МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гродненский государственный университет им. Янки Купалы»

Физико-технический факультет

|  |  |
| --- | --- |
| Кафедра информационных  систем и технологий | ДОПУЩЕН К ЗАЩИТЕ  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.  Зав. Кафедрой информационных  систем и технологий  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ю.Р.Бейтюк |

Специальность: I-38.02.01 Информационно-измерительная техника

КУРСОВАЯ РАБОТА

По дисциплине: Измерительные приборы и системы

Тема: Разработка и визуализация жестовых команд для систем управления

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

|  |  |  |
| --- | --- | --- |
| Разработал: |  | Ступакевич М.Р. |
|  | (подпись, дата) |  |
| Руководитель: |  | Ассанович Б.А. |
|  | (подпись, дата) |  |

2021

Содержание

[Введение 3](#_Toc90147538)

[1. Теоретическая часть 5](#_Toc90147539)

[1.1 Определение набора жестовых команд 5](#_Toc90147540)

[1.2 Описание используемых данных 7](#_Toc90147541)

[2. Реализации системы классификатора 8](#_Toc90147542)

[2.1 Структурная схема системы 8](#_Toc90147543)

[2.2 Используемое программное обеспечение 10](#_Toc90147544)

[2.3 Блок-схема и описание работы алгоритма 11](#_Toc90147545)

[ЗАКЛЮЧЕНИЕ 17](#_Toc90147546)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 18](#_Toc90147547)

[Приложение 19](#_Toc90147548)

# Введение

Обычно, когда мы хотим что-то сделать, то всегда сопровождаем это какими-либо физическими действиями для достижения ожидаемого результата. Например, идём в магазин, чтобы купить продукты, протягиваем руку, чтобы взять нужный предмет или присаживаемся, чтобы завязать шнурки, поднимаем руку, чтобы обратить на себя внимание, поворачиваем голову, чтобы лучше что-то услышать. И в этом смысле жесты или движения куда более интуитивны чем статическое нажатие на кнопку вкл/выкл.

Поэтому переход к жестам кажется таким же логичным как, в своё время, переход от кнопок к сенсору.

Как же определить руку с жестом по видео? Оказывается, есть алгоритмы, которые умеют распознавать в видеопотоке человеческую ладонь. Работает это так же, как и лицом.

У Google есть своя открытая библиотека Google MediaPipe Hands, с помощью которой были получены данные для этой курсовой работы.

Google MediaPipe Hands - это высококачественное решение для отслеживания рук и пальцев. Он использует машинное обучение (ML) для определения 21 3D-ориентира руки всего по одному кадру.

Целью данной курсовой работы является разработка и визуализация жестовых команд и обработка данных для дальнейшего использования в обучении нейронной сети, определяющей жестовые команды на видеопотоке.

Для разработки и визуализации жестовых команд необходимо решить следующие задачи:

1. Определить набор жестовых команд, данные которых будут обрабатываться и визуализироваться.
2. Получить набор данных (датасет) отображающих соответствие параметров измеряемых физических пар-ров выбранным жестовым командам.
3. Выполнить определение связей между ключевыми точками кисти руки.
4. Получить визуализацию каждого кадра жеста руки.
5. Проанализировать полученную визуализацию и произвести обработку.

# 1. Теоретическая часть

## **1.1 Определение набора жестовых команд**

В настоящее время есть существует несколько устройств, использующих управление жестами. Например, телевизоры Philips, управление которыми осуществляется следующим образом: следует поднять руку и направить ладонь в сторону телевизора, после чего активируется управление жестами, далее сжатием и расжатием кулака открывается меню, а навигация по меню реализована использованием руки в качестве указки с сжатием кулака как сигнал нажать на кнопку. Или новое устройство от СБЕР «SberBox Top», в котором, на данный момент, реализовано использование пяти жестов (Рисунок - 1.1): «Салют» для старта/ответа на звонок, «Стоп» для паузы, «Лайк» и «Дизлайк» для оценки видеоконтента и «Тихо» для включения/выключения микрофона в звонке. Кстати, алгоритмы SberBox Top, основаны и обучены на данных, также полученных с помощью Google MediaPipe Hands.



Рисунок – 1.1 Жесты SberBox

Для выполнения задачи формирования набора жестовых команд управления, необходимо определить интуитивно понятные человеку движения и достаточно простые для распознавания алгоритмом, во избежание нечётких срабатываний.

Так же жесты должно в полной мере удовлетворять базовые потребности в управлении. За пример был взят обычный пульт с стандартными кнопками: Вверх, Вниз, Влево, Вправо, Назад, ОК. Таким образом были определены следующие движения рукой с ладонью направленной в сторону камеры (Рисунок - 1.2):

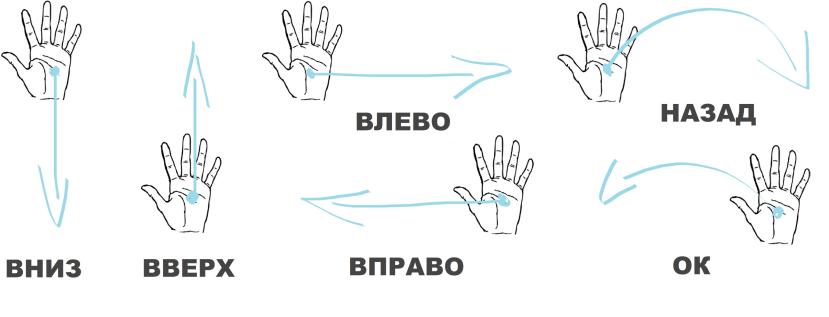


Рисунок – 1.2 Жесты управления

Во избежание непреднамеренных активаций команд было принято использовать так называемый «синхрожест», который представляет собой наполовину сжатый кулак (Рисунок - 1.3) и сопровождает управляющий жест от начала до конца.



Рисунок – 1.3 Пример синхрожеста

## **1.2 Описание используемых данных**

Используемые данные представляют собой набор из 6 жестов, по 30 записей, по 40 кадров, в сумме 7200 .npy файлов (Библиотеки Numpy для Python), который состоит из 126 чисел с плавающей точкой, описывающих 3D**-**координаты двух рук по 21 ключевой точке.

Обрабатывать и визуализировать было принято только одну правую руку с целью экономии вычислительных ресурсов, т.к. наличие двух рук увеличивает количество фич (признаков) вдвое и обучение нейронной сети на данных большей размерности увеличивается на порядок, что уж говорить о большем количестве рук. Для демонстрации и решения поставленной задачи достаточно лишь одной руки (При наличии более высоких мощностей возможно легко адаптировать представленный алгоритм для обработки и визуализации двух и более рук).

Коллекция обнаруженных/отслеженных рук, где каждая рука представлена в виде списка из 21 ориентира руки, и каждый ориентир состоит из **x, y, z**. Где **x** и **y** нормализуются от 0.0 до 1.0 по ширине и высоте изображения соответственно. А **z** представляет глубину ориентира, причем глубина на запястье является началом координат, и чем меньше значение, тем ближе ориентир к камере. Величина **z** вычисляется примерно того же масштаба, **x** что и **y**. Ключевые точки в файлах расположены следующим образом (Рисунок - 1.4):

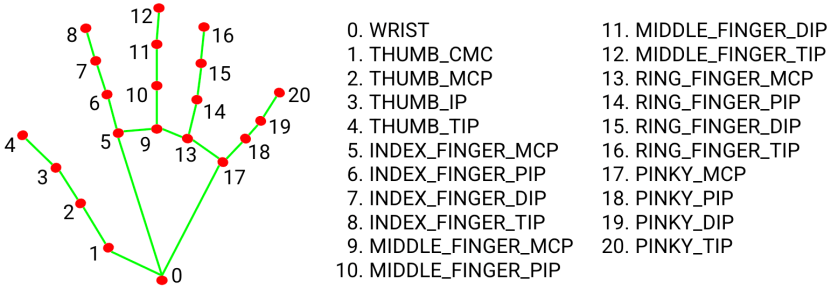


Рисунок – 1.4 Нумерация ключевых точек

# 2. Реализации системы классификатора

## **2.1 Структурная схема системы**

Алгоритм, выполняющий поставленную цель должен на входе получить данные и, произведя некоторую обработку, вывести их визуализацию, которая представляет собой набор ключевых точек кисти руки с проиллюстрированными связами между ними в евклидовом пространстве, на выходе. Для этого необходимо следовать следующим пунктам:

1. Определить набор жестовых команд, данные которых будут обрабатываться и визуализироваться;
2. Получить набор данных (датасет) отображающих соответствие параметров измеряемых физических пар-ров выбранным жестовым командам из файлов и поместить их в единый объект (переменную).
3. Выполнить определение связей между ключевыми точками кисти руки.
4. Извлечь координаты каждой ключевой точки в указанном файле, указанного жеста.
5. Получить визуализацию ключевых точек кадра жеста руки и визуализировать связи между ними;
6. Проанализировать полученную визуализацию и произвести обработку.
7. Произвести обработку и повторить пункт 5-6.

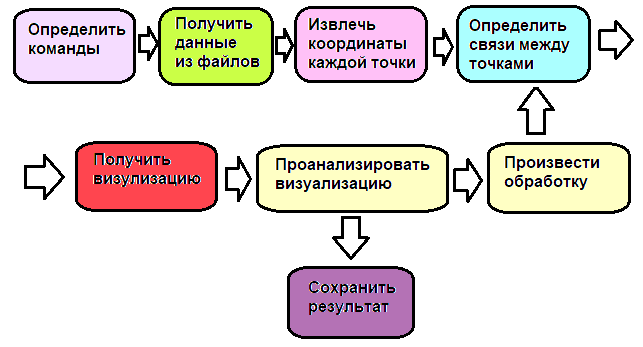


Рисунок - 2.1 Структурная схема разработки алгоритма

## **2.2 Используемое программное обеспечение**

Решение разрабатывается на языке Python и дополнительных библиотек для работы с данными и их визуализацией с использованием интерпретатора IPython в локальной веб-среде Jypyter, т.к. IPython позволяет выполнять программный код последовательно по частям и визуально отслеживать результаты на всех этапах выполнения программы.

Используемые Python библиотеки:

OS ­­­­­­­— стандартная библиотека Python, позволяющая работать с операционной системой. В программе использовался метод path.join, создающий относительный путь к файлу независимо от типа операционной системы. Это необходимо для указания файлов и последующего извлечения необходимых данных.

NumPy — это библиотека языка Python, добавляющая поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых (и очень быстрых) математических функций для операций с этими массивами. Использовались такие методы как: array для создания массива, load для загрузки данных из файлов, shape для определения размерности массива данных и interp для линейной интерполяции данных.

Matplotlib — это мощная бибилиотека двумерной графики для Python, с помощью которой можно создавать высококачественные рисунки различных форматов. В программе использование данный библиотеки ограничивалось методами plot и show для создания и отображения трёхмерных графиков, представляющих собой эвклидово пространство с расположенными в нём ключевыми точками кисти руки. Также, для установки «точки зрения» и масштабирования видимой области, использовались методы view\_init и xlim/ylim. Данные операции были необходимы для визуального анализа полученных и преобразованных данных.

## **2.3 Блок-схема и описание работы алгоритма**



Рисунок - 2.2 Блок-схема получения датасета

Разработку алгоритма необходимо начать с получения данных. Для этого необходимо указать относительный путь к папке где расположены все файлы.

Так как алгоритм заранее не знает какое количество жестов, сколько записей и их размер, необходимо явно это указать.Так же необходимо указать связи между ключевыми точками кисти руки.

Далее, с учётом известных параметров, данные извлекаются из файлов и помещаются в объект (переменную).

Данные операции выполняет следующий код:

import os

DATA\_PATH = os.path.join('MP\_Data\\Hands\_Data')

actions = np.array([ 'up', 'down', 'left', 'right', 'ok', 'back'])

no\_sequences = 30

sequence\_length = 40

label\_map = {label:num for num, label in enumerate(actions)}

parts = [

[0, 1, 0, 5, 0, 17, 'green'],

[5,9,9,13,13,17,'red'],

[1,2,2,3,3,4,'blue'],

[5,6,6,7,7,8,'yellow'],

[9,10,10,11,11,12,'orange'],

[13,14,14,15,15,16,'pink'],

[17,18,18,19,19,20,'purple']

]

sequences, labels = [], []

for action in actions:

for sequence in range(no\_sequences):

window = []

for frame\_num in range(sequence\_length):

res = np.load(os.path.join(DATA\_PATH, action, str(sequence), "{}.npy".format(frame\_num)))

window.append(res)

sequences.append(window)

labels.append(label\_map[action])

Перед тем как визуализировать данные необходимо их обработать. Произведя визуализацию можно увидеть отсутствие данных в некоторых кадрах, т.н. пробелы. Таким образом для заполнения пробелов необходимо использовать интерполяцию, в частности линейную. Алгоритм интерполяции данных следующий:



Рисунок - 2.3 Блок-схема интерполяции данных

Данные, извлеченные из файлов, передаются в функцию. Далее происходит разделение данных по координатам на наборы X, Y и Z соответственно. В последствии каждый набор линейно интерполируется по всей последовательности. Эти операции выполняет следующая функция:

def interp\_coords(x):

coords = []

for num,i in enumerate(x):

if np.count\_nonzero(i) != 0:

coords.append([i,num])

result = []

for i in range(63):

result.append(np.interp(range(40),[e[1] for e in coords],[e[0][i] for e in coords]))

return np.array(result).transpose()

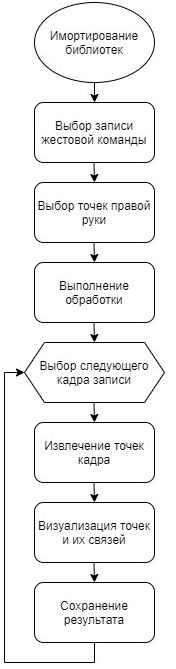


Рисунок - 2.4 Блок-схема получения визуализации

Затем можно приступать к визуализации. Для визуализации реализована функция, принимающая следующие параметры: координаты ключевых точек кисти руки, связи ключевых точек, флаги сохранения, фиксированного масштаба координатных осей и динамического изменения точки зрения. В функции для каждого кадра извлекаются ключевые точки и помещаются в трёхмерный график. Затем на график помещаются связи между ключевыми точками. После этого выбирается масштаб осей автоматический или фиксированный, демонстрирующий всю область видимости ключевых точек, в зависимости от установленного флага fixed\_axes. И выбирается динамическая или статическая точка зрения на график флагом dynamic\_view. В зависимости от значения флага save результат сохраняется. В конечном итоге с помощью библиотеки MatPlotLib получаем визуализацию каждого отдельного кадра. Например, ()

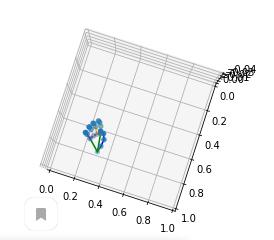


Рисунок - 2.5 Пример визуализации кадра

Код данной функции и используемых внутри неё:

def draw\_parts(ax, dx, dy, dz, parts):

ax.plot3D([dx[parts[0]],dx[parts[1]]], [dy[parts[0]],dy[parts[1]]], [dz[parts[0]],dz[parts[1]]], parts[6])

ax.plot3D([dx[parts[2]],dx[parts[3]]], [dy[parts[2]],dy[parts[3]]], [dz[parts[2]],dz[parts[3]]], parts[6])

ax.plot3D([dx[parts[4]],dx[parts[5]]], [dy[parts[4]],dy[parts[5]]], [dz[parts[4]],dz[parts[5]]], parts[6])

def get\_coords(X):

dx = X[:63:3]

dy = X[1:63:3]

dz = X[2:63:3]

return dx,dy,dz

def draw\_hand(x, parts, save=False, fixed\_axes=False, dinamic\_view=False):

for num,i in enumerate(x):

dx,dy,dz = get\_coords(i)

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d', label=num)

ax.scatter(dx, dy, dz)

for i in parts:

draw\_parts(ax, dx, dy, dz, i)

ax.view\_init(260, -120)

figure = plt.gcf()

if fixed\_axes:

plt.xlim([0,1])

plt.ylim([0,1])

if dinamic\_view:

ax.view\_init(260+2\*num, -120+4\*num)

plt.show()

if save:

figure.savefig(os.path.join("images", "{}.jpg".format(num)))

plt.close(fig)

Описанная функция ничего не возвращает, только демонстрирует matplotlib графики для каждого кадра записи движения. Внутри функции используются функции get\_coords для получения координат конкретного кадра и функция draw\_parts для визуализации связей между ключевыми точками.

# ЗАКЛЮЧЕНИЕ

Таким образом в ходе выполнения курсовой работы были определены жестовые команды управления и удалось реализовать алгоритм обработки и визуализации данных ключевых точек кисти руки для последующего их использования в обучении рекуррентной нейронной LSTM сети, определяющей жестовые команды на видеопотоке.

Для достижения поставленной цели были выполнены следующие задачи:

1. Определен набор жестовых команд, данные которых будут обрабатываться и визуализироваться;
2. Получен набор данных (датасет) отображающих соответствие параметров измеряемых физических пар-ров выбранным жестовым командам.
3. Выполнено определение связей между ключевыми точками кисти руки.
4. Получена визуализацию каждого кадра жеста руки;
5. Проанализирована полученная визуализация и произведена обработка.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Google MediaPipe Hands [Электронный ресурс]. – Режим доступа: https://google.github.io/mediapipe/solutions/hands.html - Дата доступа: 16.11.2021

2. MatPlotLib [Электронный ресурс]. – Режим доступа: https://matplotlib.org/ - Дата доступа 16.11.2021

# Приложение

Приложение А - Программный код, выполняющий извлечение данных из файлов, обработки и получения визуализации

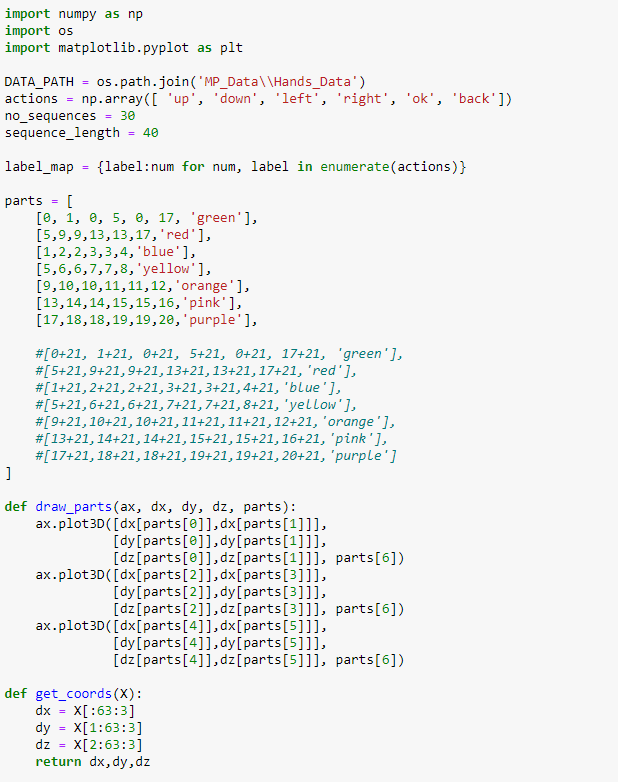


Рисунок – П.1.1

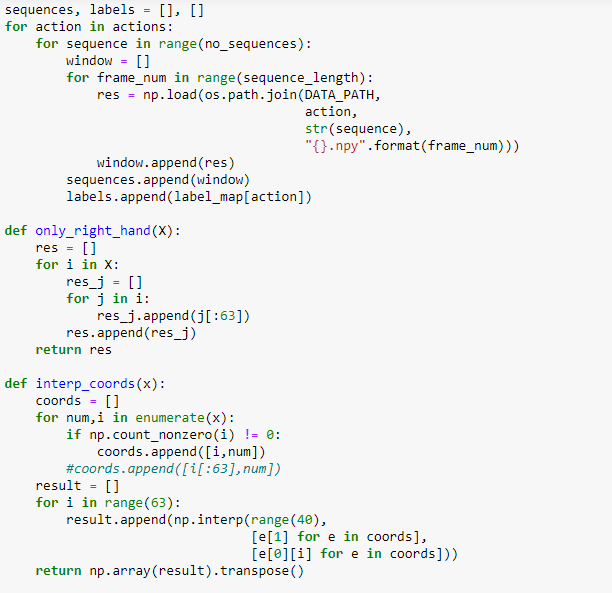


Рисунок – П.1.2

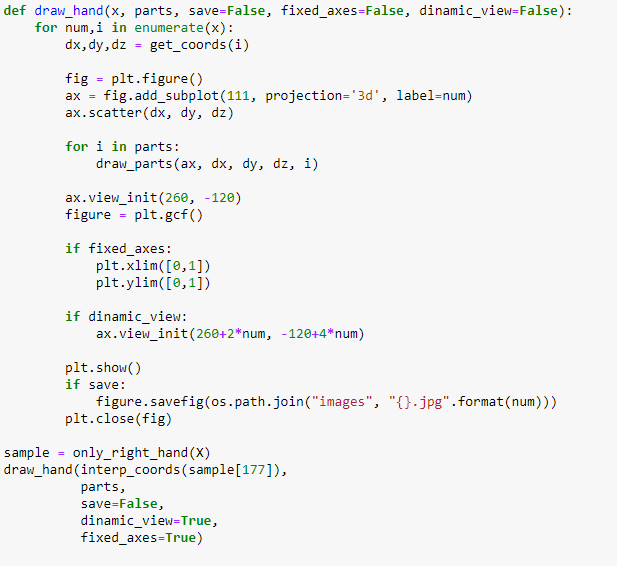


Рисунок – П.1.3

Приложение Б- Графики зависимости отношения сторон ограничительной рамки от номера кадра:

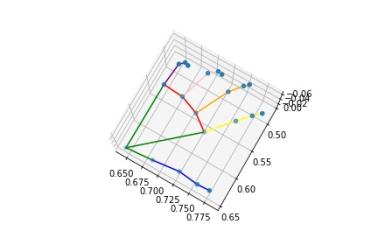
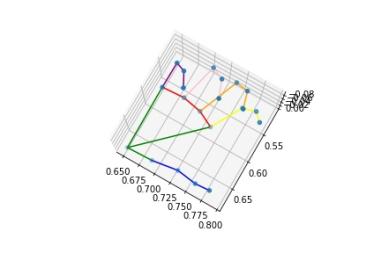
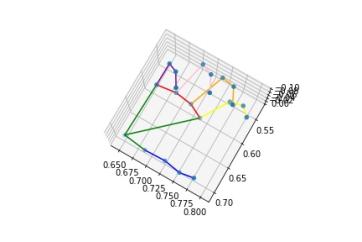
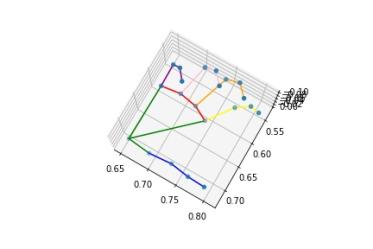
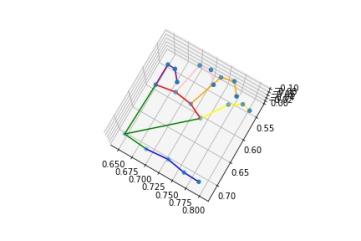
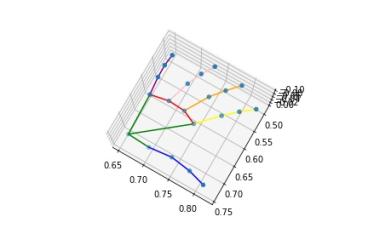


Рисунок – П.2.1

Приложение В - Блок-схемы алгоритма

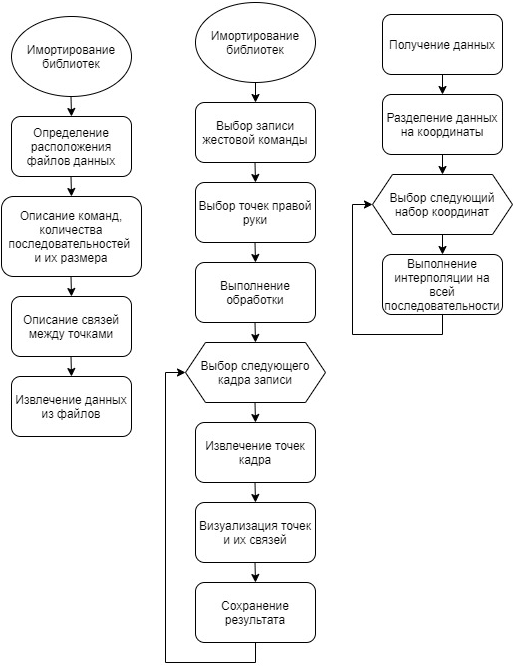


Рисунок – П.3.1