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

Отчет по лабораторной работе №4

Дисциплина: Методы и средства цифровой обработки информации

Выполнил
студент гр. 13541/2 _____ Зобков Д. А.
(подпись)

Преподаватель _____ Абрамов Н. А.
(подпись)

“___”_____ 2017 г.

Санкт-Петербург
2017 г.

СОДЕРЖАНИЕ

1 Техническое задание

Необходимо извлечь два последовательных кадра из видео и вычислить для них вектора смещений, используя метрику SAD или SSD. По найденным векторам смещения восстановить предыдущий кадр. Визуализировать результат. Исследовать влияние параметров блока и окна поиска на результат восстановления.

2 Ход работы

2.1 Описание алгоритма

1. Делим текущий кадр на равные блоки (NxN);
2. Производим обход всех блоков текущего изображения, где для каждого блока производится обход некоторой окрестности блока в предыдущем кадре в поиске максимального соответствия (с помощью метрик SAD, SSD);
3. Таким образом, после завершения поиска мы получаем набор векторов, указывающих „движение“ блоков между кадрами. С их помощью восстанавливаем предыдущий кадр.

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

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

2.2 Взятие последовательных кадров

Напишем скрипт (листинг А.1 на с. 11) для взятия последовательных кадров из видео. В результате, в каталоге „frames“ появилось определенное количество последовательных кадров, сохраненных в формате png. Для исследования выберем следующие:

Рис. 2.1. Кадр T-1

Рис. 2.2. Кадр T

2.3 Вычисление векторов смещения и восстановления кадра

Реализация алгоритма представлена в прил. А на с. 11. Для исследования влияния параметров блока и окна поиска проведем следующие эксперименты (для всех случаев шаг поиска равен размеру блока):

Оценка влияния окна поиска:
Размер блока 16 на 16. Окно поиска 32 (64 на 64)

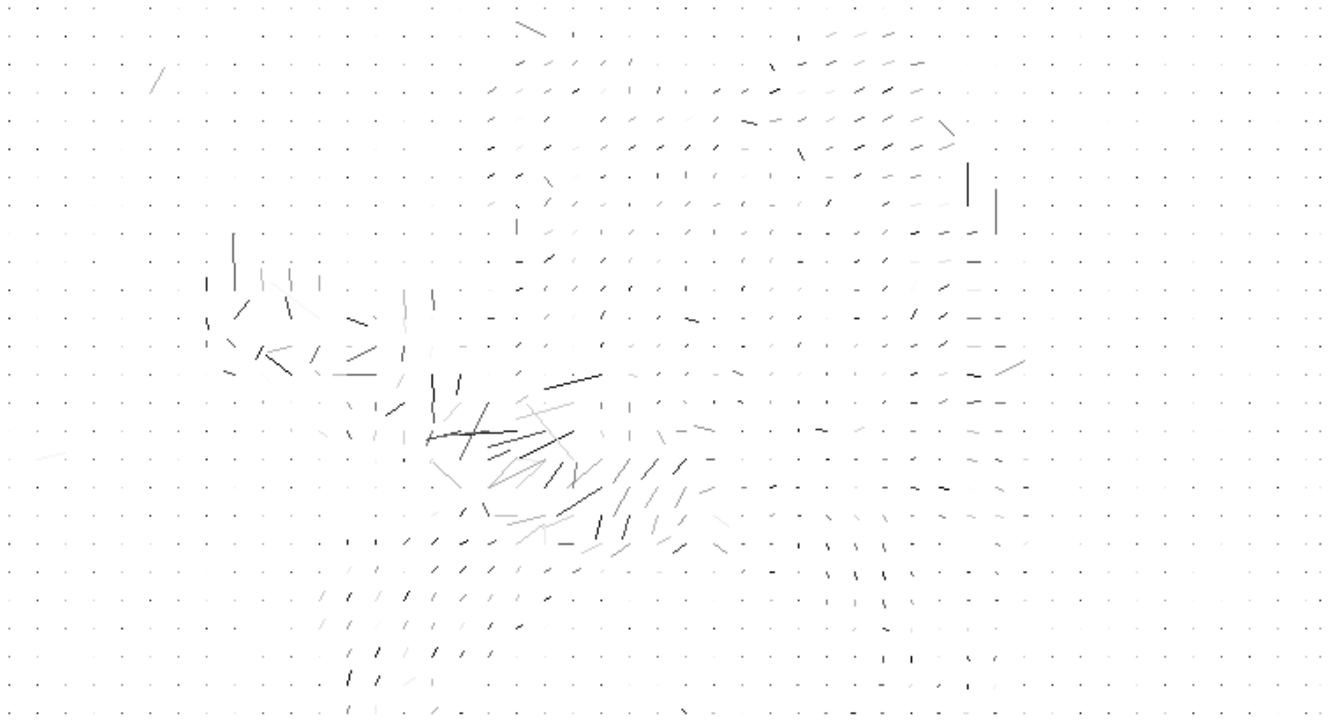

Рис. 2.3. Векторы смещения (16x16, 64x64)

Рис. 2.4. Восстановленное изображение (16x16, 64x64)

Размер блока 16 на 16. Окно поиска 40 (80 на 80)

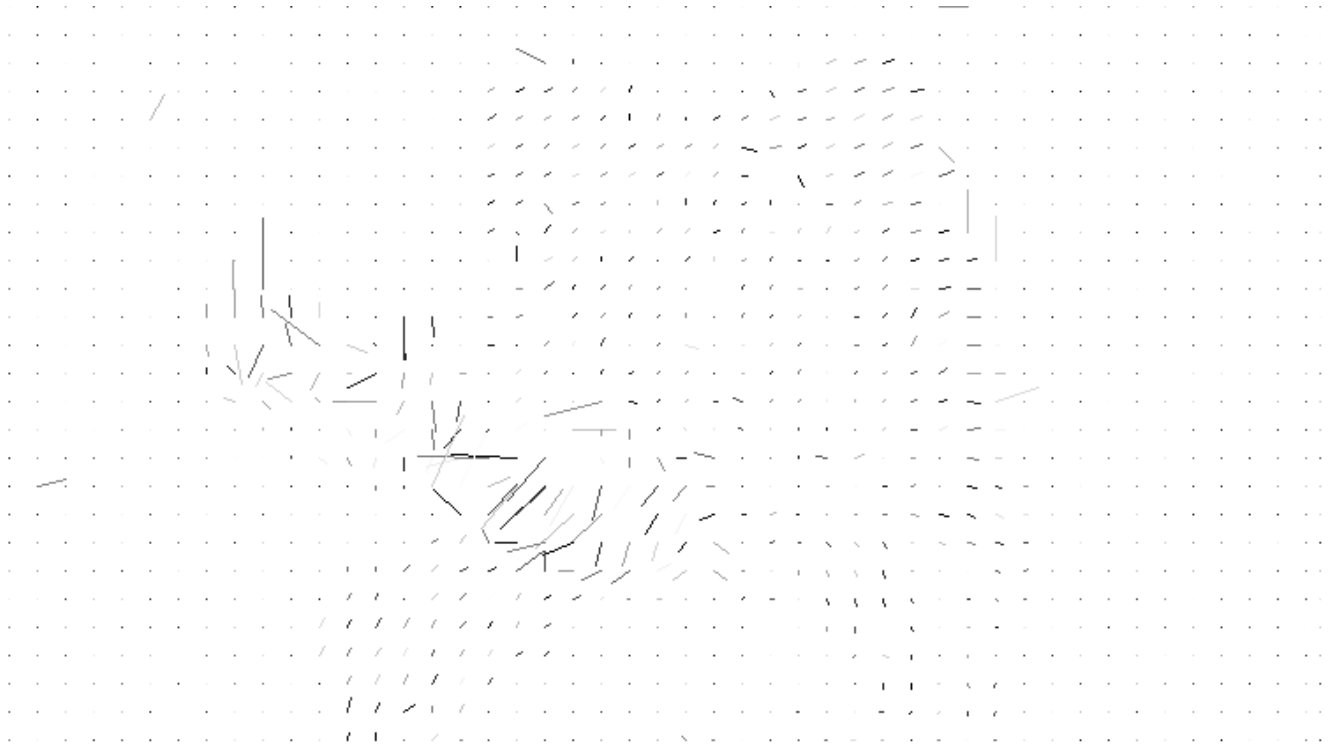

Рис. 2.5. Векторы смещения (16x16, 80x80)

Рис. 2.6. Восстановленное изображение (16x16, 80x80)

Оценка влияния размера блока:
Размер блока 8 на 8. Окно поиска 40 (80 на 80)

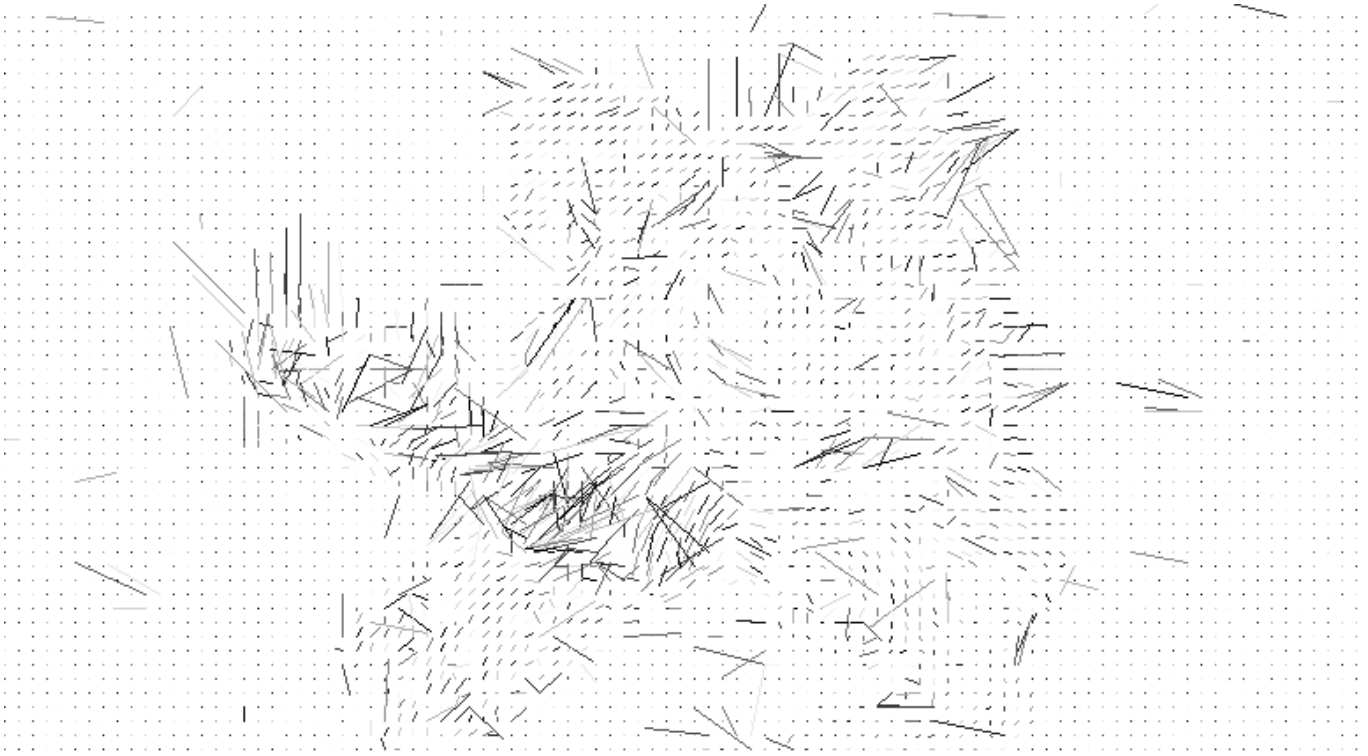

Рис. 2.7. Векторы смещения (8x8, 80x80)

Рис. 2.8. Восстановленное изображение (8x8, 80x80)

Размер блока 5 на 5. Окно поиска 40 (80 на 80)

Рис. 2.9. Векторы смещения (5x5, 80x80)

Рис. 2.10. Восстановленное изображение (5x5, 80x80)

Также попробуем провести поиск векторов смещения для кадра, сдвинутого на несколько пикселей вправо:

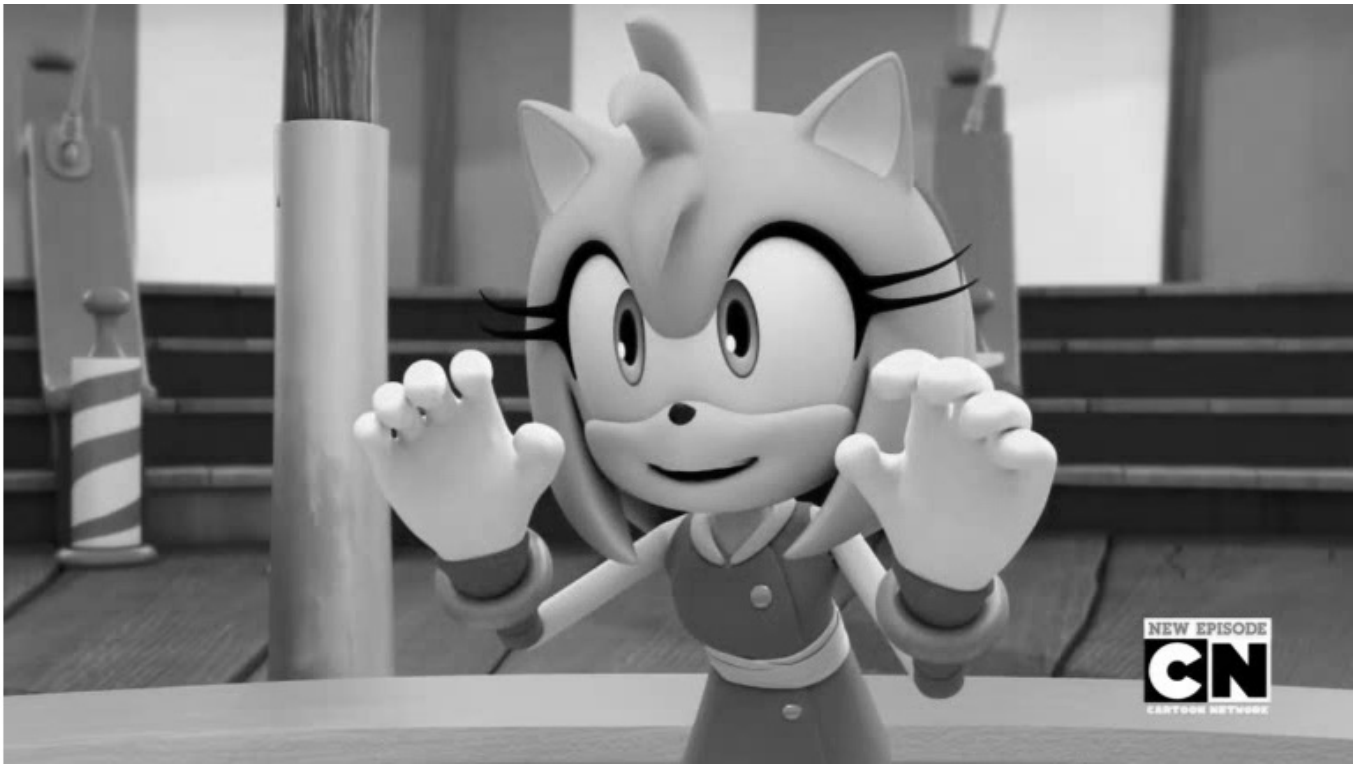

Рис. 2.11. Сдвинутое изображение

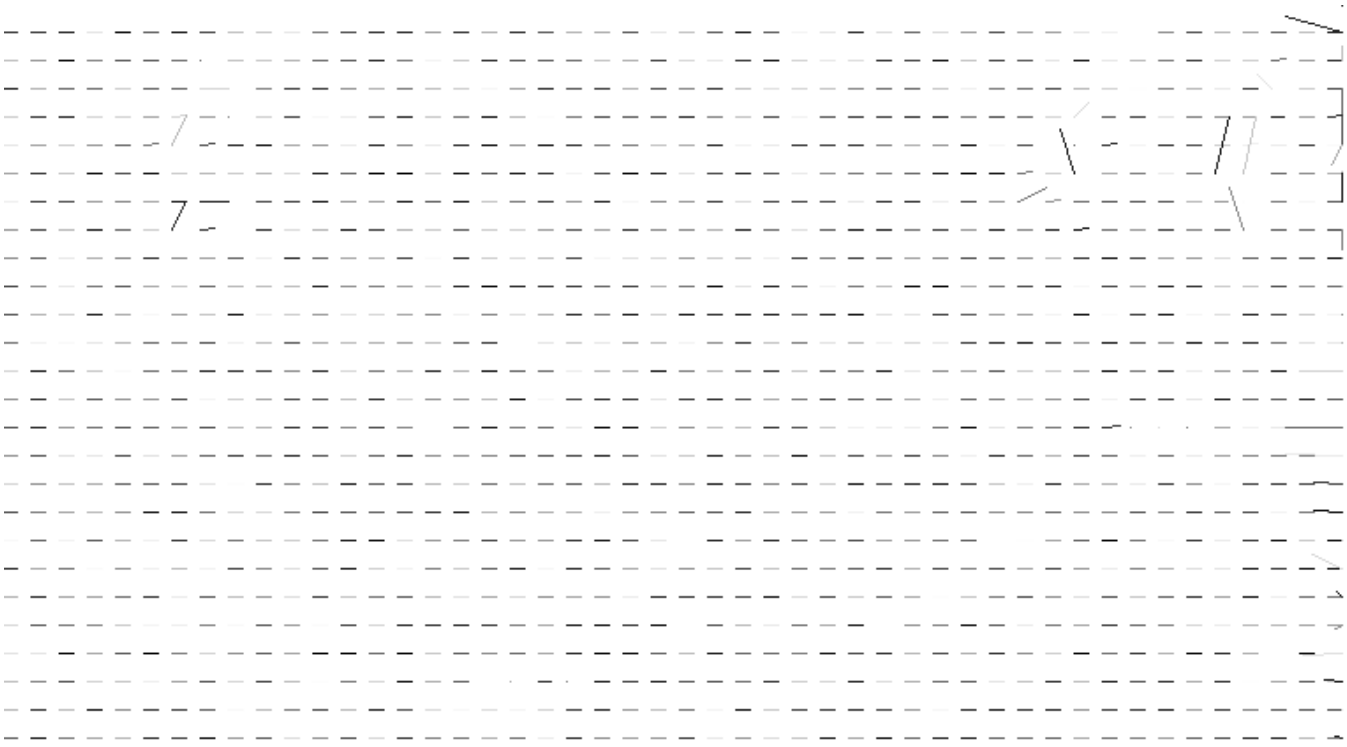

Рис. 2.12. Векторы смещения (16x16, 64x64)

Как можно заметить, сдвиг большинства блоков был обнаружен корректно, некорректные векторы скорее всего вызваны идентичностью этих блоков, из-за чего произошло ложное срабатывание.

3 Выводы

Существуют разные методы компенсации движения в видео. Все они могут быть адаптированы под решение различного рода задач, например, для сжатия данных. Одним из наиболее популярных является блочный метод. Он позволяет по найденным векторам смещения восстанавливать изображение.

В ходе выполнения данной работы был проведен ряд экспериментов, из которых установлено, что такие параметры как „размер блока“ и „размер окна поиска“ оказывают наибольшее влияние на качество восстановления. Для тестовых фреймов было установлено, что при размере блока 5x5 и окна поиска 40x40 восстановление проходит почти без потери качества. Следует отметить, что параметр „шаг поиска“ также влияет на качество изображения, однако для того чтобы его использование было целесообразно, необходимо реализовать качественный алгоритм для „сложения“ пикселей (в данной работе при реализации используется среднее арифметическое).

Как можно заметить, чем меньше размер блока, тем меньше фрагментация восстановленного изображения. Увеличение окна поиска вместе с уменьшением шага поиска обеспечивают большее количество потенциальных кандидатов на искомый блок, что увеличивает вероятность и точность его нахождения. Взамен все эти улучшения качества поиска увеличивают время его выполнения.

Приложение А Исходный код программы

```
import cv2

vid_cap = cv2.VideoCapture('test.avi')
success, image = vid_cap.read()
count = 0
success = True
while success and count < 1000:
    success, image = vid_cap.read()
    print('Read a new frame: ', success)
    cv2.imwrite("frames/frame%d.png" % count, image)
    count += 1
```

Листинг А.1. get_frames.py

```
import random
import numpy as np
import cv2

class ShiftVector:
    def __init__(self, pathList, blockSize, step, windowSize):
        self.BLOCK_SIZE = [int(blockSize), int(blockSize)]
        self.STEP_SIZE = int(step)
        self.WINDOW_SIZE = int(windowSize)

        self.t_1 = cv2.imread(pathList[0], 0)
        self.t = cv2.imread(pathList[1], 0)

        cv2.imshow("1 Frame", self.t_1)
        cv2.imshow("2 Frame", self.t)
        cv2.waitKey(0)

        self.t_rec = self.t_1.copy()
        self.t_vec = self.t_1.copy()
        self.t_rec2 = self.t_1.copy()
        self.t_vec.fill(255)
        self.t_rec.fill(0)
        self.t_rec2.fill(0)

    @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 sliding_block(image, block_size, step):
        """
        Функция разбиения исходного изображения на блоки
```

```
        :param image: Исходное изображение
        :param block_size: Размер блока [NxL]
        :param step: Шаг блока
        :return: Нарезанные блоки (yield)
        """
        for x_coord in range(0, image.shape[0], step):
            if x_coord + block_size[0] > image.shape[0]:
                x_coord = image.shape[0] - block_size[0]
            for y_coord in range(0, image.shape[1], step):
                if y_coord + block_size[1] > image.shape[1]:
                    y_coord = image.shape[1] - block_size[1]
                yield (x_coord, y_coord, image[x_coord:x_coord +
block_size[0], y_coord:y_coord + block_size[1]])

    @staticmethod
    def replace_block(a, b, block_size, x_coord, y_coord):
        """
        Вставка найденного блока в изображение для восстонавления
        :param a: Восстанавливаемое изображение
        :param b: Блок для замены
        :param c: Акк
        :param block_size: Размер блока [NxL]
        :param x_coord: Начало заменяемого блока на оси x
        :param y_coord: Начало заменяемого блока на оси y
        """
        a[x_coord:x_coord + block_size[0], y_coord:y_coord + block_size
[1]] = b[0:block_size[0], 0:block_size[1]]

    @staticmethod
    def search_window(image, x_coord, y_coord, 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
)
        """
        if x_coord - size < 0:
            x_begin = 0
            x_end = x_coord + size
        elif x_coord + size > image.shape[0]:
            x_begin = x_coord - size
            x_end = image.shape[0]
        else:
            x_begin = x_coord - size
            x_end = x_coord + size

        if y_coord - size < 0:
            y_begin = 0
            y_end = y_coord + size
        elif y_coord + size > image.shape[1]:
            y_begin = y_coord - size
```

```
            y_end = image.shape[1]
        else:
            y_begin = y_coord - size
            y_end = y_coord + size

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

    def find_vectors(self):
        for (x_block, y_block, im_block) in self.sliding_block(self.t_1
, self.BLOCK_SIZE, self.STEP_SIZE):
            sad = 99999
            to_x = 0
            to_y = 0
            im_w = None
            flag = False
            (x_start, y_start, im_window) = self.search_window(self.t,
x_block, y_block, self.WINDOW_SIZE)
            new_SAD = self.get_sad(im_block, self.t[x_block:x_block +
self.BLOCK_SIZE[0], y_block:y_block + self.BLOCK_SIZE[1]])
            if new_SAD < 10:
                sad = new_SAD
                to_x = x_block
                to_y = y_block
                im_w = self.t[x_block: x_block + self.BLOCK_SIZE[0],
y_block: y_block + self.BLOCK_SIZE[1]]
                self.replace_block(self.t_rec, im_w, self.BLOCK_SIZE,
x_block, y_block)
                cv2.line(self.t_vec, (to_y, to_x), (y_block, x_block),
(random.randint(0, 256)), 1, 8, 0)
                continue

            for x_window in range(0, im_window.shape[0] - self.
BLOCK_SIZE[0] + 1):
                if flag is True:
                    flag = False
                    break
                for y_window in range(0, im_window.shape[1] - self.
BLOCK_SIZE[1] + 1):
                    new_SAD = self.get_sad(im_block,
                                           im_window[x_window:x_window +
self.BLOCK_SIZE[0], y_window:y_window + self.BLOCK_SIZE[1]])
                    if sad > new_SAD:
                        sad = new_SAD
                        to_x = x_start + x_window
                        to_y = y_start + y_window
                        im_w = im_window[x_window: x_window + self.
BLOCK_SIZE[0], y_window: y_window + self.BLOCK_SIZE[1]]
                        if sad < 10:
                            self.replace_block(self.t_rec, im_w, self.
BLOCK_SIZE, x_block, y_block)
                            cv2.line(self.t_vec, (to_y, to_x), (y_block
, x_block), (random.randint(0, 256)), 1, 8, 0)
                            flag = True
                            break
```

```
132
                self.replace_block(self.t_rec, im_w, self.BLOCK_SIZE,
    x_block, y_block)
134             cv2.line(self.t_vec, (to_y, to_x), (y_block, x_block), (
    random.randint(0, 256)), 1, 8, 0)

136         cv2.imshow("Restored 1 frame", self.t_rec)
            cv2.imshow("Shift vectors", self.t_vec)
```

Листинг А.2. shiftvector.py

```
1  import sys
2  from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout,
       QHBoxLayout, QPushButton, QLabel, QDesktopWidget, QLineEdit,
       QFileDialog
   from shiftvector import ShiftVector
4
   class MainWindow(QWidget):
6      def __init__(self):
           super().__init__()
8          self.initUI()

10     def initUI(self):
           self.setWindowTitle('Поиск векторов смещения')
12
           layout = QVBoxLayout()
14         self.firstFrameBtn = QPushButton("1 кадр")
           self.secondFrameBtn = QPushButton("2 кадр")
16         self.firstFrameLine = QLineEdit(self)
           self.secondFrameLine = QLineEdit(self)
18
           self.blockSize = QLineEdit(self)
20         self.stepSize = QLineEdit(self)
           self.windowSize = QLineEdit(self)
22
           self.blockSizeLabel = QLabel("Размер блока")
24         self.stepSizeLabel = QLabel("Шаг поиска")
           self.windowSizeLabel = QLabel("Окно поиска")
26
           self.startBtn = QPushButton("Старт!")
28
           self.firstFrameBtn.clicked.connect(self.selectFile)
30         self.secondFrameBtn.clicked.connect(self.selectFile)
           self.startBtn.clicked.connect(self.start)
32
           h1l = QHBoxLayout()
34         h1l.addWidget(self.firstFrameLine)
           h1l.addWidget(self.firstFrameBtn)
36
           h2l = QHBoxLayout()
38         h2l.addWidget(self.secondFrameLine)
           h2l.addWidget(self.secondFrameBtn)
40
           v1l = QVBoxLayout()
```

```
42          v1l.addWidget(self.blockSizeLabel)
            v1l.addWidget(self.blockSize)
44
            v2l = QVBoxLayout()
46          v2l.addWidget(self.stepSizeLabel)
            v2l.addWidget(self.stepSize)
48
            v3l = QVBoxLayout()
50          v3l.addWidget(self.windowSizeLabel)
            v3l.addWidget(self.windowSize)
52
            h3l = QHBoxLayout()
54          h3l.addLayout(v1l)
            h3l.addLayout(v2l)
56          h3l.addLayout(v3l)

58          layout.addLayout(h1l)
            layout.addLayout(h2l)
60          layout.addLayout(h3l)
            layout.addWidget(self.startBtn)
62          self.setLayout(layout)

64          self.setGeometry(0, 0, 400, 100)
            window = self.frameGeometry()
66          centerPoint = QDesktopWidget().availableGeometry().center()
            window.moveCenter(centerPoint)
68          self.move(window.topLeft())

70          self.show()

72      def selectFile(self):
            sender = self.sender()
74          name = QFileDialog.getOpenFileName(self, 'Выберите {}'.format(
    sender.text()))[0]
            if sender.text() == '1 кадр':
76              self.firstFrameLine.setText(name)
            else:
78              self.secondFrameLine.setText(name)

80      def start(self):
            self.startBtn.setEnabled(False)
82          self.startBtn.setText("Пожалуйста, подождите пару минут...")

84          sv = ShiftVector(
                [self.firstFrameLine.text(), self.secondFrameLine.text()],
86              self.blockSize.text(),
                self.stepSize.text(),
88              self.windowSize.text())
            sv.find_vectors()
90
            self.startBtn.setText("Старт!")
92          self.startBtn.setEnabled(True)

94  if __name__ == '__main__':
```

```
96      app = QApplication(sys.argv)
        mw = MainWindow()
98      sys.exit(app.exec_())
```

Листинг А.3. 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.4. Файл с использованными зависимостями для virtualenv