Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Курсовая работа

Дисциплина: Методы и средства цифровой обработки информации

Выполнили
студенты гр. 13541/2 _____ Зобков Д. А.
(подпись)

_____ Костусев В. А.
(подпись)

Преподаватель _____ Абрамов Н. А.
(подпись)

“___”_____ 2017 г.

Санкт-Петербург
2017 г.

СОДЕРЖАНИЕ

1 Техническое задание

- Подобрать видео с движущимся объектом;
- Обозначить координаты объекта на первом кадре;
- Определить координаты объекта на последующих кадрах с использованием векторов смещения;
- Убедиться в корректности найденного объекта с помощью сравнения по гистограмме и отобразить кадры с обнаруженным (или потерянным) объектом.

2 Ход работы

Для выполнения работы были разработаны классы ObjectRectangle (обозначение координат объекта на первом кадре) и Tracker (определение координат объекта на последующих кадрах) с использованием библиотек OpenCV и Matplotlib (листинг А.1 на с. 9), а также классы графического интерфейса MainWindow (основное окно) и VideoWindow (простейший видеоплеер) с использованием библиотеки PyQt5 (листинг А.2) на языке Python. Использованные зависимости для virtualenv приведены в листинге А.3.

2.1 Описание алгоритма

1. Определяем координаты объекта (NxM);
2. Для каждого последующего кадра видео производится обход некоторой окрестности объекта (2Nx2M) в предыдущем кадре в поиске максимального соответствия (с помощью метрик SAD, SSD);
3. Таким образом, мы получаем смещения объекта, указывающие его „движение“ между кадрами. С их помощью устанавливаем координаты текущего расположения объекта.
4. Корректность найденного смещения проверяется с помощью сравнения гистограмм объекта предыдущего кадра и текущего (допускается совпадение не менее 50% гистограммы).

$$SAD(d_1,d_2) = \sum_{i=0}^{n1}\sum_{j=0}^{n2}(d_1[i,j] - d_2[i,j])$$

$$SSD(d_1,d_2) = \sum_{i=0}^{n1}\sum_{j=0}^{n2}(d_1[i,j] - d_2[i,j])^2$$

$$HI = \sum_{i=0}^{255}\min(I_{t-1}^i, I_t^i)$$

2.2 Работа приложения

Работа приложения демонстрируется на двух видеозаписях, начальные кадры которых приведены на рис. 2.1–2.2 на страницах 4–5, а объект выделен черной рамкой. Приложение отслеживало объект в течение 20-ти кадров (если объект не был потерян).

Рис. 2.1. Пример №1: первый кадр

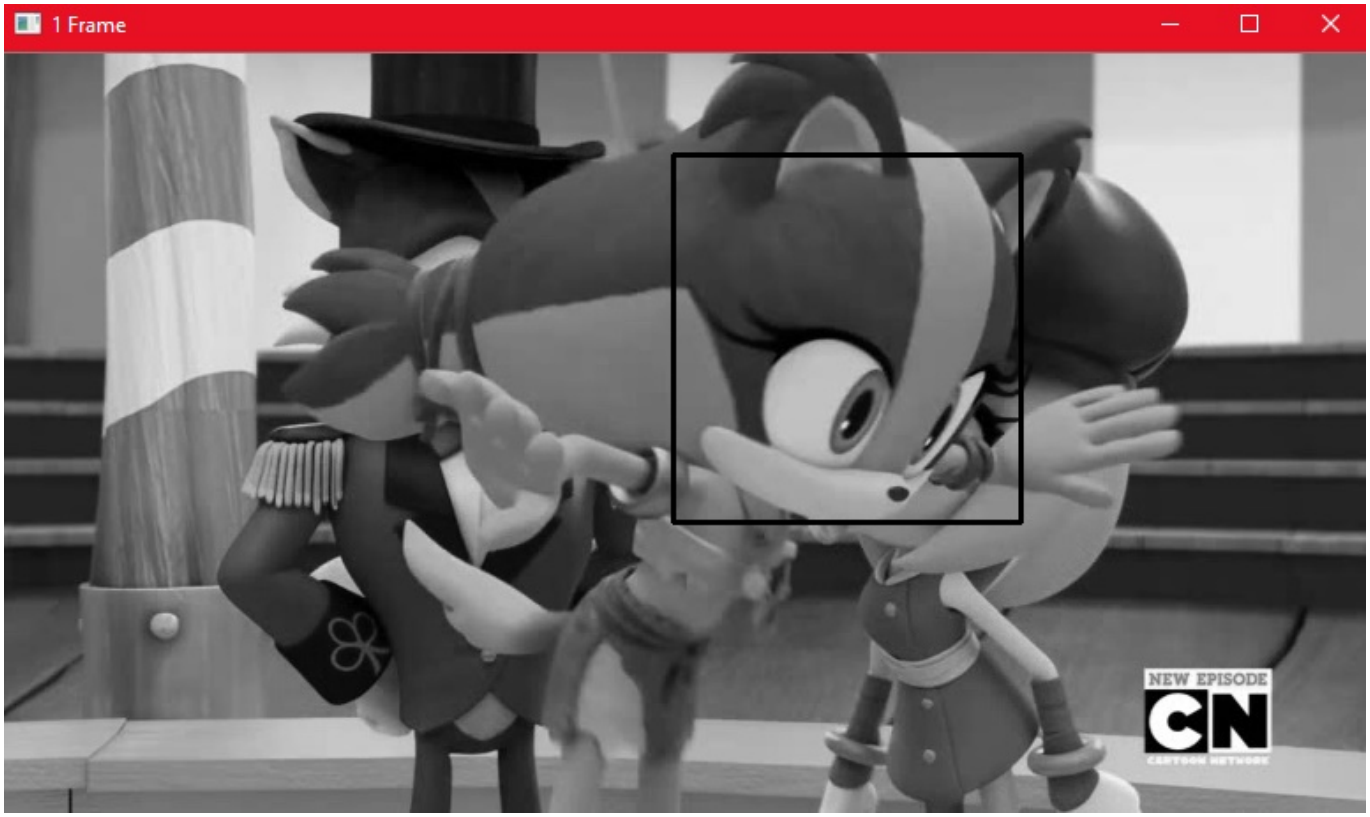

Рис. 2.2. Пример №2: первый кадр

Приведем некоторые кадры для каждого из примеров (для примера №1 — рис. 2.3–2.5 на страницах 5–6, для примера №2 — рис. 2.6–2.7 на с. 7).

Рис. 2.3. Пример №1: 12-ый кадр

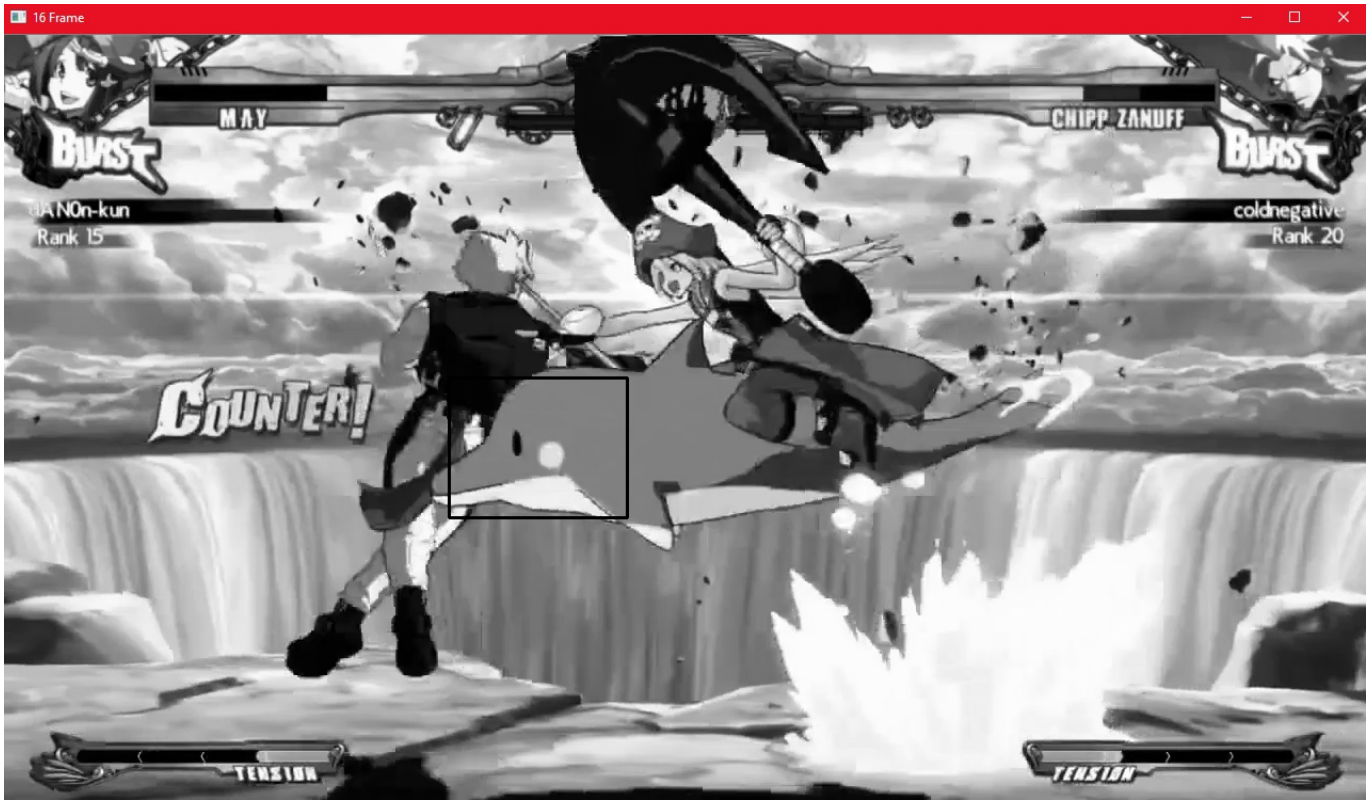

Рис. 2.4. Пример №1: 16-ый кадр

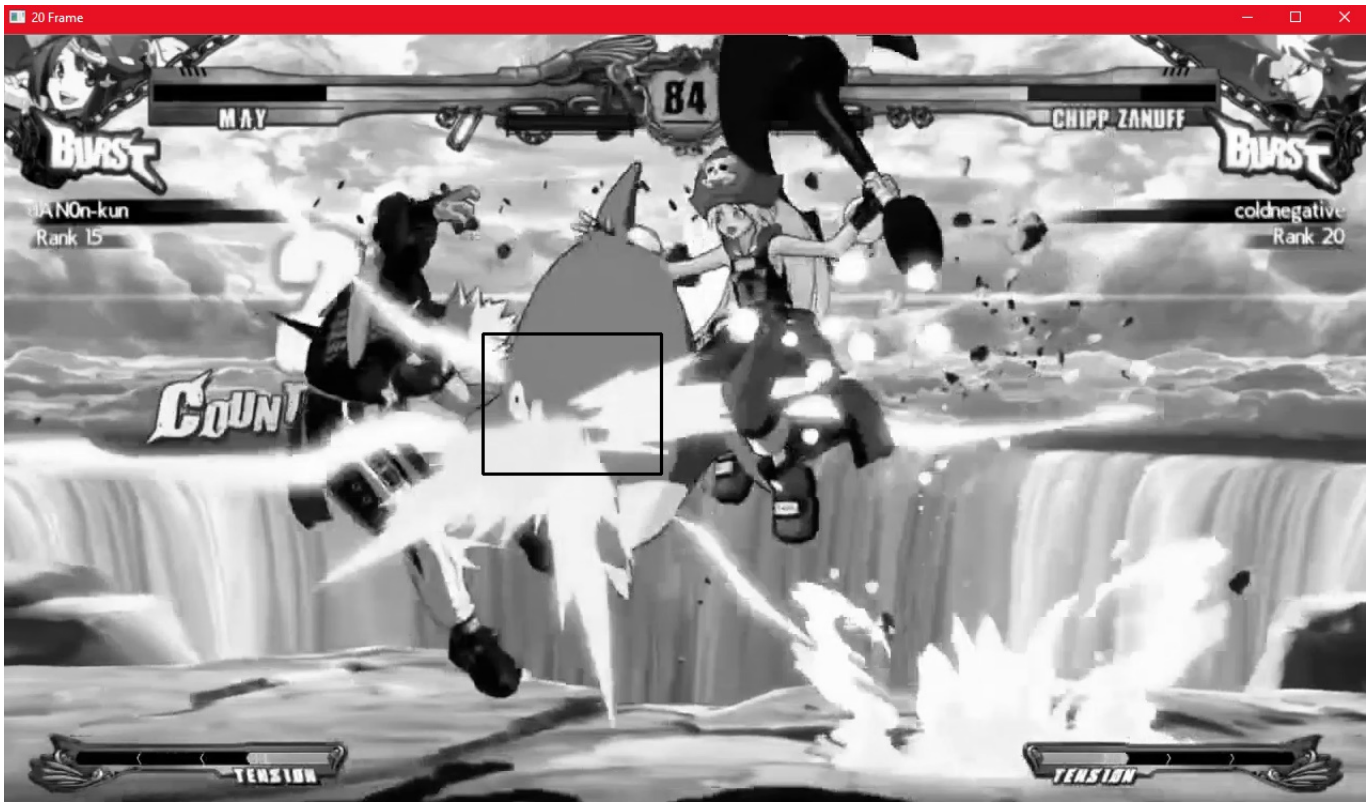

Рис. 2.5. Пример №1: 20-ый кадр

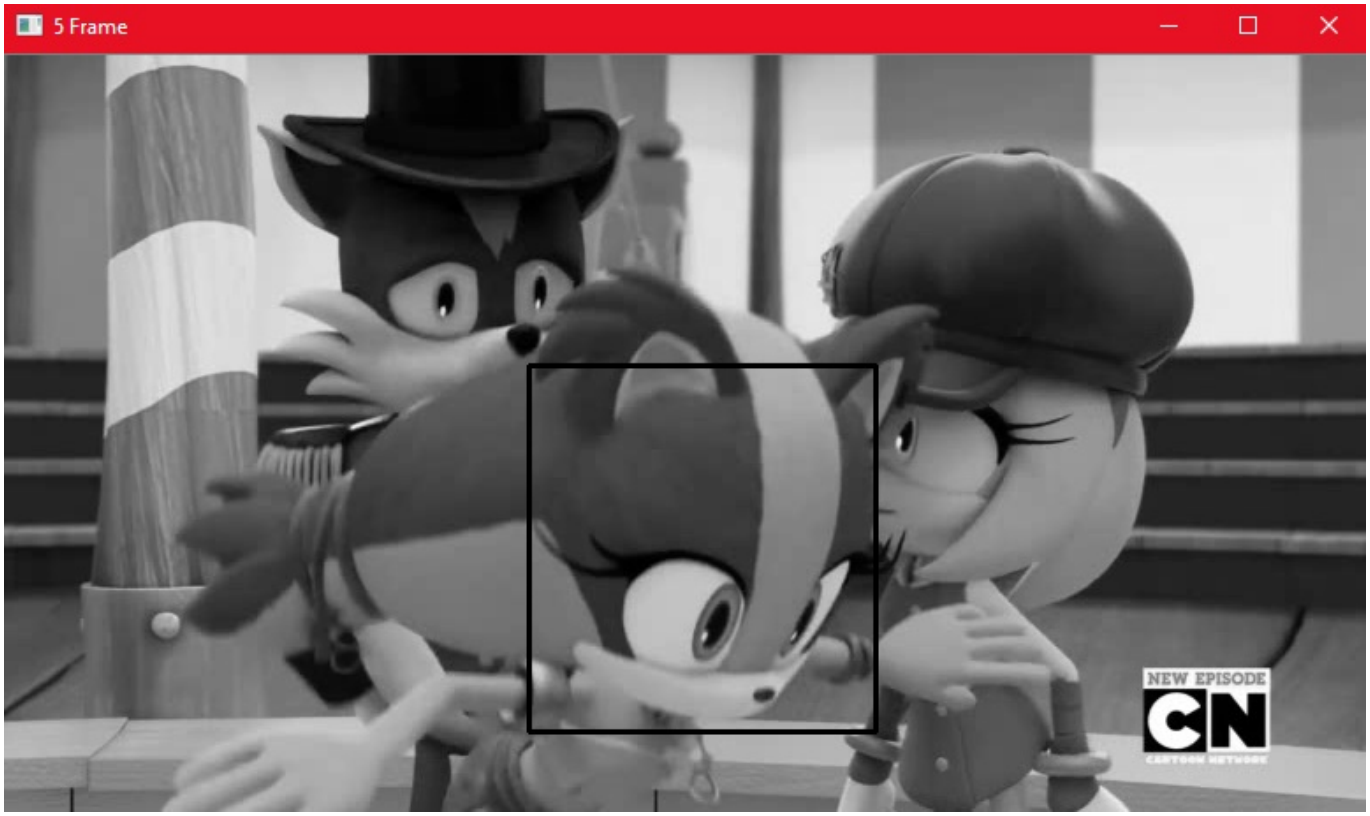

Рис. 2.6. Пример №2: 5-ый кадр

Рис. 2.7. Пример №2: 6-ой кадр (объект потерян)

В случае кадра на рис. 2.7 объект был потерян из-за его удаленности от области поиска (объект находится в ней лишь частично).

3 Выводы

В данной работе было проведено отслеживание движения объекта с использованием метрики максимального соответствия и проверкой корректности обнаруженного объекта с помощью сравнения гистограмм.

В результате было обнаружено, что полученная реализация весьма качественно находит двигающийся объект. Однако в случае его быстрого перемещения установленного окна поиска в 2Nx2M оказывалось недостаточно для обнаружения. Это можно исправить увеличением окна поиска, но вместе с этим возрастет время поиска объекта и шанс обнаружения некорректного объекта.

В случае обнаружения некорректных объектов, отсеять их помогает сравнение гистограмм, но данный метод плохо работает при сильно деформирующихся объектах. В такой ситуации необходимо понизить границу допустимого совпадения гистограмм, однако вместе с этим растет риск обнаружения ложных объектов.

Таким образом, качество работы отслеживания объекта зависит от окна поиска и границы допустимого совпадения гистограмм, которые невозможно настроить универсально. Поэтому их настройка должна проводиться в зависимости от условий, в которых проводится отслеживание.

Приложение А Исходный код программы

```
import random
import numpy as np
import cv2

class ObjectRectangle:
    def __init__(self, path, time):
        self.ix, self.iy = -1, -1
        self.drawing = False
        video = cv2.VideoCapture(path)
        video.set(0, time)

        success, self.img = video.read()
        self.drawImg = self.img + 0

        self.coord = [-1, -1, -1, -1]

    def showFrame(self):
        cv2.namedWindow('1st Frame')
        cv2.setMouseCallback('1st Frame', self.draw)

        while(1):
            cv2.imshow('1st Frame', self.drawImg)
            k = cv2.waitKey(1) & 0xFF
            if k == 27:
                cv2.destroyAllWindows()
                break

    def draw(self, event, x, y, flags, param):
        if event == cv2.EVENT_LBUTTONDOWN:
            self.drawImg = self.img + 0
            self.drawing = True
            self.ix = x
            self.iy = y

        elif event == cv2.EVENT_LBUTTONUP:
            self.drawing = False
            cv2.rectangle(self.drawImg,(self.ix,self.iy),(x,y)
   ,(0,0,255), 2)
            self.coord = [self.ix, self.iy, x, y]

    @staticmethod
    def toGreyscale(img):
        """
        Converting RGB image to greyscale
        """
        # Get color arrays
        red   = img[...,2]
        green = img[...,1]
        blue  = img[...,0]

        # Fill array with shades of grey
```

```
        outImg = np.zeros((img.shape[0], img.shape[1]))
        outImg[...] = 0.299 * red + 0.587 * green + 0.114 * blue

        # Round result shades
        outImg = np.round(outImg)

        # Update image
        return outImg

    def getCoordinates(self):
        return self.coord


class Tracker:
    def __init__(self, path, time, frameCount, coord):
        self.imgList = list()
        self.imgCoordList = list()
        self.imgCoordList.append(coord)
        self.fix_coord()

        video = cv2.VideoCapture(path)
        video.set(0, time)
        count = 0
        success = True

        while success and count < frameCount:
            success, image = video.read()
            image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            self.imgList.append(image)
            if count > 0:
                self.imgCoordList.append(None)
            count += 1

    @staticmethod
    def get_ssd(first_block, second_block):
        diff = np.sum(cv2.absdiff(first_block, second_block) ** 2)
        return diff

    @staticmethod
    def get_sad(first_block, second_block):
        diff = np.sum(cv2.absdiff(first_block, second_block))
        return diff

    @staticmethod
    def search_window(image, x_coord, y_coord, x_size, y_size):
        """
        Окно поиска блока
        :param image: Изображение, в котором осуществляется поиск
        :param x_coord: Центр окна по оси X
        :param y_coord: Центр окна по оси Y
        :param size: Размер окна
        :return: Возвращает координаты окна (x, y) и само окно (пиксели
    )
        """
```

```
104         if x_coord - x_size < 0:
                x_begin = 0
106             x_end = x_coord + x_size
            elif x_coord + x_size > image.shape[1]:
108             x_begin = x_coord - x_size
                x_end = image.shape[1]
110         else:
                x_begin = x_coord - x_size
112             x_end = x_coord + x_size

114         if y_coord - y_size < 0:
                y_begin = 0
116             y_end = y_coord + y_size
            elif y_coord + y_size > image.shape[0]:
118             y_begin = y_coord - y_size
                y_end = image.shape[0]
120         else:
                y_begin = y_coord - y_size
122             y_end = y_coord + y_size

124         return x_begin, y_begin, image[y_begin:y_end, x_begin:x_end]

126     def fix_coord(self):
            if self.imgCoordList[0][2] < self.imgCoordList[0][0]:
128             self.imgCoordList[0][0], self.imgCoordList[0][2] = self.
    imgCoordList[0][2], self.imgCoordList[0][0]
            if self.imgCoordList[0][3] < self.imgCoordList[0][1]:
130             self.imgCoordList[0][1], self.imgCoordList[0][3] = self.
    imgCoordList[0][3], self.imgCoordList[0][1]

132     @staticmethod
        def getShadeMap(img):
134         """
            Creating map with the amount of each shade
136
            :return: Shade map
138         :rtype:  numpy
            """
140         return np.bincount(img.astype(int).flat, minlength=256)

142     def histogramIntersection(self, oldImage, newImage):
            oldShade = self.getShadeMap(oldImage)
144         newShade = self.getShadeMap(newImage)

146         maxShade = np.sum(oldShade[0:])
            minimum = np.minimum(oldShade, newShade)
148         sumShade = np.sum(minimum[0:])

150         intersection = sumShade / maxShade
            return True if intersection >= 0.5 else False
152
        @staticmethod
154     def block_center(coord):
            x_coord = round((coord[0] + coord[2]) / 2)
```

```
156         y_coord = round((coord[1] + coord[3]) / 2)
            return x_coord, y_coord
158
     @staticmethod
160     def block_size(coord):
            x_size = coord[2] - coord[0]
162         y_size = coord[3] - coord[1]
            return [x_size, y_size]
164


166     @staticmethod
     def get_image_block(coord, image):
168         x_size = coord[2] - coord[0]
            y_size = coord[3] - coord[1]
170         return image[coord[1]:coord[1] + y_size, coord[0]:coord[0] +
    x_size]

172     def find_object(self):
            for index, _ in enumerate(self.imgList):
174             if index == len(self.imgList) - 1:
                    break
176
                sad = 999999
178             to_x = 0
                to_y = 0
180             im_w = None
                flag = False
182
                print(index)
184
                blockSize = self.block_size(self.imgCoordList[index])
186             im_block = self.get_image_block(self.imgCoordList[index],
    self.imgList[index])
                (x_center, y_center) = self.block_center(self.imgCoordList[
    index])
188             x_coord = self.imgCoordList[index][0]
                y_coord = self.imgCoordList[index][1]
190
                (x_start, y_start, im_window) = self.search_window(self.
    imgList[index + 1], x_center, y_center, blockSize[0], blockSize[1])
192             new_SAD = self.get_sad(im_block, self.imgList[index + 1][
    y_coord:y_coord + blockSize[1], x_coord:x_coord + blockSize[0]])
                if new_SAD < 5000:
194                 if self.histogramIntersection(im_block, self.imgList[
    index + 1][y_coord:y_coord + blockSize[1], x_coord:x_coord +
    blockSize[0]]):
                        self.imgCoordList[index + 1] = self.imgCoordList[
    index]
196                 else:
                        break
198                 continue

200             best_x = -1
                best_y = -1
```

```
202
            for x_window in range(0, im_window.shape[1] - blockSize[0]
    + 1):
204                 for y_window in range(0, im_window.shape[0] - blockSize
    [1] + 1):
                        new_SAD = self.get_sad(im_block,
206                                               im_window[y_window:y_window +
    blockSize[1], x_window:x_window + blockSize[0]])
                        if sad > new_SAD:
208                         sad = new_SAD
                            best_x = x_start + x_window
210                         best_y = y_start + y_window

212             if best_x >= 0 and best_y >= 0:
                    coord = [best_x,
214                          best_y,
                             best_x + blockSize[0],
216                          best_y + blockSize[1]]

218                 check = True
                    for item in coord:
220                     if item < 0:
                            check = False
222                 if check and self.histogramIntersection(im_block, self.
    imgList[index + 1][best_y:best_y + blockSize[1], best_x:best_x +
    blockSize[0]]):
                        self.imgCoordList[index + 1] = coord
224                 else:
                        break
226             else:
                    break
228
     def draw_frames(self):
230         for i, v in enumerate(self.imgList):
                if self.imgCoordList[i] is not None:
232                 image = v
                    cv2.rectangle(image,(self.imgCoordList[i][0],self.
    imgCoordList[i][1]),(self.imgCoordList[i][2],self.imgCoordList[i
    ][3]),(0,0,255), 2)
234                 cv2.imshow('{} Frame'.format(i + 1), image)
                else:
236                 cv2.imshow('{} Frame'.format(i + 1), v)
                    break
238         cv2.waitKey(0)
```

Листинг А.1. tracker.py

```
1 # PyQt5 Video player
2 #!/usr/bin/env python

4 from PyQt5.QtCore import QDir, Qt, QUrl, pyqtSignal
  from PyQt5.QtMultimedia import QMediaContent, QMediaPlayer
6 from PyQt5.QtMultimediaWidgets import QVideoWidget
```

```
from PyQt5.QtWidgets import (QApplication, QFileDialog, QHBoxLayout,
    QLabel,
        QPushButton, QSizePolicy, QSlider, QStyle, QVBoxLayout, QWidget
    )
from PyQt5.QtWidgets import QMainWindow,QWidget, QPushButton, QAction,
    QLineEdit, QDesktopWidget
from PyQt5.QtGui import QIcon
import sys
from tracker import ObjectRectangle, Tracker

class MainWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        self.setWindowTitle('Трекер движения объекта')

        layout = QVBoxLayout()
        self.firstFrameBtn = QPushButton("1ый кадр")
        self.firstFrameLine = QLineEdit(self)
        self.secondFrameLine = QLineEdit(self)

        self.xObjectLine = QLineEdit(self)
        self.yObjectLine = QLineEdit(self)
        self.xOffsetObjectLine = QLineEdit(self)
        self.yOffsetObjectLine = QLineEdit(self)

        self.frameCount = QLineEdit(self)

        self.frameCountLabel = QLabel("Кол-во кадров")

        self.selectBtn = QPushButton("Выбрать объект")
        self.selectBtn.setEnabled(False)
        self.startBtn = QPushButton("Старт!")

        self.firstFrameBtn.clicked.connect(self.selectFile)
        self.selectBtn.clicked.connect(self.selectObject)
        self.startBtn.clicked.connect(self.start)

        self.firstFrameLine.textChanged.connect(self.enableSelectBtn)
        self.secondFrameLine.textChanged.connect(self.enableSelectBtn)

        h1l = QHBoxLayout()
        h1l.addWidget(self.firstFrameLine, 3)
        h1l.addWidget(self.secondFrameLine, 1)
        h1l.addWidget(self.firstFrameBtn, 1)

        h2l = QHBoxLayout()
        h2l.addWidget(self.xObjectLine)
        h2l.addWidget(self.yObjectLine)
        h2l.addWidget(self.xOffsetObjectLine)
        h2l.addWidget(self.yOffsetObjectLine)
        h2l.addWidget(self.selectBtn)
```

```
58
        v1l = QVBoxLayout()
60      v1l.addWidget(self.frameCountLabel)
        v1l.addWidget(self.frameCount)
62
        h3l = QHBoxLayout()
64      h3l.addLayout(v1l)

66      layout.addLayout(h1l)
        layout.addLayout(h2l)
68      layout.addLayout(h3l)
        layout.addWidget(self.startBtn)
70      self.setLayout(layout)

72      self.player = VideoWindow(self)

74      self.setGeometry(0, 0, 500, 100)
        window = self.frameGeometry()
76      centerPoint = QDesktopWidget().availableGeometry().center()
        window.moveCenter(centerPoint)
78      self.move(window.topLeft())

80      self.show()

82   def selectFile(self):
        self.player.show()
84
   def enableSelectBtn(self):
86      if self.firstFrameLine.text() != "" and self.secondFrameLine.
    text() != "":
            self.selectBtn.setEnabled(True)
88      else:
            self.selectBtn.setEnabled(False)
90
   def selectObject(self):
92      obj = ObjectRectangle(self.firstFrameLine.text(), int(self.
    secondFrameLine.text()))
        obj.showFrame()
94      coord = obj.getCoordinates()

96      self.xObjectLine.setText(str(coord[0]))
        self.yObjectLine.setText(str(coord[1]))
98      self.xOffsetObjectLine.setText(str(coord[2]))
        self.yOffsetObjectLine.setText(str(coord[3]))
100
   def start(self):
102     self.startBtn.setEnabled(False)
        self.selectBtn.setEnabled(False)
104     self.firstFrameBtn.setEnabled(False)
        self.startBtn.setText("Пожалуйста, подождите пару минут...")
106
        coord = [
108         int(self.xObjectLine.text()),
            int(self.yObjectLine.text()),
```

```
            int(self.xOffsetObjectLine.text()),
            int(self.yOffsetObjectLine.text()),
        ]

        t = Tracker(
            self.firstFrameLine.text(),
            int(self.secondFrameLine.text()),
            int(self.frameCount.text()),
            coord)
        t.find_object()
        t.draw_frames()

        self.startBtn.setText("Старт!")
        self.startBtn.setEnabled(True)
        self.selectBtn.setEnabled(True)
        self.firstFrameBtn.setEnabled(True)

class VideoWindow(QMainWindow):

    def __init__(self, parent=None):
        super(VideoWindow, self).__init__(parent)
        self.setWindowTitle("Выберите первый кадр")
        self.resize(800, 600)

        self.mediaPlayer = QMediaPlayer(None, QMediaPlayer.VideoSurface
    )
        self.fileName = None

        videoWidget = QVideoWidget()

        self.playButton = QPushButton()
        self.playButton.setEnabled(False)
        self.playButton.setIcon(self.style().standardIcon(QStyle.
    SP_MediaPlay))
        self.playButton.clicked.connect(self.play)

        self.okBtn = QPushButton("OK")
        self.okBtn.clicked.connect(self.selectFrame)

        self.positionSlider = QSlider(Qt.Horizontal)
        self.positionSlider.setRange(0, 0)
        self.positionSlider.sliderMoved.connect(self.setPosition)

        self.errorLabel = QLabel()
        self.errorLabel.setSizePolicy(QSizePolicy.Preferred,
                QSizePolicy.Maximum)

        # Create new action
        openAction = QAction(QIcon('open.png'), '&Open', self)
        openAction.setShortcut('Ctrl+O')
        openAction.setStatusTip('Open movie')
        openAction.triggered.connect(self.openFile)

        # Create menu bar and add action
```

```
        menuBar = self.menuBar()
        fileMenu = menuBar.addMenu('&File')
        fileMenu.addAction(openAction)

        # Create a widget for window contents
        wid = QWidget(self)
        self.setCentralWidget(wid)

        # Create layouts to place inside widget
        controlLayout = QHBoxLayout()
        controlLayout.setContentsMargins(0, 0, 0, 0)
        controlLayout.addWidget(self.playButton)
        controlLayout.addWidget(self.positionSlider)

        layout = QVBoxLayout()
        layout.addWidget(videoWidget)
        layout.addLayout(controlLayout)
        layout.addWidget(self.okBtn)
        layout.addWidget(self.errorLabel)

        # Set widget to contain window contents
        wid.setLayout(layout)

        self.mediaPlayer.setVideoOutput(videoWidget)
        self.mediaPlayer.stateChanged.connect(self.mediaStateChanged)
        self.mediaPlayer.positionChanged.connect(self.positionChanged)
        self.mediaPlayer.durationChanged.connect(self.durationChanged)
        self.mediaPlayer.error.connect(self.handleError)

    def openFile(self):
        self.fileName, _ = QFileDialog.getOpenFileName(self, "Open
Movie")

        if self.fileName != '':
            self.mediaPlayer.setMedia(
                    QMediaContent(QUrl.fromLocalFile(self.fileName)))
            self.playButton.setEnabled(True)

    def play(self):
        if self.mediaPlayer.state() == QMediaPlayer.PlayingState:
            self.mediaPlayer.pause()
        else:
            self.mediaPlayer.play()

    def mediaStateChanged(self, state):
        if self.mediaPlayer.state() == QMediaPlayer.PlayingState:
            self.playButton.setIcon(
                    self.style().standardIcon(QStyle.SP_MediaPause))
        else:
            self.playButton.setIcon(
                    self.style().standardIcon(QStyle.SP_MediaPlay))

    def positionChanged(self, position):
        self.positionSlider.setValue(position)
```

```
216     def durationChanged(self, duration):
            self.positionSlider.setRange(0, duration)
218
        def setPosition(self, position):
220         self.mediaPlayer.setPosition(position)

222     def handleError(self):
            self.playButton.setEnabled(False)
224         self.errorLabel.setText("Error: " + self.mediaPlayer.
    errorString())

226     def selectFrame(self):
            if self.fileName is not None:
228             self.parent().firstFrameLine.setText(self.fileName)
                self.parent().secondFrameLine.setText(str(self.
    positionSlider.value()))
230             self.close()


232
if __name__ == '__main__':
234     app = QApplication(sys.argv)
        mw = MainWindow()
236     sys.exit(app.exec_())
```

Листинг А.2. ui.py

```
1 cycler==0.10.0
2 matplotlib==2.0.2
  numpy==1.12.1
4 opencv-python==3.3.0.10
  pyparsing==2.2.0
6 PyQt5==5.9.2
  python-dateutil==2.6.1
8 pytz==2017.2
  sip==4.19.6
10 six==1.10.0
```

Листинг А.3. Файл с использованными зависимостями для virtualenv