(True)
406         self.pushButton_2 = QtWidgets.QPushButton(self.tab)
407         self.pushButton_2.setGeometry(QtCore.QRect(540, 130, 93, 28))
408         self.pushButton_2.setObjectName("pushButton_2")
409         self.lineEdit_3 = QtWidgets.QLineEdit(self.tab)
410         self.lineEdit_3.setGeometry(QtCore.QRect(10, 210, 151, 21))
411         self.lineEdit_3.setObjectName("lineEdit_3")
412         self.lineEdit_3.setClearButtonEnabled(True)
413         self.label_3 = QtWidgets.QLabel(self.tab)
414         self.label_3.setGeometry(QtCore.QRect(10, 180, 61, 16))
415         self.label_3.setObjectName("label_3")
416         self.pushButton_3 = QtWidgets.QPushButton(self.tab)
417         self.pushButton_3.setGeometry(QtCore.QRect(360, 200, 141, 61)
    )
418         self.pushButton_3.setObjectName("pushButton_3")
419         self.progressBar = QtWidgets.QProgressBar(self.tab)
420         self.progressBar.setGeometry(QtCore.QRect(10, 310, 621, 23))
421         self.progressBar.setProperty("value", 0)
422         self.progressBar.setObjectName("progressBar")
423         self.label_11 = QtWidgets.QLabel(self.tab)
```

```
424         self.label_11.setGeometry(QtCore.QRect(260, 280, 81, 16))
425         self.label_11.setText("")
426         self.label_11.setObjectName("label_11")
427         self.pushButton_10 = QtWidgets.QPushButton(self.tab)
428         self.pushButton_10.setGeometry(QtCore.QRect(540, 60, 93, 28))
429         self.pushButton_10.setObjectName("pushButton_10")
430         self.tabWidget.addTab(self.tab, "")
431         self.tab_2 = QtWidgets.QWidget()
432         self.tab_2.setObjectName("tab_2")
433         self.label_4 = QtWidgets.QLabel(self.tab_2)
434         self.label_4.setGeometry(QtCore.QRect(10, 20, 241, 16))
435         self.label_4.setObjectName("label_4")
436         self.lineEdit_4 = QtWidgets.QLineEdit(self.tab_2)
437         self.lineEdit_4.setGeometry(QtCore.QRect(10, 50, 491, 21))
438         self.lineEdit_4.setObjectName("lineEdit_4")
439         self.lineEdit_4.setClearButtonEnabled(True)
440         self.pushButton_4 = QtWidgets.QPushButton(self.tab_2)
441         self.pushButton_4.setGeometry(QtCore.QRect(540, 60, 93, 28))
442         self.pushButton_4.setObjectName("pushButton_4")
443         self.label_5 = QtWidgets.QLabel(self.tab_2)
444         self.label_5.setGeometry(QtCore.QRect(200, 180, 91, 16))
445         self.label_5.setObjectName("label_5")
446         self.lineEdit_5 = QtWidgets.QLineEdit(self.tab_2)
447         self.lineEdit_5.setGeometry(QtCore.QRect(200, 200, 91, 21))
448         self.lineEdit_5.setObjectName("lineEdit_5")
449         self.lineEdit_5.setClearButtonEnabled(True)
450         self.label_6 = QtWidgets.QLabel(self.tab_2)
451         self.label_6.setGeometry(QtCore.QRect(200, 230, 91, 16))
452         self.label_6.setObjectName("label_6")
453         self.lineEdit_6 = QtWidgets.QLineEdit(self.tab_2)
454         self.lineEdit_6.setGeometry(QtCore.QRect(200, 250, 91, 21))
455         self.lineEdit_6.setObjectName("lineEdit_6")
456         self.lineEdit_6.setClearButtonEnabled(True)
457         self.label_7 = QtWidgets.QLabel(self.tab_2)
458         self.label_7.setGeometry(QtCore.QRect(10, 180, 101, 16))
459         self.label_7.setObjectName("label_7")
460         self.lineEdit_7 = QtWidgets.QLineEdit(self.tab_2)
461         self.lineEdit_7.setGeometry(QtCore.QRect(10, 210, 151, 21))
462         self.lineEdit_7.setObjectName("lineEdit_7")
463         self.lineEdit_7.setClearButtonEnabled(True)
464         self.checkBox_2 = QtWidgets.QCheckBox(self.tab_2)
465         self.checkBox_2.setGeometry(QtCore.QRect(10, 250, 161, 20))
466         self.checkBox_2.setChecked(True)
467         self.checkBox_2.setObjectName("checkBox_2")
468         self.pushButton_5 = QtWidgets.QPushButton(self.tab_2)
469         self.pushButton_5.setGeometry(QtCore.QRect(360, 200, 141, 61)
    )
470         self.pushButton_5.setObjectName("pushButton_5")
471         self.label_8 = QtWidgets.QLabel(self.tab_2)
472         self.label_8.setGeometry(QtCore.QRect(10, 100, 181, 16))
```

```
473             self.label_8.setObjectName("label_8")
474             self.lineEdit_8 = QtWidgets.QLineEdit(self.tab_2)
475             self.lineEdit_8.setGeometry(QtCore.QRect(10, 130, 491, 21))
476             self.lineEdit_8.setObjectName("lineEdit_8")
477             self.lineEdit_8.setClearButtonEnabled(True)
478             self.progressBar_2 = QtWidgets.QProgressBar(self.tab_2)
479             self.progressBar_2.setGeometry(QtCore.QRect(10, 310, 621, 23)
    )
480             self.progressBar_2.setProperty("value", 0)
481             self.progressBar_2.setObjectName("progressBar_2")
482             self.label_12 = QtWidgets.QLabel(self.tab_2)
483             self.label_12.setGeometry(QtCore.QRect(260, 280, 81, 16))
484             self.label_12.setText("")
485             self.label_12.setObjectName("label_12")
486             self.pushButton_11 = QtWidgets.QPushButton(self.tab_2)
487             self.pushButton_11.setGeometry(QtCore.QRect(540, 30, 93, 28))
488             self.pushButton_11.setObjectName("pushButton_11")
489             self.pushButton_12 = QtWidgets.QPushButton(self.tab_2)
490             self.pushButton_12.setGeometry(QtCore.QRect(540, 110, 93, 28)
    )
491             self.pushButton_12.setObjectName("pushButton_12")
492             self.pushButton_6 = QtWidgets.QPushButton(self.tab_2)
493             self.pushButton_6.setGeometry(QtCore.QRect(540, 140, 93, 28))
494             self.pushButton_6.setObjectName("pushButton_6")
495             self.tabWidget.addTab(self.tab_2, "")
496             self.tab_3 = QtWidgets.QWidget()
497             self.tab_3.setObjectName("tab_3")
498             self.label_9 = QtWidgets.QLabel(self.tab_3)
499             self.label_9.setGeometry(QtCore.QRect(10, 20, 241, 16))
500             self.label_9.setObjectName("label_9")
501             self.label_10 = QtWidgets.QLabel(self.tab_3)
502             self.label_10.setGeometry(QtCore.QRect(10, 100, 250, 16))
503             self.label_10.setObjectName("label_10")
504             self.lineEdit_9 = QtWidgets.QLineEdit(self.tab_3)
505             self.lineEdit_9.setGeometry(QtCore.QRect(10, 50, 491, 21))
506             self.lineEdit_9.setObjectName("lineEdit_9")
507             self.lineEdit_9.setClearButtonEnabled(True)
508             self.pushButton_7 = QtWidgets.QPushButton(self.tab_3)
509             self.pushButton_7.setGeometry(QtCore.QRect(540, 50, 93, 28))
510             self.pushButton_7.setObjectName("pushButton_7")
511             self.lineEdit_10 = QtWidgets.QLineEdit(self.tab_3)
512             self.lineEdit_10.setGeometry(QtCore.QRect(10, 130, 491, 21))
513             self.lineEdit_10.setObjectName("lineEdit_10")
514             self.lineEdit_10.setClearButtonEnabled(True)
515             self.pushButton_8 = QtWidgets.QPushButton(self.tab_3)
516             self.pushButton_8.setGeometry(QtCore.QRect(540, 130, 93, 28))
517             self.pushButton_8.setObjectName("pushButton_8")
518             self.pushButton_9 = QtWidgets.QPushButton(self.tab_3)
519             self.pushButton_9.setGeometry(QtCore.QRect(360, 200, 141, 61)
    )
```

```
520         self.pushButton_9.setObjectName("pushButton_9")
521         self.progressBar_3 = QtWidgets.QProgressBar(self.tab_3)
522         self.progressBar_3.setGeometry(QtCore.QRect(10, 310, 621, 23)
    )
523         self.progressBar_3.setProperty("value", 0)
524         self.progressBar_3.setObjectName("progressBar_3")
525         self.radioButton = QtWidgets.QRadioButton(self.tab_3)
526         self.radioButton.setGeometry(QtCore.QRect(10, 200, 95, 20))
527         self.radioButton.setChecked(True)
528         self.radioButton.setObjectName("radioButton")
529         self.radioButton_2 = QtWidgets.QRadioButton(self.tab_3)
530         self.radioButton_2.setGeometry(QtCore.QRect(10, 240, 95, 20))
531         self.radioButton_2.setObjectName("radioButton_2")
532         self.label_13 = QtWidgets.QLabel(self.tab_3)
533         self.label_13.setGeometry(QtCore.QRect(180, 200, 101, 16))
534         self.label_13.setObjectName("label_13")
535         self.lineEdit_11 = QtWidgets.QLineEdit(self.tab_3)
536         self.lineEdit_11.setGeometry(QtCore.QRect(180, 230, 101, 21))
537         self.lineEdit_11.setObjectName("lineEdit_11")
538         self.lineEdit_11.setClearButtonEnabled(True)
539         self.tabWidget.addTab(self.tab_3, "")
540         MainWindow.setCentralWidget(self.centralwidget)
541         self.menubar = QtWidgets.QMenuBar(MainWindow)
542         self.menubar.setGeometry(QtCore.QRect(0, 0, 651, 26))
543         self.menubar.setObjectName("menubar")
544         self.menu = QtWidgets.QMenu(self.menubar)
545         self.menu.setObjectName("menu")
546         self.menu_2 = QtWidgets.QMenu(self.menu)
547         self.menu_2.setObjectName("menu_2")
548         MainWindow.setMenuBar(self.menubar)
549         self.statusbar = QtWidgets.QStatusBar(MainWindow)
550         self.statusbar.setObjectName("statusbar")
551         MainWindow.setStatusBar(self.statusbar)
552         self.action_3 = QtWidgets.QAction(MainWindow)
553         self.action_3.setObjectName("action_3")
554         self.action_4 = QtWidgets.QAction(MainWindow)
555         self.action_4.setObjectName("action_4")
556         self.action_5 = QtWidgets.QAction(MainWindow)
557         self.action_5.setObjectName("action_5")
558         self.action_6 = QtWidgets.QAction(MainWindow)
559         self.action_6.setObjectName("action_6")
560         self.action_7 = QtWidgets.QAction(MainWindow)
561         self.action_7.setObjectName("action_7")
562         self.label_14 = QtWidgets.QLabel(self.tab_3)
563         self.label_14.setGeometry(QtCore.QRect(260, 280, 81, 16))
564         self.label_14.setText("")
565         self.label_14.setObjectName("label_14")
566         self.menu_2.addAction(self.action_4)
567         self.menu_2.addAction(self.action_6)
568         self.menu_2.addAction(self.action_5)
```

```
569         self.menu.addAction(self.menu_2.menuAction())
570         self.menu.addAction(self.action_7)
571         self.menu.addAction(self.action_3)
572         self.menubar.addAction(self.menu.menuAction())
573 
574         self.retranslateUi(MainWindow)
575         self.tabWidget.setCurrentIndex(0)
576         self.checkBox_2.stateChanged.connect(self.setState)
577         self.radioButton.toggled.connect(self.btnstate)
578         self.radioButton_2.toggled.connect(self.btnstate1)
579         self.pushButton.clicked.connect(self.showPicCho2)
580         self.pushButton_2.clicked.connect(self.showPicCho3)
581         self.pushButton_3.clicked.connect(self.beginencrypt)
582         self.pushButton_4.clicked.connect(self.showDirCho1)
583         self.pushButton_5.clicked.connect(self.begindecrypt)
584         self.pushButton_6.clicked.connect(self.showDirCho2)
585         self.pushButton_7.clicked.connect(self.showMovCho1)
586         self.pushButton_8.clicked.connect(self.showDirCho4)
587         self.pushButton_9.clicked.connect(self.beginvihand)
588         self.pushButton_10.clicked.connect(self.showDirCho3)
589         self.pushButton_11.clicked.connect(self.showPicCho1)
590         self.pushButton_12.clicked.connect(self.showFileCho1)
591         self.action_3.triggered.connect(MainWindow.close)
592         self.action_4.triggered.connect(self.openWeb)
593         self.signal.connect(self.wrongd)
594         self.progressChanged.connect(self.progressBar.setValue)
595         self.progressChanged2.connect(self.progressBar_2.setValue)
596         self.progressChanged3.connect(self.progressBar_3.setValue)
597         QtCore.QMetaObject.connectSlotsByName(MainWindow)
598 
599     def beginencrypt(self):
600         self.progressChanged.emit(0)
601         if self.lineEdit_3.text() == "":
602             QMessageBox.information(MainWindow, '加密密码', '请输入加密
    密码！')
603             return
604         if self.lineEdit.text() == "":
605             QMessageBox.information(MainWindow, '选择', '请选择要附加隐
    藏信息的图片或文件夹！')
606             return
607         if self.lineEdit_2.text() == "":
608             QMessageBox.information(MainWindow, '选择', '请选择要隐藏的
    图片！')
609             return
610         if not os.path.exists(self.lineEdit.text()):
611             QMessageBox.critical(MainWindow, "文件或文件夹不存在",
612                                  "你输入的要附加隐藏信息的图片或文件夹不
    存在！请重新选择！")
613             return
614         if not os.path.exists(self.lineEdit_2.text()):
```

```
615             QMessageBox.critical(MainWindow, "文件不存在", "你输入的要
    隐藏的图片不存在！请重新选择！")
616             return
617         if not os.path.isfile(self.lineEdit_2.text()):
618             QMessageBox.critical(MainWindow, "请选择一个文件",
619                                  "请为要隐藏的图片选择一个图片文件而不是
    文件夹！")
620             return
621         self.label_11.setText("请稍后···")
622         self.pushButton_3.setEnabled(False)
623         self.tab_2.setEnabled(False)
624         self.tab_3.setEnabled(False)
625         t = threading.Thread(target=self.Thren, args=(self.lineEdit.t
    ext(
626         ), self.lineEdit_2.text(), self.lineEdit_3.text(), self.check
    Box.isChecked()))
627         t.setDaemon(True)
628         t.start()
629
630     def begindecrypt(self):
631         self.progressChanged2.emit(0)
632         if self.lineEdit_7.text() == "":
633             QMessageBox.information(MainWindow, '解密密码', '请输入解密
    密码！')
634             return
635         if self.lineEdit_5.text() == "":
636             QMessageBox.information(MainWindow, '被隐藏图片宽', '请输入
    被隐藏图片宽度！')
637             return
638         if self.lineEdit_6.text() == "":
639             QMessageBox.information(MainWindow, '被隐藏图片高', '请输入
    被隐藏图片高度！')
640             return
641         if self.lineEdit_4.text() == "":
642             QMessageBox.information(MainWindow, '选择', '请选择被附加隐
    藏信息的图片或文件夹：')
643             return
644         if self.checkBox_2.isChecked() and self.lineEdit_8.text() ==
    "":
645             QMessageBox.information(MainWindow, '选择', '请选择图片还原
    辅助信息文件或文件夹！')
646             return
647         if not os.path.exists(self.lineEdit_4.text()):
648             QMessageBox.critical(MainWindow, "文件或文件夹不存在",
649                                  "被附加隐藏信息的图片或文件夹不存在！请
    重新选择！")
650             return
651         if self.checkBox_2.isChecked() and not os.path.exists(self.li
    neEdit_8.text()):
652             QMessageBox.critical(MainWindow, "文件或文件夹不存在",
```

```
653                                 "图片还原辅助信息文件或文件夹不存在！请重新选择！")
654                 return
655             try:
656                 temp = int(self.lineEdit_5.text())
657                 if temp > 0:
658                     pass
659                 else:
660                     QMessageBox.critical(MainWindow, "被隐藏图片宽", "请为被隐藏图片宽度输入一个正整数！")
661                     return
662             except Exception:
663                 QMessageBox.critical(MainWindow, "被隐藏图片宽", "请为被隐藏图片宽度输入一个正整数！")
664                 return
665             try:
666                 temp = int(self.lineEdit_6.text())
667                 if temp > 0:
668                     pass
669                 else:
670                     QMessageBox.critical(MainWindow, "被隐藏图片宽", "请为被隐藏图片高度输入一个正整数！")
671                     return
672             except Exception:
673                 QMessageBox.critical(MainWindow, "被隐藏图片宽", "请为被隐藏图片高度输入一个正整数！")
674                 return
675             self.label_12.setText("请稍后···")
676             self.pushButton_5.setEnabled(False)
677             self.tab.setEnabled(False)
678             self.tab_3.setEnabled(False)
679             t = threading.Thread(target=self.Thrde, args=(self.lineEdit_4.text(), self.lineEdit_5.text(
680             ), self.lineEdit_6.text(), self.lineEdit_7.text(), self.checkBox_2.isChecked(), self.lineEdit_8.text()))
681             t.setDaemon(True)
682             t.start()
683 
684         def beginvihand(self):
685             self.progressChanged3.emit(0)
686             if self.radioButton.isChecked():
687                 if self.lineEdit_9.text() == "":
688                     QMessageBox.information(MainWindow, '视频文件', '请输入视频文件！')
689                     return
690                 if self.lineEdit_10.text() == "":
691                     QMessageBox.information(MainWindow, '选择文件夹', '请选择拆解的图片保存文件夹！')
692                     return
693                 if not os.path.exists(self.lineEdit_9.text()):
```

```
694                 QMessageBox.critical(
695                     MainWindow, "文件或文件夹不存在", "你输入的视频文件不存在！请重新选择！")
696                 return
697             if not os.path.isfile(self.lineEdit_9.text()):
698                 QMessageBox.critical(MainWindow, '选择文件', '请输入视频文件而并非一个文件夹！')
699                 return
700             if not os.path.exists(self.lineEdit_10.text()):
701                 QMessageBox.critical(
702                     MainWindow, "文件或文件夹不存在", "你输入的拆解图片保存文件夹不存在！请重新选择！")
703                 return
704             if os.path.isfile(self.lineEdit_10.text()):
705                 QMessageBox.critical(MainWindow, "选择文件夹",
706                                      "请输入拆解图片保存文件夹而并非一个文件！")
707                 return
708             self.label_14.setText("请稍后···")
709             self.pushButton_9.setEnabled(False)
710             self.tab_2.setEnabled(False)
711             self.tab.setEnabled(False)
712             t = threading.Thread(target=self.video2pic, args=(
713                 self.lineEdit_9.text(), self.lineEdit_10.text()))
714             t.setDaemon(True)
715             t.start()
716         elif self.radioButton_2.isChecked():
717             if self.lineEdit_9.text() == "":
718                 QMessageBox.information(
719                     MainWindow, '选择文件夹', '请选择要保存生成视频文件的目标文件夹')
720                 return
721             if self.lineEdit_10.text() == "":
722                 QMessageBox.information(
723                     MainWindow, '选择文件夹', '请选择要合成的图片所在的文件夹！')
724                 return
725             if not os.path.exists(self.lineEdit_9.text()):
726                 QMessageBox.critical(
727                     MainWindow, "文件或文件夹不存在", "你输入的要保存生成视频文件的目标文件夹不存在！请重新选择！")
728                 return
729             if os.path.isfile(self.lineEdit_9.text()):
730                 QMessageBox.critical(MainWindow, "选择文件夹",
731                                      "请输入要保存生成视频文件的目标文件夹而并非一个文件！")
732                 return
733             if not os.path.exists(self.lineEdit_10.text()):
734                 QMessageBox.critical(
```

```
735                     MainWindow, "文件或文件夹不存在", "你输入的要合成的图
    片所在的文件夹不存在！请重新选择！")
736                 return
737             if os.path.isfile(self.lineEdit_10.text()):
738                 QMessageBox.critical(MainWindow, "选择文件夹",
739                                      "请输入要合成的图片所在的文件夹而并
    非一个文件！")
740                 return
741             try:
742                 temp = float(self.lineEdit_11.text())
743                 if temp > 0:
744                     pass
745                 else:
746                     QMessageBox.critical(
747                         MainWindow, "被隐藏图片宽", "请为视频帧率（fps）
    输入一个正整数！")
748                     return
749             except Exception:
750                 QMessageBox.critical(MainWindow, "被隐藏图片宽",
751                                      "请为视频帧率（fps）输入一个正整
    数！")
752                 return
753             self.label_14.setText("请稍后···")
754             self.pushButton_9.setEnabled(False)
755             self.tab_2.setEnabled(False)
756             self.tab.setEnabled(False)
757             t = threading.Thread(target=self.pic2video, args=(
758                 self.lineEdit_9.text(), self.lineEdit_10.text(), floa
    t(self.lineEdit_11.text())))
759             t.setDaemon(True)
760             t.start()
761
762     def Thren(self, hidepath, markpath, passwd, info):
763         flist = []
764         if os.path.isfile(hidepath):
765             flist.append(hidepath)
766         else:
767             listl = os.listdir(hidepath)
768             for i in range(len(listl)):
769                 path = os.path.join(hidepath, listl[i])
770                 if os.path.isfile(path):
771                     flist.append(path)
772         l = []
773         for i in flist:
774             filepath, tempfilename = os.path.split(i)
775             savefile = os.path.join(
776                 filepath, "ec"+os.path.splitext(tempfilename)[0]+".pn
    g")
777             threadmax.acquire()
778             t = threading.Thread(target=self.encrypt, args=(
```

```
779                 i, os.path.abspath(markpath), savefile, passwd, info)
    )
780             t.setDaemon(True)
781             t.start()
782             l.append(t)
783         for t in l:
784             t.join()
785         self.label_11.setText("")
786         self.pushButton_3.setEnabled(True)
787         self.tab_2.setEnabled(True)
788         self.tab_3.setEnabled(True)
789 
790     def Thrde(self, entimage, mwidth, mheight, passwd, NKey, keyfile)
    :
791         flist = []
792         if os.path.isfile(entimage):
793             flist.append(entimage)
794         else:
795             listl = os.listdir(entimage)
796             for i in range(len(listl)):
797                 path = os.path.join(entimage, listl[i])
798                 if os.path.isfile(path) and os.path.splitext(path)[1]
     != '.npy':
799                     flist.append(path)
800         l = []
801         for i in flist:
802             filepath, tempfilename = os.path.split(i)
803             realfile = ""
804             if NKey == True:
805                 if not os.path.isfile(keyfile) and os.path.isfile(os.
    path.abspath(keyfile)+"/"+os.path.splitext(tempfilename)[0]+'.npy'):
806                     realfile = os.path.abspath(
807                         keyfile)+"/"+os.path.splitext(tempfilename)[0
    ]+'.npy'
808                 elif os.path.isfile(keyfile):
809                     realfile = os.path.abspath(keyfile)
810                 else:
811                     self.signal.emit("在指定文件夹中无法找到
    "+tempfilename +
812                                      "的图片还原辅助信息，请确保指定文件
    夹下存在"+os.path.splitext(tempfilename)[0]+'.npy 文件！ ')
813                     continue
814             savefile = os.path.join(
815                 filepath, "de"+os.path.splitext(tempfilename)[0]+".pn
    g")
816             threadmax.acquire()
817             t = threading.Thread(target=self.decrypt, args=(
818                 i, savefile, int(mwidth), int(mheight), passwd, NKey,
     realfile))
819             t.setDaemon(True)
```

```
820             t.start()
821             l.append(t)
822         for t in l:
823             t.join()
824         self.label_12.setText("")
825         self.pushButton_5.setEnabled(True)
826         self.tab.setEnabled(True)
827         self.tab_3.setEnabled(True)
828 
829     def btnstate(self):
830         self.label_9.setText("选择视频文件：")
831         self.label_10.setText("选择拆解的图片保存文件夹：")
832         self.lineEdit_9.setText("")
833         self.lineEdit_10.setText("")
834 
835     def btnstate1(self):
836         self.label_9.setText("选择要保存生成视频文件的目标文件夹：")
837         self.label_10.setText("选择要合成的图片所在的文件夹：")
838         self.lineEdit_9.setText("")
839         self.lineEdit_10.setText("")
840 
841     def setState(self):
842         if self.checkBox_2.isChecked():
843             self.lineEdit_8.setEnabled(True)
844             self.pushButton_12.setEnabled(True)
845             self.pushButton_6.setEnabled(True)
846         else:
847             self.lineEdit_8.setEnabled(False)
848             self.pushButton_12.setEnabled(False)
849             self.pushButton_6.setEnabled(False)
850 
851     def wrongd(self, info):
852         QMessageBox.critical(MainWindow, "错误", info)
853 
854     def openWeb(self):
855         if QtGui.QDesktopServices.openUrl(QtCore.QUrl("https://raw.gi
    thubusercontent.com/parkmftsai/digital_watermarking/master/paper/High-
    capacity%20Robust%20Watermarking%20Approach%20for%20Protecting%20Owner
    ship%20Right.pdf")):
856             pass
857 
858     def showPicCho1(self):
859         fileName_choose, filetype = QFileDialog.getOpenFileName(
860             None,  "选取图片",  self.cwd, "图片文
    件 (*.png;*.jpg;*.jpeg;*.bmp;*.dib;*.jpeg;*.jpe;*.tif;*.tiff)")
861         if fileName_choose == "":
862             return
863         else:
864             self.lineEdit_4.setText(fileName_choose)
865 
```

```
866     def showPicCho3(self):
867         fileName_choose, filetype = QFileDialog.getOpenFileName(
868             None,  "选取图片",  self.cwd, "图片文
    件 (*.png;*.jpg;*.jpeg;*.bmp;*.dib;*.jpeg;*.jpe;*.tif;*.tiff)")
869         if fileName_choose == "":
870             return
871         else:
872             self.lineEdit_2.setText(fileName_choose)
873 
874     def showPicCho2(self):
875         fileName_choose, filetype = QFileDialog.getOpenFileName(
876             None,  "选取图片",  self.cwd, "图片文
    件 (*.png;*.jpg;*.jpeg;*.bmp;*.dib;*.jpeg;*.jpe;*.tif;*.tiff)")
877         if fileName_choose == "":
878             return
879         else:
880             self.lineEdit.setText(fileName_choose)
881 
882     def showDirCho1(self):
883         dir_choose = QFileDialog.getExistingDirectory(None, "选取文件
    夹", self.cwd)
884         if dir_choose == "":
885             return
886         else:
887             self.lineEdit_4.setText(dir_choose)
888 
889     def showFileCho1(self):
890         fileName_choose, filetype = QFileDialog.getOpenFileName(
891             None,  "选取图片",  self.cwd, "文件 (*.npy)")
892         if fileName_choose == "":
893             return
894         else:
895             self.lineEdit_8.setText(fileName_choose)
896 
897     def showMovCho1(self):
898         if self.radioButton.isChecked():
899             fileName_choose, filetype = QFileDialog.getOpenFileName(
900                 None,  "选取视频",  self.cwd, "视频文
    件 (*.mp4;*.avi;*.mkv)")
901             if fileName_choose == "":
902                 return
903             else:
904                 self.lineEdit_9.setText(fileName_choose)
905         if self.radioButton_2.isChecked():
906             dir_choose = QFileDialog.getExistingDirectory(
907                 None, "选取文件夹", self.cwd)
908             if dir_choose == "":
909                 return
910             else:
911                 self.lineEdit_9.setText(dir_choose)
```

```
912
913     def showDirCho2(self):
914         dir_choose = QFileDialog.getExistingDirectory(None, "选取文件
    夹", self.cwd)
915         if dir_choose == "":
916             return
917         else:
918             self.lineEdit_8.setText(dir_choose)
919
920     def showDirCho3(self):
921         dir_choose = QFileDialog.getExistingDirectory(None, "选取文件
    夹", self.cwd)
922         if dir_choose == "":
923             return
924         else:
925             self.lineEdit.setText(dir_choose)
926
927     def showDirCho4(self):
928         dir_choose = QFileDialog.getExistingDirectory(None, "选取文件
    夹", self.cwd)
929         if dir_choose == "":
930             return
931         else:
932             self.lineEdit_10.setText(dir_choose)
933
934     def retranslateUi(self, MainWindow):
935         _translate = QtCore.QCoreApplication.translate
936         MainWindow.setWindowTitle(_translate(
937             "MainWindow", "Hollow 图片（视频）信息隐藏加解密软件 V1.0"))
938         MainWindow.setWindowIcon(QIcon('icon.png'))
939         self.checkBox.setText(_translate("MainWindow", "保存图片还原辅
    助信息"))
940         self.pushButton.setText(_translate("MainWindow", "选择图片"))
941         self.label.setText(_translate("MainWindow", "要附加隐藏信息的图
    片（文件夹）："))
942         self.label_2.setText(_translate("MainWindow", "要隐藏的图片：
    "))
943         self.pushButton_2.setText(_translate("MainWindow", "选择图片
    "))
944         self.label_3.setText(_translate("MainWindow", "加密密码："))
945         self.pushButton_3.setText(_translate("MainWindow", "开始加密！
    "))
946         self.pushButton_10.setText(_translate("MainWindow", "选择文件
    夹"))
947         self.tabWidget.setTabText(self.tabWidget.indexOf(
948             self.tab), _translate("MainWindow", "隐藏加密"))
949         self.label_4.setText(_translate("MainWindow", "选择被附加隐藏信
    息的图片（文件夹）："))
950         self.pushButton_4.setText(_translate("MainWindow", "选择文件夹
    "))
```

```
951         self.label_5.setText(_translate("MainWindow", "被隐藏图片宽：
    "))
952         self.label_6.setText(_translate("MainWindow", "被隐藏图片高：
    "))
953         self.label_7.setText(_translate("MainWindow", "解密密码："))
954         self.checkBox_2.setText(_translate("MainWindow", "有图片还原辅
    助信息"))
955         self.pushButton_5.setText(_translate("MainWindow", "开始解密！
    "))
956         self.label_8.setText(_translate("MainWindow", "图片还原辅助信息
    文件（夹）："))
957         self.pushButton_11.setText(_translate("MainWindow", "选择图片
    "))
958         self.pushButton_12.setText(_translate("MainWindow", "选择文件
    "))
959         self.pushButton_6.setText(_translate("MainWindow", "选择文件夹
    "))
960         self.tabWidget.setTabText(self.tabWidget.indexOf(
961             self.tab_2), _translate("MainWindow", "隐藏解密"))
962         self.label_9.setText(_translate("MainWindow", "选择视频文件：
    "))
963         self.label_10.setText(_translate("MainWindow", "选择拆解的图片
    保存文件夹："))
964         self.pushButton_7.setText(_translate("MainWindow", "浏览"))
965         self.pushButton_8.setText(_translate("MainWindow", "浏览"))
966         self.pushButton_9.setText(_translate("MainWindow", "开始！"))
967         self.radioButton.setText(_translate("MainWindow", "视频拆解"))
968         self.radioButton_2.setText(_translate("MainWindow", "视频合成
    "))
969         self.label_13.setText(_translate("MainWindow", "视频帧率
    （fps）："))
970         self.tabWidget.setTabText(self.tabWidget.indexOf(
971             self.tab_3), _translate("MainWindow", "视频处理"))
972         self.menu.setTitle(_translate("MainWindow", "选项"))
973         self.menu_2.setTitle(_translate("MainWindow", "关于"))
974         self.action_3.setText(_translate("MainWindow", "退出"))
975         self.action_4.setText(_translate("MainWindow", "原理"))
976         self.action_5.setText(_translate("MainWindow", "作者"))
977         self.action_6.setText(_translate("MainWindow", "帮助"))
978         self.action_7.setText(_translate("MainWindow", "设定线程数"))
979 
980     def encrypt(self, srcimage, markimage, outimage, passwd, NKey):
981         global countw, threadmax, lock
982         try:
983             lock.acquire()
984             source = Image.open(srcimage)
985             if len(source.split()) < 3:
986                 source = source.convert("RGB")
987             self.progressChanged.emit(10)
988             img = np.array(source)
```

```
989             width, height = source.size
990             lock.release()
991         except Exception:
992             lock.release()
993             self.signal.emit(srcimage+"的图片格式不受支持！")
994             self.progressChanged.emit(0)
995             threadmax.release()
996             return
997         pixelred = np.zeros((img.shape[0], img.shape[1]), int)
998         pixelgreen = np.zeros((img.shape[0], img.shape[1]), int)
999         pixelblue = np.zeros((img.shape[0], img.shape[1]), int)
1000         pixelred[:, :] = img[:, :, 0]
1001         pixelred = pixelred.T
1002         pixelgreen[:, :] = img[:, :, 1]
1003         pixelgreen = pixelgreen.T
1004         pixelblue[:, :] = img[:, :, 2]
1005         pixelblue = pixelblue.T
1006         threads = []
1007         for i in [pixelred, pixelgreen, pixelblue]:
1008             t = MyThread(dwt, (i, width, height))
1009             threads.append(t)
1010         for i in range(3):
1011             threads[i].start()
1012         for i in range(3):
1013             threads[i].join()
1014         self.progressChanged.emit(20)
1015         dwtred = threads[0].get_result()
1016         dwtgreen = threads[1].get_result()
1017         dwtblue = threads[2].get_result()
1018         try:
1019             lock.acquire()
1020             mark = Image.open(markimage)
1021             if len(mark.split()) < 3:
1022                 mark = mark.convert("RGB")
1023             mwidth, mheight = mark.size
1024             lock.release()
1025         except Exception:
1026             lock.release()
1027             self.signal.emit(markimage+"的图片格式不受支持！")
1028             self.progressChanged.emit(0)
1029             threadmax.release()
1030             return
1031         threads = []
1032         flag = 0
1033         self.progressChanged.emit(30)
1034         lock.acquire()
1035         for i in [dwtred, dwtgreen, dwtblue]:
1036             t = MyThread(hiding_data, (i, markimage, flag, True, pas
     swd))
1037             flag += 1
```

```
1038            threads.append(t)
1039        for i in range(3):
1040            threads[i].start()
1041        for i in range(3):
1042            threads[i].join()
1043        lock.release()
1044        self.progressChanged.emit(40)
1045        complex1 = threads[0].get_result()
1046        complex2 = threads[1].get_result()
1047        complex3 = threads[2].get_result()
1048        pixel = np.zeros((width, height), int)
1049        self.progressChanged.emit(50)
1050        idwt(dwtred, pixel, width, height)
1051        self.progressChanged.emit(65)
1052        idwt(dwtgreen, pixel, width, height)
1053        self.progressChanged.emit(78)
1054        idwt(dwtblue, pixel, width, height)
1055        self.progressChanged.emit(90)
1056        output_file(outimage, dwtred, dwtgreen, dwtblue, width, heig
    ht)
1057        if NKey == True:
1058            complexd = np.zeros((mwidth*mheight, 3), float)
1059            complexd[:, 0] = complex1[:]
1060            complexd[:, 1] = complex2[:]
1061            complexd[:, 2] = complex3[:]
1062            np.save(os.path.splitext(outimage)[0], complexd)
1063        countw += 1
1064        self.progressChanged.emit(100)
1065        self.centralwidget.setStatusTip("已处理文件："+str(countw)+"个
    ")
1066        threadmax.release()
1067
1068    def decrypt(self, entimage, outimage, mwidth, mheight, passwd, N
    Key, keyfile=''):
1069        global countw, threadmax, lock
1070        complex1 = 0
1071        complex2 = 0
1072        complex3 = 0
1073        if NKey == True:
1074            try:
1075                lock.acquire()
1076                complexd = np.load(keyfile)
1077                lock.release()
1078            except Exception:
1079                lock.release()
1080                self.progressChanged2.emit(0)
1081                self.signal.emit(keyfile+"图片还原辅助信息加载错误，请
    确认是否为正确的文件！")
1082                threadmax.release()
1083                return
```

```
1084            complex1 = np.zeros((complexd.shape[0]), float)
1085            complex2 = np.zeros((complexd.shape[0]), float)
1086            complex3 = np.zeros((complexd.shape[0]), float)
1087            complex1[:] = complexd[:, 0]
1088            complex2[:] = complexd[:, 1]
1089            complex3[:] = complexd[:, 2]
1090        else:
1091            complex1 = np.zeros((mwidth*mheight), float)
1092            complex2 = np.zeros((mwidth*mheight), float)
1093            complex3 = np.zeros((mwidth*mheight), float)
1094        self.progressChanged2.emit(10)
1095        try:
1096            lock.acquire()
1097            source = Image.open(entimage)
1098            if len(source.split()) < 3:
1099                source = source.convert("RGB")
1100            img = np.array(source)
1101            lock.release()
1102        except Exception:
1103            lock.release()
1104            self.progressChanged2.emit(0)
1105            self.signal.emit(entimage+"的图片格式不受支持！")
1106            threadmax.release()
1107            return
1108        self.progressChanged2.emit(10)
1109        width, height = source.size
1110        pixelred = np.zeros((img.shape[0], img.shape[1]), int)
1111        pixelgreen = np.zeros((img.shape[0], img.shape[1]), int)
1112        pixelblue = np.zeros((img.shape[0], img.shape[1]), int)
1113        pixelred[:, :] = img[:, :, 0]
1114        pixelred = pixelred.T
1115        pixelgreen[:, :] = img[:, :, 1]
1116        pixelgreen = pixelgreen.T
1117        pixelblue[:, :] = img[:, :, 2]
1118        pixelblue = pixelblue.T
1119        self.progressChanged2.emit(20)
1120        outdata = Image.new("RGB", (mwidth, mheight))
1121        threads = []
1122        for i in [pixelred, pixelgreen, pixelblue]:
1123            t = MyThread(dwt, (i, width, height))
1124            threads.append(t)
1125        for i in range(3):
1126            threads[i].start()
1127        for i in range(3):
1128            threads[i].join()
1129        self.progressChanged2.emit(50)
1130        dwtred = threads[0].get_result()
1131        dwtgreen = threads[1].get_result()
1132        dwtblue = threads[2].get_result()
1133        threads = []
```

```
1134            l = [complex1, complex2, complex3]
1135            count = 0
1136            lock.acquire()
1137            try:
1138                for i in [dwtred, dwtgreen, dwtblue]:
1139                    t = MyThread(extract_watermark, (i, width, height,
1140                                                     mwidth, mheight, l[
    count], passwd))
1141                    count += 1
1142                    threads.append(t)
1143                for i in range(3):
1144                    threads[i].start()
1145                for i in range(3):
1146                    threads[i].join()
1147                lock.release()
1148            except Exception:
1149                lock.release()
1150                self.progressChanged2.emit(0)
1151                self.signal.emit(entimage+"图片还原辅助信息与实际不符！")
1152                threadmax.release()
1153                return
1154            self.progressChanged2.emit(70)
1155            red_wm = threads[0].get_result()
1156            green_wm = threads[1].get_result()
1157            blue_wm = threads[2].get_result()
1158            position_pixel = 0
1159            self.progressChanged2.emit(90)
1160            for i in range(mwidth):
1161                for j in range(mheight):
1162                    outdata.putpixel(
1163                        (i, j), (red_wm[position_pixel], green_wm[positi
    on_pixel], blue_wm[position_pixel]))
1164                    position_pixel += 1
1165            outdata.save(outimage)
1166            countw += 1
1167            self.progressChanged2.emit(100)
1168            self.centralwidget.setStatusTip("已处理文件："+str(countw)+"个
    ")
1169            threadmax.release()
1170
1171        def video2pic(self, video, pic_path):
1172            global countw
1173            try:
1174                vc = cv2.VideoCapture(video)
1175                c = 0
1176                rval = vc.isOpened()
1177                while rval:
1178                    c = c + 1
1179                    rval, frame = vc.read()
1180                    if rval:
```

```
1181                        cv2.imencode('.png', frame)[1].tofile(
1182                            pic_path + '/'+str(c) + '.png')
1183                        cv2.waitKey(1)
1184                    else:
1185                        break
1186                self.lineEdit_11.setText(str(vc.get(cv2.CAP_PROP_FPS)))
1187                vc.release()
1188            except Exception:
1189                self.progressChanged2.emit(0)
1190                self.signal.emit(video+"的格式不受支持！")
1191            self.label_14.setText("")
1192            self.pushButton_9.setEnabled(True)
1193            self.tab_2.setEnabled(True)
1194            self.tab.setEnabled(True)
1195            countw += 1
1196            self.progressChanged3.emit(100)
1197            self.centralwidget.setStatusTip("已处理文件："+str(countw)+"个
    ")
1198
1199        def pic2video(self, savepath, path, fps):
1200            global countw
1201            try:
1202                file_path = os.path.abspath(
1203                    savepath) + '/' + str(int(time.time())) + ".avi"
1204                fourcc = cv2.VideoWriter_fourcc('X', 'V', 'I', 'D')
1205                size = (0, 0)
1206                item = os.path.abspath(path) + '/ec1.png'
1207                img = cv2.imdecode(np.fromfile(
1208                    item, dtype=np.uint8), cv2.IMREAD_COLOR)
1209                if img.shape[0] < img.shape[1]:
1210                    size = (img.shape[1], img.shape[0])
1211                else:
1212                    size = img.shape[:2]
1213                video = cv2.VideoWriter(file_path, fourcc, fps, size)
1214                i = 1
1215                while os.path.isfile(os.path.abspath(path) + '/ec' + str
    (i)+'.png'):
1216                    item = os.path.abspath(path) + '/ec' + str(i)+'.png'
1217                    img = cv2.imdecode(np.fromfile(
1218                        item, dtype=np.uint8), cv2.IMREAD_COLOR)
1219                    video.write(img)
1220                    i += 1
1221                video.release()
1222            except Exception:
1223                self.progressChanged2.emit(0)
1224                self.signal.emit(
1225                    "无法找到图片，请确认文件夹下存放着文件名为‘ecX.png’（X
    为连续的从 1 开始的数字）的图片集！")
1226            self.label_14.setText("")
1227            self.pushButton_9.setEnabled(True)
```

```
1228            self.tab_2.setEnabled(True)
1229            self.tab.setEnabled(True)
1230            countw += 1
1231            self.progressChanged3.emit(100)
1232            self.centralwidget.setStatusTip("已处理文件：“+str(countw)+"个
    ")
1233 if __name__ == "__main__":
1234     threadNum = 4
1235     threadmax = threading.BoundedSemaphore(threadNum)
1236     lock = threading.Lock()
1237     countw = 0
1238     app = QtWidgets.QApplication(sys.argv)
1239     MainWindow = QtWidgets.QMainWindow()
1240     ui = Ui_MainWindow()
1241     ui.setupUi(MainWindow)
1242     helpwd = QtWidgets.QDialog()
1243     ThrNumd = QtWidgets.QDialog()
1244     AuInford = QtWidgets.QDialog()
1245     helpwr = helpw()
1246     helpwr.setupUi(helpwd)
1247     ThrNumr = ThrNum()
1248     ThrNumr.setupUi(ThrNumd)
1249     AuInfor = AuInfo()
1250     AuInfor.setupUi(AuInford)
1251     MainWindow.show()
1252     ui.action_5.triggered.connect(AuInford.show)
1253     ui.action_6.triggered.connect(helpwd.show)
1254     ui.action_7.triggered.connect(ThrNumd.show)
1255     sys.exit(app.exec_())
```