Санкт-Петербургский политехнический университет Петра Великого

Институт металлургии, машиностроения и транспорта

Кафедра «Мехатроника и роботостроение» при ЦНИИ РТК

.

КУРСОВАЯ РАБОТА

РАЗРАБОТКА МОДУЛЯ ОПРЕДЕЛЕНИЯ ВЫПОЛНЕНИЯ ШАБЛОННЫХ ДВИЖЕНИЙ ИНЕРЦИАЛЬНЫМ КОНТРОЛЛЕРОМ-ПЕРЧАТКОЙ, ОСНОВАННОГО НА МАШИННОМ ОБУЧЕНИИ

Выполнил

студент гр.3341506/90401 А.В. Попков

Проверил А.В. Бахшиев

Санкт-Петербург

2020

СОДЕРЖАНИЕ

Введение 3

1.1 Возможности 5

1.2 Сфера использования 6

1.3 Выводы по разделу 7

2 Используемый контроллер и его структура 8

2.1 Определение ориентации 8

2.2 Определение изгиба пальцев 11

2.3 Определение шаблонных движений 12

2.4 Структурная схема аппаратной части 15

2.5 Структурная схема программной части 16

2.6 Блок определения выполнения шаблонных движений 17

2.7 Выводы по разделу 19

3 Машинное обучение 21

3.1 Генерация набора перемещений 21

3.2 Выделение характерных признаков 28

3.3 Алгоритм машинного обучения 31

3.3 Выводы по разделу 37

Заключение 38

Список использованных источников 39

ВВЕДЕНИЕ

С появлением большего числа различных робототехнических средств, возникает все больше проблем в их управлении [1], некоторые из них связаны с неудобством использования стандартных контроллеров. С каждым днем создается все больше новых способов управления и контроллеров, с целью найти более удобный вариант для управления каким-либо конкретным средством.

Управляющие контроллеры входят в состав пользовательского интерфейса. Пользовательский интерфейс – интерфейс, обеспечивающий передачу информации между пользователем-человеком и программно-аппаратными компонентами компьютерной системы [2]. Сами пользовательские интерфейсы могут подразделяться на:

– визуальный (графический и текстовый, который включает в себя командную строку);

– жестовый;

– голосовой;

– осязательный (материальный).

Перечисленные интерфейсы также могут быть скомбинированы между собой.

В настоящей работе будет рассмотрен контроллер, как часть жестового пользовательского интерфейса, с возможностью определения ориентации и распознавания шаблонных движений. Цель контроллера является формирование сигналов управления, зависящих от ориентации контроллера, а также от совершения шаблонных движений или жестов. Также в рамках работы будет предложен способ определения выполнения шаблонных движений инерциальным контроллером-перчаткой, основанный на машинном обучении.

Целью работы является разработка алгоритма определения характера движения, выполняемого контроллером, и отношения его к конкретному шаблону. Для ее достижения необходимо:

– рассмотреть возможности и особенности используемого контроллера;

– создать набор данных (перемещений) для их исследования, а также для дальнейшего обучения алгоритма машинного обучения;

– выделить характерные признаки, по которым можно было бы предсказать характер движения;

– разработать алгоритм получения этих признаков;

– реализовать алгоритм машинного обучения и проверить его работу на тестовой выборке.

1 Перчатка-контроллер

Перчатка-контроллер или киберперчатка (рисунок 1) – это устройство ввода данных для взаимодействия человека с компьютером в виде перчатки. Различные датчики улавливают движения изгибов пальцев и всей кисти. Активируя сенсоры определёнными движениями, можно выполнять команды, связанные с ними. Затем эти движения пальцев и всей руки расшифровываются с помощью программного обеспечения, идущего вместе с перчаткой. Каждое движение и вращение пальцем или всей рукой означает определенную команду.

Рисунок 1 – Перчатка виртульной реальности Peregrine Glove [3]

1.1 Возможности

В настоящий момент в киберперчатках используются несколько средств, которые позволяют использовать ее, как контроллер.

1.1.1 Определение ориентации в пространстве

В перчатку встраивается инерциальный модуль, содержащий трехосевые акселерометр, гироскоп и, в некоторых случаях, компас, при помощи которых, используя специальные алгоритмы разной сложности, можно определить локальные или глобальные, относительно некоторой системы координат, углы

ориентации самой перчатки. Данные углы используются, как стики перчатки-контроллера, а также в дальнейших вычислениях.

1.1.2 Определение изгиба пальцев

В перчатку встраиваются различные датчики – датчики изгиба, датчики касания и т.д. В зависимости от типа датчика и алгоритма, изгибы пальцев можно использовать, как стики контроллера или как кнопки.

При использовании датчиков изгиба можно получить величину, зависящую от угла изгиба пальцев, что при пересчете можно использовать, как стик. Также можно задать порог фиксирования изгиба, при превышении которого величиной с датчика изгиба, контроллер будет сигнализировать об изгибе и наоборот, т.е. использовать изгибы пальцев как кнопки.

Также есть другой способ использовать изгиб пальцев, как кнопки контроллера. На кончики пальцев перчатки устанавливаются токопроводящий материал, он подключается к источнику питания, такой же материал устанавливается на ладонях и подключается к выводу микроконтроллера, при изгибе пальца микроконтроллер фиксирует замыкание цепи.

1.1.3 Определение перемещений

Некоторые перчатки-контроллеры, со встроенным инерциальным модулем или другими средствами, могут определять свое перемещение в пространстве, а также отслеживать выполнение жестов. Как правило, такие перчатки имеют очень высокую стоимость.

1.2 Сфера использования

В настоящее время киберперчатки, а также другие подобные устройства, используют в основном в VR-технологиях. При использовании очков виртуальной реальности, неудобно пользоваться стандартными средствами ввода – клавиатурой мышью и т.д., поэтому для полного погружения человека в

виртуальную реальность используют привычный жестовый интерфейс и возможности трекинга.

В робототехнике данный контроллер полезен при использовании супервизорного или ручного управления роботом. Выполнение определенного жеста, выполненного контроллером, может расцениваться, как определенная команда роботу. Также робот может непосредственно управляться посредством данных, получаемых с контроллера.

1.3 Выводы по разделу

Рассмотрены технологии, которые могут быть использованы в киберперчатке, а также сфера применения киберперчаток.

2 Используемый контроллер и его структура

В рамках данной работы будет рассматриваться конкретная киберперчатка (рисунок 2), и именно под нее будет разрабатываться алгоритм определения выполнения шаблонных движений.

Рисунок 2 – Прототип рассматриваемой перчатки-контроллера

2.1 Определение ориентации

Для определения ориентации в пространстве в контроллере используется MEMS инерциальный модуль mpu9250 [4]. Модуль содержит акселерометр и гироскоп, с которых, получая и фильтруя данные, можно получить углы Эйлера.

Определение ориентации осуществляется за счет фильтра Маджвика [5] для инерционных систем, включающих в себя акселерометр и гироскоп. Фильтр Маджвика компенсирует смещения углов поворота инерциальной системы, накопившиеся в результате шумов датчиков, смещения нуля гироскопа и др. Фильтр Маджвика, основанный на данных только с акселерометра и гироскопа, может дать довольно точные углы поворота только для двух осей инерционной системы, что используется в данной перчатке. Данный фильтр описывается при помощи кватернионов [6].

Упрощенный фильтр Маджвика для инерционной системы включающей трехосевой акселерометр и трехосевой гироскоп, с силой тяжести направленной вдоль оси z некоторой системы координат, относительно которой отсчитываются углы поворота, приведен в формуле (1).

$$ {}^{S}_{E}\boldsymbol{q}^{(n)} = {}^{S}_{E}\boldsymbol{q}^{(n-1)} + {}^{S}_{E}\dot{\boldsymbol{q}}^{(n)}\Delta\boldsymbol{t} \tag{1} $$

где ${}^{S}_{E}\boldsymbol{q}^{(n)}$ – действующий кватернион, определяющий ориентацию контроллера относительно начала системы отсчета на текущей n итерации, для первой итерации ${}^{S}_{E}\boldsymbol{q}^{(0)} = [1 \quad 0 \quad 0 \quad 0]$;

$\Delta\boldsymbol{t}$ – приращение времени между измерениями;

${}^{S}_{E}\dot{\boldsymbol{q}}^{(n)}$– расчетная скорость изменения ориентации, ее можно определить выражением (2).

$$ {}^{S}_{E}\dot{\boldsymbol{q}}^{(n)} = {}^{S}_{E}\dot{\boldsymbol{q}}_{\omega}{}^{(n)} - \beta {}^{S}_{E}\hat{\dot{\boldsymbol{q}}}_{\epsilon}{}^{(n)} \tag{2} $$

где ${}^{S}_{E}\dot{\boldsymbol{q}}_{\omega}{}^{(n)}$ – скорость изменения ориентации, измеренная гироскопом вычисляется по формуле (3);

${}^{S}_{E}\hat{\dot{\boldsymbol{q}}}_{\epsilon}{}^{(n)}$ – предполагаемая ошибка, полученная на основе измерений показаний с акселерометра, при помощи нее компенсируется возможная ошибка измерений гироскопа, вычисляется по формуле (4);

β – коэффициент усиления фильтра, зависит от ошибки измерения нуля гироскопа.

$$ {}^{S}_{E}\dot{\boldsymbol{q}}_{\omega}{}^{(n)} = \frac{1}{2}\,{}^{S}_{E}\boldsymbol{q}^{(n-1)} \otimes {}^{S}\boldsymbol{\omega}^{(n)}, \tag{3} $$

$$ {}^{S}_{E}\dot{\widehat{\boldsymbol{q}}}_{\epsilon}{}^{(n)} = \frac{\nabla \boldsymbol{f}^{(\mathrm{n})}}{\|\nabla \boldsymbol{f}^{(\mathrm{n})}\|} \tag{4} $$

где ${}^{S}_{E}\boldsymbol{q}^{(n-1)}$ – предыдущий результат оценки ориентации;

${}^{S}\boldsymbol{\omega}^{(n)} = \begin{bmatrix} 0 & \boldsymbol{\omega}_x{}^{(n)} & \boldsymbol{\omega}_y{}^{(n)} & \boldsymbol{\omega}_z{}^{(n)} \end{bmatrix}$ – угловая скорость, измеренная трехосевым гироскопом на данной итерации, представляется в виде кватерниона, составленного из угловых скоростей в разных осях;

$\nabla \boldsymbol{f}^{(\mathrm{n})}$ – градиент, который вычисляется по формуле (5).

$$ \nabla \boldsymbol{f}^{(\mathrm{n})} = \boldsymbol{J}\left({}^{S}_{E}\boldsymbol{q}^{(n-1)}\right)^{T} \boldsymbol{f}\left({}^{S}_{E}\boldsymbol{q}^{(n-1)}, {}^{S}\boldsymbol{a}^{(n)}\right) \tag{5} $$

где ${}^{S}\boldsymbol{a}^{(n)} = \begin{bmatrix} 0 & \boldsymbol{a}_x{}^{(n)} & \boldsymbol{a}_y{}^{(n)} & \boldsymbol{a}_z{}^{(n)} \end{bmatrix}$ – ускорение, измеренное с трехосевого акселерометра на данной итерации, представляется в виде кватерниона, составленного из ускорений в разных осях;

$\boldsymbol{f}$ – целевая функция, которая вычисляется по формуле (6);

$\boldsymbol{J}$ – Якобиан целевой функции, вычисляется по формуле (7).

$$ \boldsymbol{f}\left({}^{S}_{E}\boldsymbol{q}, {}^{S}\boldsymbol{a}\right) = \begin{bmatrix} 2(q_2 q_4 - q_1 q_3) - a_x \\ 2(q_1 q_2 + q_3 q_4) - a_y \\ 2\left(\frac{1}{2} - q_2^2 - q_3^2\right) - a_z \end{bmatrix}, \tag{6} $$

$$ \boldsymbol{J}\left({}^{S}_{E}\boldsymbol{q}\right) = \begin{bmatrix} -2q_3 & 2q_4 & -2q_1 & 2q_2 \\ 2q_2 & 2q_1 & 2q_4 & 2q_3 \\ 0 & -4q_2 & -4q_3 & 0 \end{bmatrix} \tag{7} $$

где $q1$,$q2$,$q3$,$q4$ − составляющие параметрического кватерниона ${}^{S}_{E}\boldsymbol{q}$;

ax, ay, az − составляющие параметрического кватерниона ${}^{S}\boldsymbol{a}$.

Получить углы Эйлера из текущего кватерниона можно по формуле (8).

$$\begin{cases} yaw = atan2(2q_2q_3 - 2q_1q_4,\ 2{q_1}^2 + 2{q_2}^2 - 1) \\ pitch = -\text{asin}(2q_2q_4 + 2q_1q_3) \\ roll = atan2(2q_3q_4 - 2q_1q_2,\ 2{q_1}^2 + 2{q_4}^2 - 1) \end{cases} \quad (8)$$

где yaw, $pitch$, $roll$ – углы Эйлера.

2.2 Определение изгиба пальцев

Для определения изгиба в перчатке используются датчики изгиба, которые после вычислений с некоторой погрешностью могут дать величину изгиба пальцев.

Для определения изгибов пальцев в перчатке используются оптические датчики изгиба. В данном датчике используется гибкая светопроводящая трубка, покрытая светопоглощающим материалом, по краям этой трубки установлены: светодиод, как источник света, и фоторезистор, как приемник света. При изгибе и превышении некоторого угла полного внутреннего отражения, часть света выходит за пределы светопроводника и поглощается материалом внешнего слоя (рисунок 3).

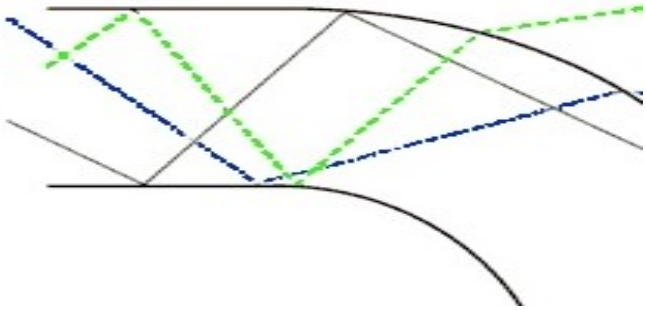

Рисунок 3 – Преломление света в светопроводнике датчика изгиба [7]

У данного датчика есть недостатки в виде повышенного энергопотребления, нелинейной зависимости сопротивления от величины изгиба, предельного радиуса изгиба после которого трубка пережимается и датчик работает неправильно. Преимущества данного датчика в том, что он имеет низкую стоимость, имеет широкий диапазон измерений, что позволяет включать его в цепь без специальных компонентов в виде операционных усилителей, также данный датчик прост в изготовлении.

2.3 Определение шаблонных движений

В данной перчатке уже существует алгоритм определения шаблонных движений. Он также использует данные о перемещении, однако данный алгоритм не дает особой точности при определении движений, имеет высокую сложность настройки, а также в некоторых случаях не инвариантен к скорости выполнения конкретного движения.

В системе, использующей только гироскоп и акселерометр, невозможно точно определить абсолютное перемещение объекта, поэтому в перчатке измеряется небольшие относительные перемещения контроллера, с последующей небольшой задержкой на компенсацию смещений углов фильтром Маджвика.

Рассмотрим систему измерения ускорения MEMS акселерометром (рисунок 4).

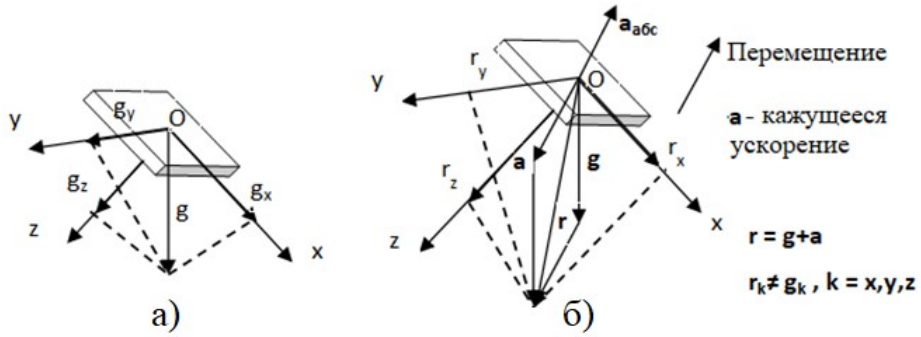

Рисунок 4 – а) измерение акселерометром только ускорения свободного падения - **g**, когда он находится в состоянии покоя или движется равномерно прямолинейно, б) измерение акселерометром результирующего кажущегося ускорения **r**, когда его начинают перемещать

Для получения перемещения, необходимо узнать абсолютное ускорение $\vec{a}_{абс}$, акселерометр же измеряет общее кажущееся ускорение $\vec{r}$, однако, зная углы Эйлера, которые мы определили ранее, относительно некоторой глобальной системы координат, мы можем убрать из $\vec{r}$ компоненту ускорения свободного падения $\vec{g}$ – формула (9).

$$\vec{a}_{абс} = \vec{r} + \vec{g} \tag{9}$$

где $\vec{a}_{абс}$ – абсолютное ускорение, далее $a_{абс}$ будет обозначаться, как a;

$\vec{r}$ – общее кажущееся ускорение;

$\vec{g}$ - ускорение свободного падения.

Перемещение определяется итерационным методом, где каждая итерация – одно измерение ускорения – формулы (10) и (11).

$$S^{(n)} = \frac{\Delta t^2 a^{(n)}}{2} + V^{(n-1)}\Delta t + S^{(n-1)} \quad (10)$$

$$V^{(n)} = \Delta t a^{(n)} + V^{(n-1)} \quad (11)$$

где $S^{(n)}$ – перемещение на n итерации;

$V^{(n)}$ – скорость на n итерации;

$a^{(n)}$ – ускорение на n итерации;

Δt – приращение времени между измерениями.

Полученное таким образом перемещение в отсутствии какой-либо компенсации будет довольно быстро уходить в бесконечность, поэтому для распознавания шаблонных движений желательно переставлять глобальную систему координат в точку начала движения, при этом сбрасывая предыдущие показатели скорости и перемещения – формула (12).

$$V^{(n-1)} = 0, \; S^{(n-1)} = 0 \quad (12)$$

где $S^{(n-1)}$ – перемещение на предыдущей итерации;

$V^{(n-1)}$ – скорость на предыдущей итерации.

При выполнении каждого шаблонного движения создается новая система координат, относительно которой будет определяться выполнение жеста, из-за этого производится компенсация ориентации контроллера, единственное накладываемое ограничение – время выполнения шаблонного движения не должно превышать некоторой величины после которого смещение углов превысит некоторый критический уровень.

По вышеописанным причинам человеку, использующему контроллер, придется сигнализировать о начале и окончании движения (например, нажатием кнопки на контроллере: если кнопка была нажата – началось движение, если кнопка была отжата – движение закончилось), т.к. из-за дрейфа перемещения будет невозможно определить их.

Есть и другие способы оповещения контроллера о начале и окончании выполнения шаблонного движения, как уже говорилось, каждое шаблонное движение, для его однозначного определения, должно сопровождаться активацией и деактивацией. В данном контроллере в качестве активации и деактивации определения перемещения предлагается использовать статичные жесты, изгибы пальцев, ориентация джойстика. Такой способ позволяет создавать и использовать множество шаблонных движений, не теряя удобство использования, также можно еще больше повысить эргономичность управления, если использовать в качестве активации и деактивации невербальные человеческие жесты. При определенном шаблонном движении, человек инстинктивно совершает какой-либо жест, например, сгибание мизинца, среднего и безымянного пальца при перелистывании страницы.

2.4 Структурная схема аппаратной части

На рисунке 5 представлена структурная схема аппаратной части контроллера. В системе присутствует микроконтроллер, который опрашивает датчики изгиба и инерциальный сенсор, а также фильтрует и первоначально обрабатывает данные. Далее данные через приемопередатчик попадают на персональный компьютер или другое вычислительное устройство, с помощью которого осуществляется дальнейшая обработка и работа с контроллером.

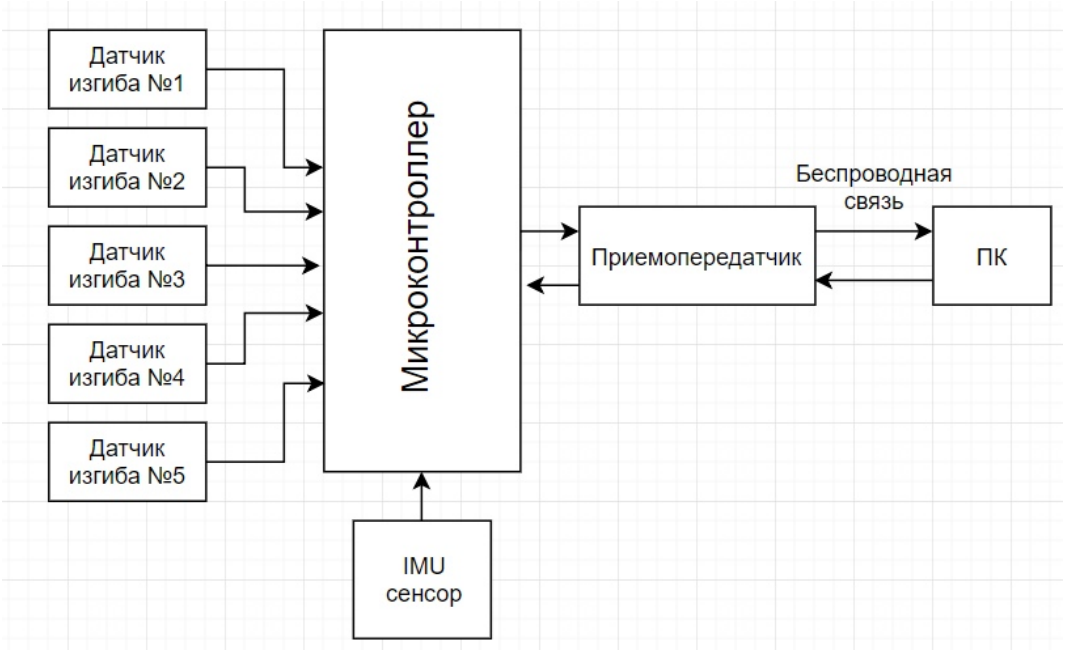

Рисунок 5 – Структурная схема аппаратной части

2.5 Структурная схема программной части

Программная составляющая системы делится на две части (рисунок 6). Первая часть относится к самому контроллеру, вторая часть является программным интерфейсом, с которым взаимодействует пользователь. Пользователь через конфигурацию задает пользовательские шаблоны движений, которые будут проверяться на выполнение. При выполнении конкретного движения, оно будет сопоставлено с шаблоном в блоке определения выполнения пользовательских шаблонов. Если движение совпало с шаблоном, то активируется событие о выполнении конкретного движения.

Через блок контроля работы пользователь может управлять состоянием перчатки, калибровать датчики и т.д.

Блок приемо-передачи данных и в контроллере, и на ПК выполняет прием и передачу пакетов, их упаковку и распаковку, и устранение ошибочных пакетов.

Блок управления перчаткой по приходящим пакетам задает состояние перчатки. Блок работы с датчиками считывает данные с датчиков, фильтрует и первоначально обрабатывает их.

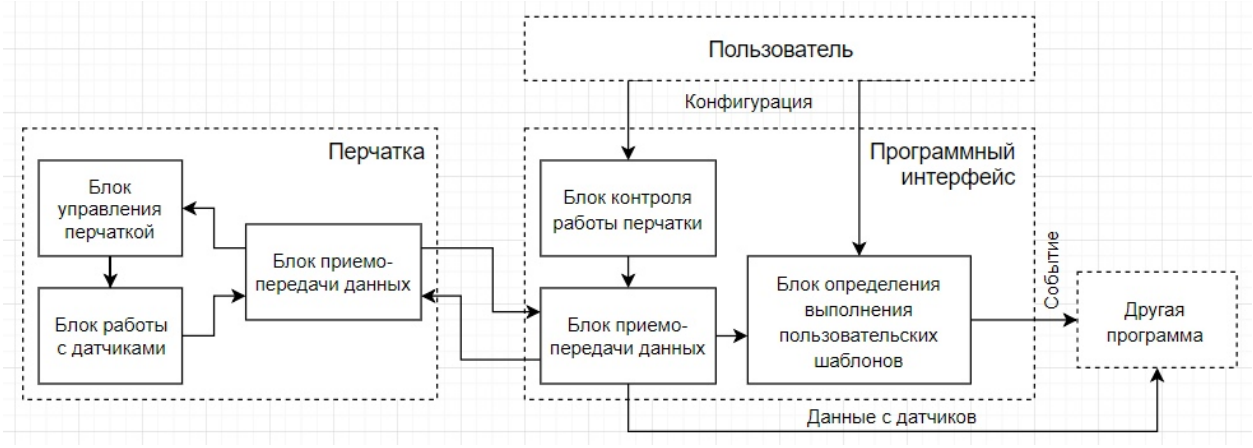

Рисунок 6 – Структурная схема программной части

2.6 Блок определения выполнения шаблонных движений

Когда приходит новый пакет данных от устройства, вызывается событие, обработчиком которого является определение выполнения шаблонных движений. Конечный автомат данного блока схематично представлен на рисунке 7.

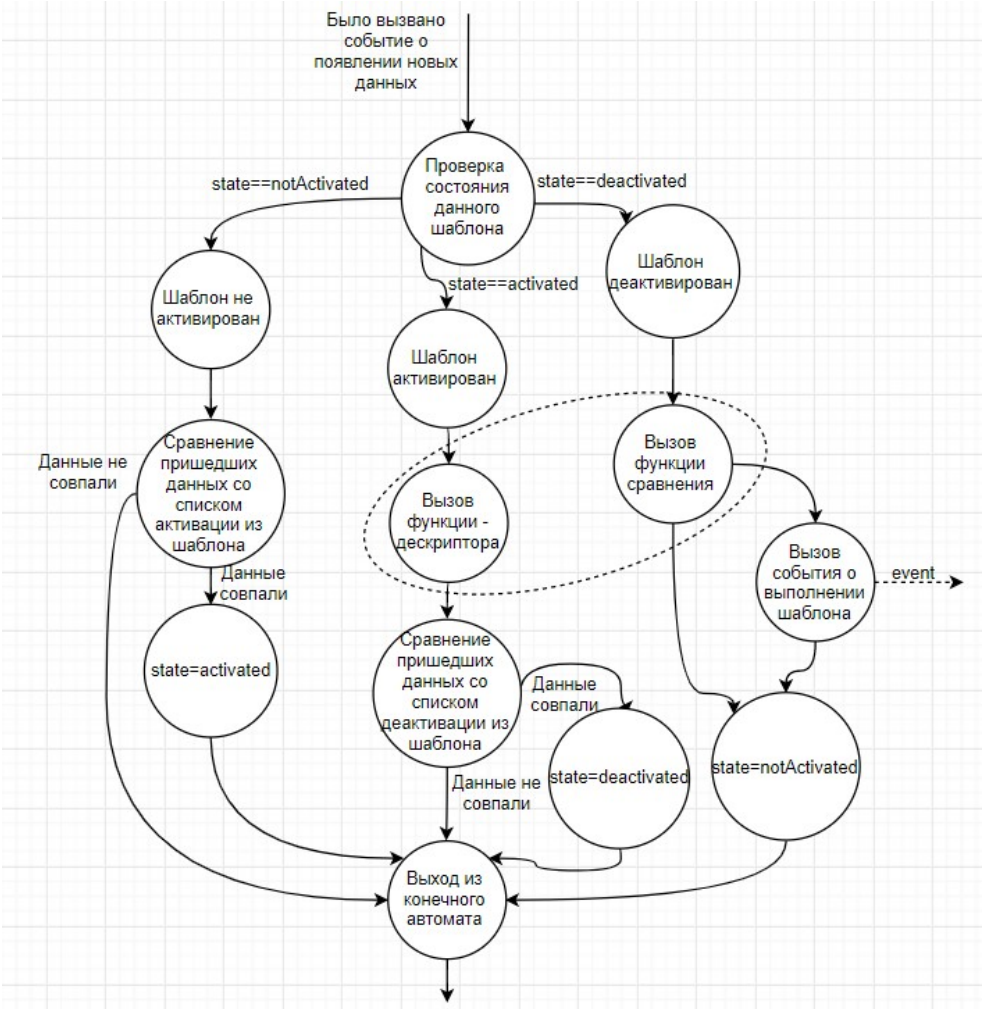

Рисунок 7 – Конечный автомат блока определения шаблонных движений

При появлении новых данных данный алгоритм последовательно применяется к каждому загруженному пользователем шаблону.

Конечный автомат имеет три главных состояния – шаблон не активирован, активирован и деактивирован. Подробнее про активацию, деактивацию было написано в подразделе 2.3.

При вхождении в конечный автомат проверяется текущее состояние (state) текущего шаблона.

Если шаблон еще не активирован (state равно notActivated), то, пока не придут данные, совпадающие со списком активации текущего шаблона, конечный автомат не выйдет из этого состояния, и каждые пришедшие данные будут пропускаться данным шаблоном, в противном случае конечный автомат

переключит свое состояние (state присваивается состояние activated), т.е. активирует шаблон, и при следующем вхождении в конечный автомат программа пойдет по другой ветви алгоритма.

Если шаблон уже был активирован (state равно activated), будет вызвана функция дескриптор, которая определяет все параметры для каждой итерации, необходимые для дальнейшего сравнения с пользовательским шаблоном (скорости, перемещения и т.д.) и сохраняет их в некоторой области видимости. После этого производится сравнение пришедших данных со списком деактивации из шаблона, если они не совпадают, то на каждой итерации повторно будет вызываться функция дескриптор, пока данные из списка деактивации не совпадут с пришедшими и шаблон не будет деактивирован (state присваивается состояние deactivated). В результате работы этой ветви алгоритма, для каждого шаблона будут получены характеристики движения, которые в дальнейшем можно сравнивать с шаблонными характеристиками.

Если шаблон был деактивирован (state равно deactivated), будет вызвана функция сравнения, которая сравнивает характеристики движения, определенные функцией-дескриптором с характеристиками, записанными в шаблоне движения. После сравнения конечный автомат меняет свое состояние (state присваивается состояние notActivated), но если сравниваемые данные совпали, то вызывается событие о выполнении данного шаблона.

В данной работе будет предложен свой блок определения выполнения пользовательских шаблонов, на замену текущему используемому в перчатке, а именно будет заменена функция сравнения, которая будет сравнивать выполненные движения с некоторыми шаблонами.

2.7 Выводы по разделу

Были рассмотрены технологии и средства, используемые в инерциальном контроллере-перчатке, а также структурные схемы аппаратной и программной части.

В результате проведенного анализа, можно заключить несколько фактов:

– Выходными данными перчатки являются значения ускорения $\boldsymbol{a}^{(n)} = \left[\boldsymbol{a}_x{}^{(n)} \quad \boldsymbol{a}_y{}^{(n)} \quad \boldsymbol{a}_z{}^{(n)}\right]$, угловой скорости $\boldsymbol{\omega}^{(n)} = \left[\boldsymbol{\omega}_x{}^{(n)} \quad \boldsymbol{\omega}_y{}^{(n)} \quad \boldsymbol{\omega}_z{}^{(n)}\right]$, данных об изгибах пальцев $\boldsymbol{d}^{(n)} = \left[\boldsymbol{d}_1{}^{(n)} \quad \boldsymbol{d}_2{}^{(n)} \quad \boldsymbol{d}_3{}^{(n)} \quad \boldsymbol{d}_4{}^{(n)} \quad \boldsymbol{d}_5{}^{(n)}\right]$, промежутка времени с предыдущего измерения $\boldsymbol{dt}^{(n)}$ на n-итерации. Из данных об ускорении, угловой скорости и времени высчитываются углы Эйлера $\boldsymbol{yaw, pitch, roll}$;

– по текущему алгоритму шаблонные движения определяются при помощи определения небольших локальных перемещений $\boldsymbol{S}^{(n)}$, относительно системы координат в момент сигнала начала движения;

– мировая система координат не жесткая, она определяется в момент начала движения, согласно соответствующим осям MEMS датчиков – допущение сделанное, т.к. в текущем варианте перчатки задействован только акселерометр и гироскоп, из-за чего нельзя точно задать мировые оси. Допущение смягчается тем, что активация движения производится только при заданных в шаблоне условиях о состоянии $\boldsymbol{a}$, $\boldsymbol{\omega}$, $\boldsymbol{yaw, pitch, roll}$, $\boldsymbol{d}$, $\boldsymbol{dt}$ – это означает, что несмотря на то, что само перемещение не связано с мировыми координатами, само шаблонное движение может быть с ними связано при помощи сырых параметров (например $\boldsymbol{yaw, pitch, roll}$). Из-за этого, например, линейное перемещение перчатки вверх-вниз, влево-вправо не будет определяться, как одно и то же шаблонное движение;

– система не может определять слишком сложные шаблонные движения из-за быстрого накопления ошибки перемещения, движения ограничены некоторым временем.

– распознавание шаблонных перемещений, в данный момент, производится путем проверки вхождения всех полученных точек $\boldsymbol{S(t)}$ в шаблон, содержащий верхнюю $\boldsymbol{M(t)}$ и нижнюю $\boldsymbol{m(t)}$ ограничивающую последовательность, т.е. проверки условия $\boldsymbol{M(t) > S(t) > m(t)}$.

3 Машинное обучение

По описанным в п. 2.6 фактам можно сделать несколько заключений:

– Задача определения шаблонного движения должна хорошо решаться при помощи задачи классификации машинного обучения;

– можно было бы попробовать определять шаблонные движения по непосредственным сырым данным с перчатки, не прибегая к двойному интегрированию без компенсации ошибки (т.е. по ускорению и углам Эйлера), однако тогда непонятно по каким признакам можно было бы сравнивать движения;

– если определять шаблонные движения по перемещению, то входными данными общего алгоритма будет являться набор точек $\boldsymbol{S(t)} = [\boldsymbol{S_x(t)} \quad \boldsymbol{S_y(t)} \quad \boldsymbol{S_z(t)}]$, причем известно, что $\boldsymbol{S(0)} = \boldsymbol{0}$, а также, что система координат набора точек всегда совпадает с начальной системой координат шаблона, т.е. можно не заботиться о том, что полученные данные $\boldsymbol{S(t)}$ нужно будет как-то преобразовывать (поворачивать, перемещать), что бы привести их к системе координат шаблона (например, движение перчатки вдоль оси, врезающейся в ладонь – неважно в какую сторону будет происходить движение, оно будет восприниматься только как перемещение вдоль $\boldsymbol{S_z}$).

3.1 Генерация набора перемещений

Для дальнейшего исследования, необходимо создать набор данных о перемещениях контроллера в пространстве. Изначально предполагалось сгенерировать данные, снимая их непосредственно с самой перчатки, однако по техническим причинам, это оказалось невозможным. Вместо этого было решено искусственно сгенерировать данные о шаблонных перемещениях.

Для начала, было создано три шаблона движения: логарифмическая спираль, подобие прямоугольника и прямая линия.

3.1.1 Логарифмическая спираль

Логарифмическую спираль можно описать формулой (13), и выглядит она так, как представлено на рисунке 8.

$$\begin{cases} r = a \cdot e^{b \cdot t} \\ S_x(t) = -r \cdot cos(c \cdot t) + a \\ S_y(t) = r \cdot sin(c \cdot t) + a \\ S_z(t) = 0 \end{cases} \quad (13)$$

где $S_x(t), S_y(t), S_z(t)$ – перемещения;

a, b, c – коэффициенты логарифмической спирали;

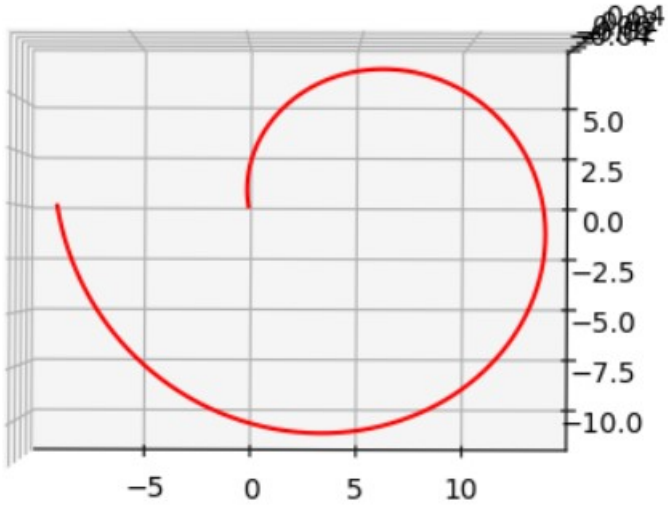

Рисунок 8 – Пример логарифмической спирали

Для обучения алгоритма необходимо создать выборку близкую к реальным данным, поэтому к данному шаблону необходимо добавить значения интегральной ошибки.

Для этого, согласно формуле (10), к значению чистого ускорения необходимо прибавить значения ошибки измеренного ускорения ${a_{err}}^{(n)}$ (формула (14))

$$\begin{cases} a^{(n)} = {a_{real}}^{(n)} + {a_{err}}^{(n)} \\ S^{(n)} = \dfrac{\Delta t^2\left({a_{real}}^{(n)} + {a_{err}}^{(n)}\right)}{2} + V^{(n-1)}\Delta t + S^{(n-1)} \\ V^{(n)} = \Delta t\left({a_{real}}^{(n)} + {a_{err}}^{(n)}\right) + V^{(n-1)} \end{cases} \quad (14)$$

Из формулы (14) можно выявить формулы (15) и (16):

$$\begin{cases} S^{(n)} = {S_{real}}^{(n)} + {S_{err}}^{(n)} \\ V^{(n)} = {V_{real}}^{(n)} + {V_{err}}^{(n)} \\ {S_{err}}^{(n)} = \dfrac{\Delta t^2 \cdot {a_{err}}^{(n)}}{2} + {V_{err}}^{(n-1)}\Delta t + {S_{err}}^{(n-1)} \\ {V_{err}}^{(n)} = \Delta t \cdot {a_{err}}^{(n)} + {V_{err}}^{(n-1)} \end{cases} \quad (15)$$

$${a_{err}}^{(n)} = scale \cdot random(0,1) - 0{,}5 \quad (16)$$

где ${S_{real}}^{(n)}$ – реальное перемещение, которое берется напрямую из чистого шаблона из формулы 13;

${S_{err}}^{(n)}$ – ошибка перемещения;

${V_{err}}^{(n)}$ – ошибка скорости;

${a_{err}}^{(n)}$ – ошибка ускорения, которая генерируется случайным образом.

Пример логарифмической спирали, с некоторым значением ошибки, можно увидеть на рисунке 9.

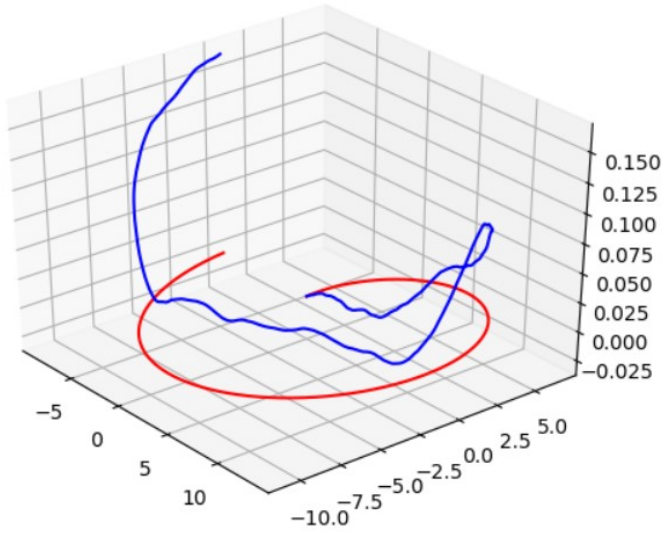

Рисунок 9 – Действительная логарифмическая спираль и логарифмическая спираль с некоторым значением ошибки

Из формулы полученной логарифмической спирали с некоторым значением ошибки и с различными параметрами теперь необходимо создать датасет, в данной работе решено было ограничиться набором из 1000 кривых. На рисунке 10 показано часть из них, а именно 100 кривых.

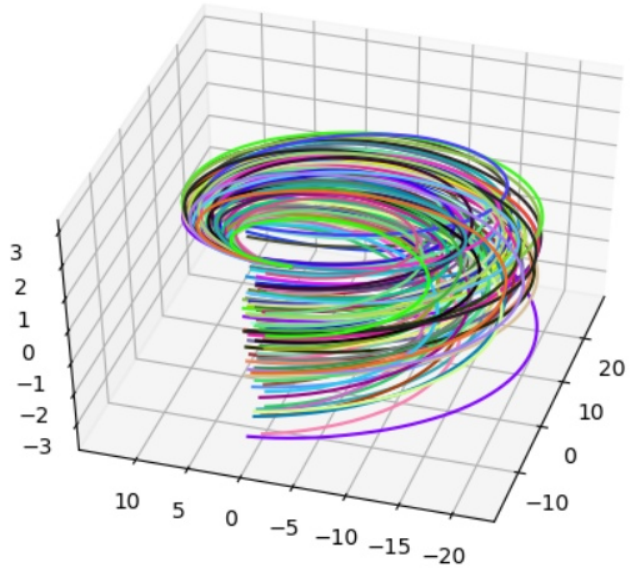

Рисунок 10 – Набор логарифмических спиралей с ошибкой

3.1.2 Псевдопрямоугольник

Аналогично логарифмической спирали были также сгенерированы подобия прямоугольников, которые можно описать по формуле 17.

$$\begin{cases} t = t_0, .. t_1, .. t_2, .. t_3, .. t_4 \\ fe(t) = \begin{vmatrix} (0,0,0)\ if\ t \notin [t_0, t_1) \\ (0,0,a \cdot t)\ if\ t \in [t_0, t_1) \end{vmatrix} \\ se(t) = \begin{vmatrix} (0,0,0)\ if\ t \notin [t_1, t_2) \\ (b \cdot t, 0, a)\ if\ t \in [t_1, t_2) \end{vmatrix} \\ te(t) = \begin{vmatrix} (0,0,0)\ if\ t \notin [t_2, t_3) \\ (b, 0, a - c \cdot t)\ if\ t \in [t_2, t_3) \end{vmatrix} \\ foe(t) = \begin{vmatrix} (0,0,0)\ if\ t \notin [t_3, t_4] \\ (b - d \cdot t, 0, a - c)\ if\ t \in [t_3, t_4] \end{vmatrix} \\ rect(t) = fe(t) + se(t) + te(t) + foe(t) \end{cases} \quad (17)$$

где $rect(t)$ – перемещения по всем трем осям;

a, b, c, d – длины сторон псевдопрямоугольника;

Пример псевдопрямоугольника, полученного по формуле (17) можно увидеть на рисунке 11.

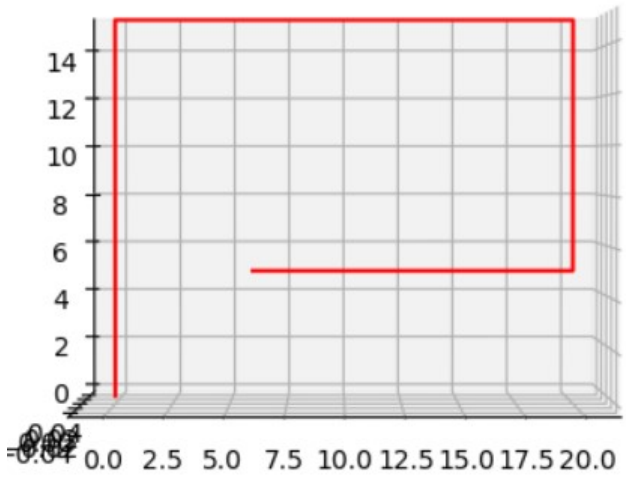

Рисунок 11 – Пример псевдопрямоугольника

Также согласно формулам (15) и (16) была добавлена интегральная ошибка, пример кривой с ошибкой можно увидеть на рисунке 12.

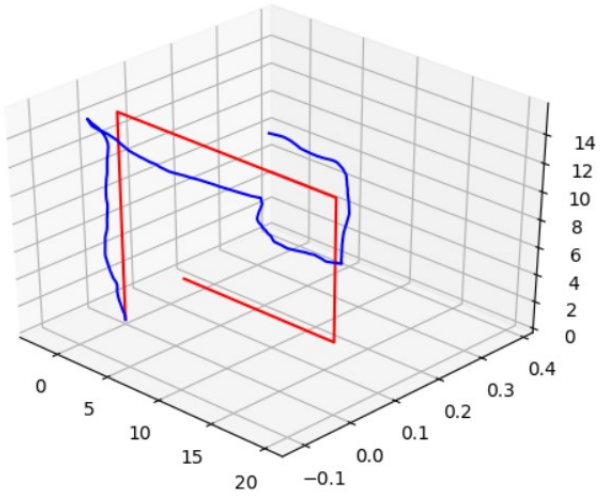

Рисунок 12 – Пример действительного псевдопрямоугольника и псевдопрямоугольника с ошибкой

Из формулы полученного псевдопрямоугольника с некоторым значением ошибки и с различными параметрами теперь необходимо создать датасет, также решено было ограничиться набором из 1000 кривых. На рисунке 13 показано часть из них, а именно 100 кривых.

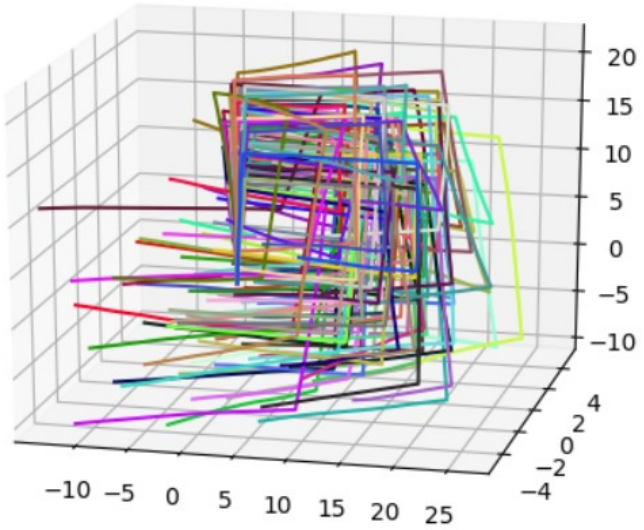

Рисунок 13 – Набор псевдопрямоугольников с ошибкой

3.1.3 Прямая линия

Прямую линию можно описать формулой (18).

$$line(t) = (d \cdot t, 0, 0) \tag{18}$$

где d – длина линии.

Пример линии, построенной по формуле (18) можно увидеть на рисунке 14.

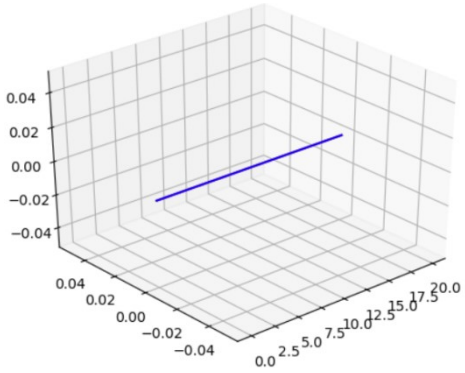

Рисунок 14 – Пример прямой линии

Также согласно формулам (15) и (16) была добавлена интегральная ошибка, пример прямой с ошибкой можно увидеть на рисунке 15.

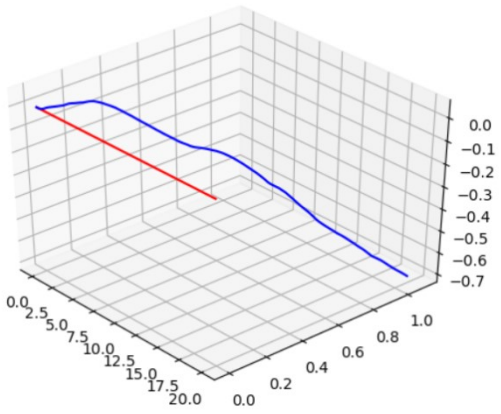

Рисунок 15 – Пример действительной прямой и прямой с ошибкой

По формулам (18), (15), (16) также был сгенерирован датасет с различными параметрами, также решено было ограничиться набором из 1000 кривых. На рисунке 16 показано часть из них, а именно 100 прямых.

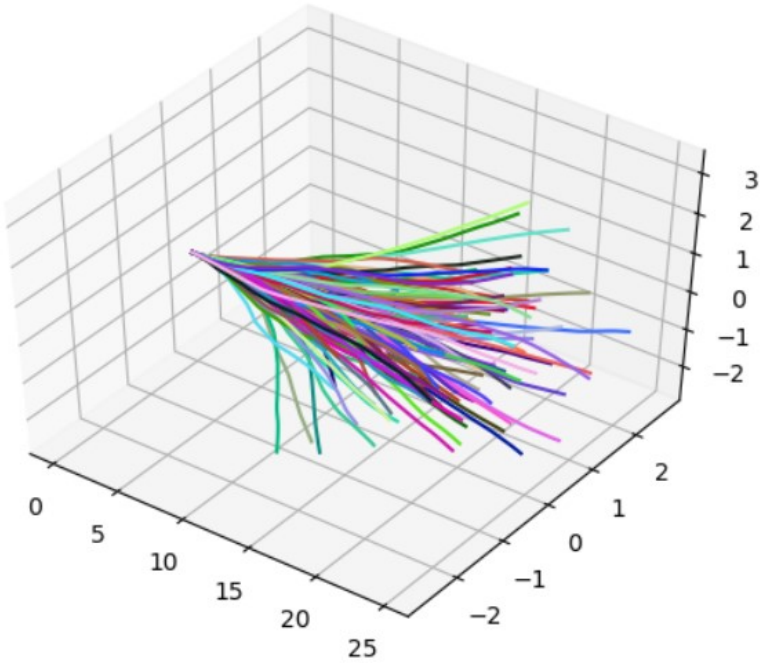

Рисунок 16 – Набор прямых с ошибкой

3.2 Выделение характерных признаков

Для того чтобы определить, какое конкретно перемещение произошло, необходимо определить характер полученной кривой и соотнести ее с определенным шаблоном. Сравнивать линии можно непосредственно и по самому перемещению, однако, тогда не получится достаточно хорошо достичь инвариантности к масштабу, а также сложно будет выявить разрывы и компенсировать небольшие ошибки перемещения. Сравнивать линии в пространстве можно по кривезне самой линии.

В данной работе в качестве признаков перемещений было решено использовать параметры функции кривизны.

В качестве функции (кривой) кривизны может выступать кривая, образованная двойным взятием градиента от изначального перемещения по каждой конкретной его оси с параметром в виде времени (формула (19)).

$$\begin{cases} K_x(t) = grad\left(grad\left(S_x(t)\right)\right) \\ K_y(t) = grad\left(grad\left(S_y(t)\right)\right) \\ K_z(t) = grad\left(grad\left(S_z(t)\right)\right) \end{cases} \quad (19)$$

Пример функции кривизны, которая образуется от произвольной кривой, показана на рисунке 17.

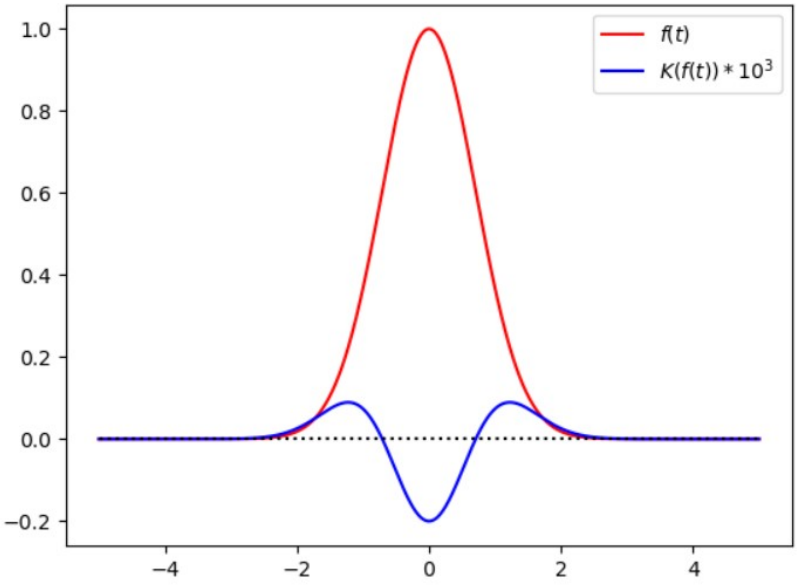

Рисунок 17 – Пример функции кривизны, образующейся от некоторой функции (функция кривизны, для наглядности, увеличена в 1000 раз)

Для большей точности, функция кривизны и перемещения делится на n частей, каждая из которых будет иметь свои признаки, т.к. если брать средние признаки на всю кривую, они могут не иметь достаточной точности для сравнения. Чем больше количество делений будет иметь функция кривизны, тем больше будут выделяться признаки и тем больше можно будет добавить

различных и близких друг к другу шаблонов движений, однако с этим пропорционально растет количество признаков, что уменьшает скорость работы программы. Для набора шаблонов, перечисленных ранее, можно не делить функцию кривизны, т.к. у всех них и так будут различные параметры, перечисленные ранее, однако в работе, для универсальности, функция кривизна поделена на 10 частей.

Из функции кривизны и самого перемещения можно получить следующие параметры:

– Среднее значения $mean(K(t))$ или мода $mode(K(t))$ функции кривизны. Эти параметры могут показать общее поведение кривой – если она близка к прямой, то значение параметров будет близко к 0, в противном случае, параметры будут иметь большие абсолютные значения, если изначальная кривая имеет большую кривизну. Рекомендуется брать моду, а не среднее значение, т.к. если в изначальной функции есть разрывы первого рода, то функция кривизны в некоторых местах будет иметь аномально большие значения, что может испортить среднее значение;

– Максимальное абсолютное значение функции кривизны $max(abs(K(t)))$. Этот параметр покажет, есть ли в наборе точек разрывы первого рода. Этот параметр необходим, если в шаблонных движениях есть перемещения с резкими скачками или прямыми углами и т.д. Пример реакции функции кривизны на разрыв первого рода можно увидеть на рисунке 18;

– Проекционные углы конца каждого из отрезков, на которые были разбиты изначальные перемещения. Альтернативой проекционным углам могут являться знаки координат точки конца каждого из отрезков, которые покажут, в каком из восьми квадратнов находится конец отрезка. Данные параметры необходимы только, если в шаблонах есть идентичные движения, но они инвертированы, учитывается только полная инверсия, если движения просто повернуты, то данные параметры не нужны.

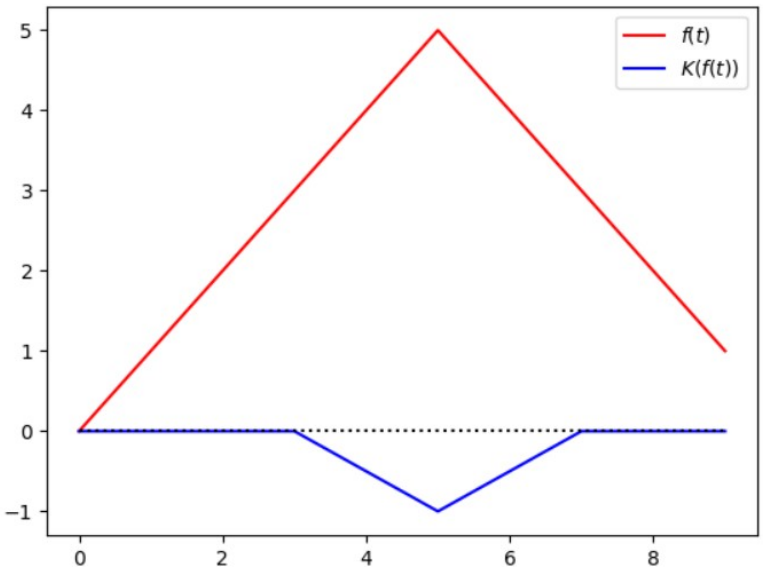

Рисунок 18 – Пример реакции функции кривизны на разрыв первого рода (стоит учесть, что в данном примере масштаб по сравнению с оригинальной функцией сохранен, по сравнению с рисунком 17, где масштаб был увеличен в 1000 раз)

3.3 Алгоритм машинного обучения

Были созданы генераторы датасетов линий, перечисленных ранее, эти датасеты были обработаны и из них были выделены признаки, также были реализованы алгоритмы машинного обучения [8].

3.3.1 Метод ближайших соседей

В качестве алгоритма классификации был выбран метод ближайших соседей. Суть метода в нахождении k ближайших соседей к рассматриваемому объекту, и объекту присваивается тот класс, который присвоен большинству его соседей. Близость соседей определяется расстоянием в системе координат, образованной признаками объектов. Существует и взвешенный вариант метода – когда при голосовании учитывается не только класс соседа, но и его значение веса. Например, если в качестве весов брать расстояние до соседей, то, таким

образом, более близкие соседи имеют больший вес, что может помочь в ситуациях, когда число соседей разных классов (примерно) одинаково.

В работе используется как раз взвешенный вариант метода, где в качестве весов используется расстояние d_i, а значение веса рассчитывается как $\omega_i = \frac{1}{d_i}$. В качестве метрики используется евклидова метрика. Число k ближайших соседей, как выяснилось на основе тестирования алгоритма на тестовой выборке, в разумных масштабах, для небольшого количества шаблонов, не сильно влияет на точность работы алгоритма, поэтому, в зависимости от количества шаблонов, можно задать параметр k в районе 2-5.

3.3.2 Подготовка датасета

Из описанных в п. 3.1 данных были созданы датасеты с ключами $S_x,\ S_y,\ S_z,\ dt,$ которые сохраняют перемещения, а также параметр времени. К этим данным необходимо подобрать признаки, описанные в предыдущем пункте.

Для определения поведения кривой в качестве основных признаков были выбраны мода функции кривизны и максимальное абсолютное значение функции кривизны, т.к. шаблонов с инвертированным движением сгенерировано не было, дополнительный признак, предназначенный для этого, не используется.

Далее, перемещение и функция кривизны были разбиты на 10 частей, для каждой из которых были найдены признаки: мода части функции кривизны $mode(K_{0_x})..mode(K_{9_x}),\ mode\left(K_{0_y}\right)..mode\left(K_{9_y}\right),\ mode(K_{0_z})..mode(K_{9_z})$, а также максимальное абсолютное значение части функции кривизны $\max\left(abs(K_{0_x})\right)..\max\left(abs(K_{9_x})\right),\ \max\left(abs\left(K_{0_y}\right)\right)..\max\left(abs\left(K_{9_y}\right)\right),$ $\max\left(abs(K_{0_z})\right)..\max\left(abs(K_{9_z})\right)$.

Т.к. перемещения и функция кривизны делятся на 10 отрезков, при этом каждый из параметров рассматривается для перемещений в трех ортогональных направлениях – x, y, z, общее количество параметров для каждого шаблонного движения равняется $10 \cdot 3 \cdot 2 = 60$.

Каждому сгенерированному перемещению был присвоен свой численный индекс, который относит его к конкретному шаблонному движению (0 – логарифмическая спираль, 1 – псевдопрямоугольник, 2 – прямая линия). Численное значение индекса является выходом алгоритма классификации, таким образом, определяется принадлежность движения к конкретному шаблону.

Далее, из датасета были выброшены промежуточные данные S_x, S_y, S_z, dt, которые в обучении алгоритма не участвуют, а сам датасет был разбит на две части, одна из которых будет принимать участие в обучении, а другая в тестировании алгоритма.

3.3.2 Обучение и тестирование

В качестве реализации алгоритма классификации был выбран метод ближайших соседей, реализованный в фреймворке sklearn [9]. Было произведено обучение алгоритма на тренировочной выборке, созданной ранее.

На тестовой выборке было произведено тестирование, ошибка тестирования получилась равной нулю ($e_{test} = 0$) . Данный результат обусловлен либо тем, что признаки именно данных шаблонных движений между собой имеют маленькую корреляцию, что должно измениться, если добавить большое количество близких друг к другу шаблонов, либо переобучением алгоритма.

Также в качестве алгоритма классификации было решено попробовать использовать метод решающих деревьев, реализованный в том же фреймворке.

Также было произведено обучение и тестирование алгоритма. При соотношении 4 к 1 обучающей выборке к тестовой, ошибка тестирования

составляет $e_{test} = 0{,}17$, при ошибке обучения $e_{train} = 0$. Само дерево можно увидеть на рисунке 19.

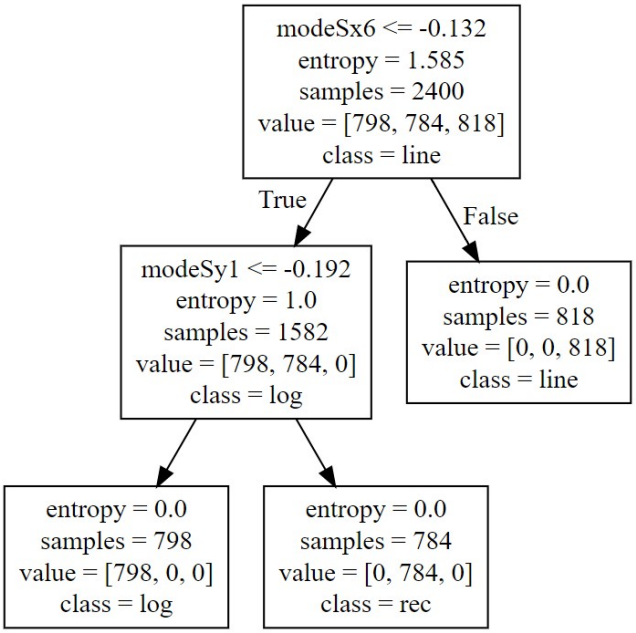

Рисунок 19 – Блок-схема решающего дерева для трех шаблонов

Также для тестирования было решено добавить еще пару шаблонов: псевдотреугольник (рисунок 20) и символ V (рисунок 21).

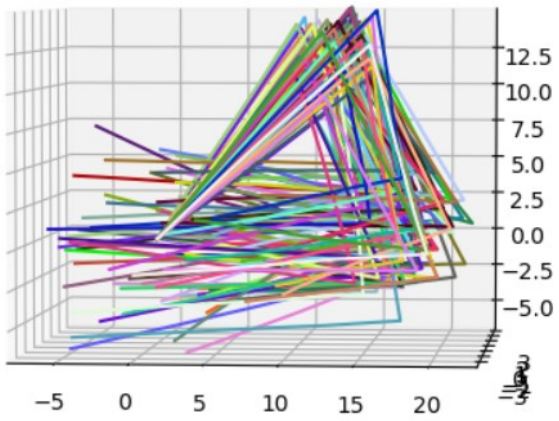

Рисунок 20 – Набор псевдотреугольников

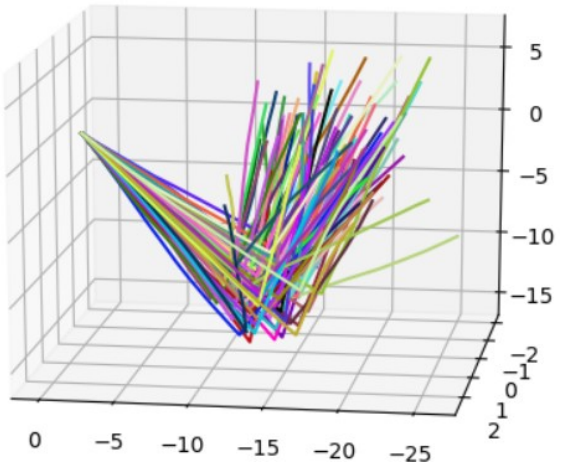

Рисунок 21 – Набор символов V

После добавления псевдотреугольников и обучения алгоритма, ошибка на тестовой выборке у метода решающих деревьев составила $e_{test} = 0{,}125$, при ошибке обучения $e_{train} = 0$. Сам вид решающего дерева можно посмотреть на рисунке 22.

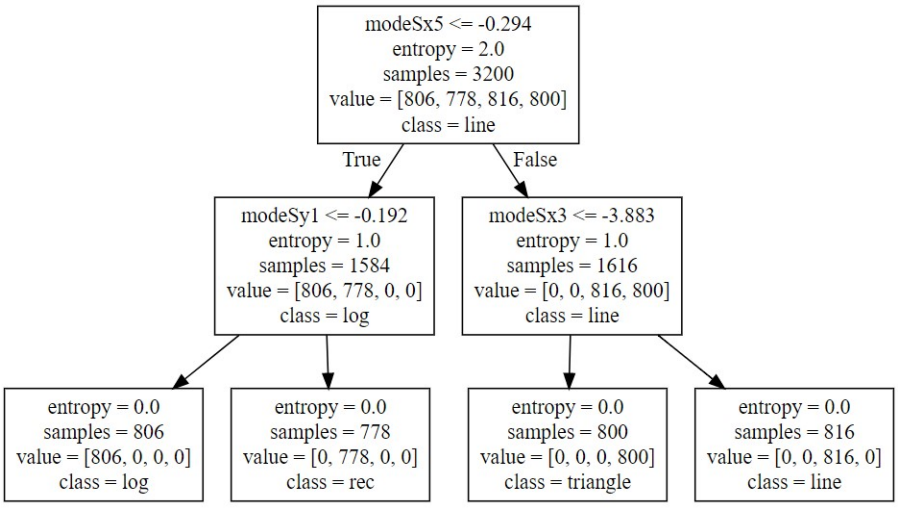

Рисунок 22 – Блок-схема решающего дерева для четырех шаблонов

После добавления символа V и обучения алгоритма, ошибка на тестовой выборке у метода решающих деревьев составила $e_{test} = 0$. Сам вид решающего дерева можно посмотреть на рисунке 23.

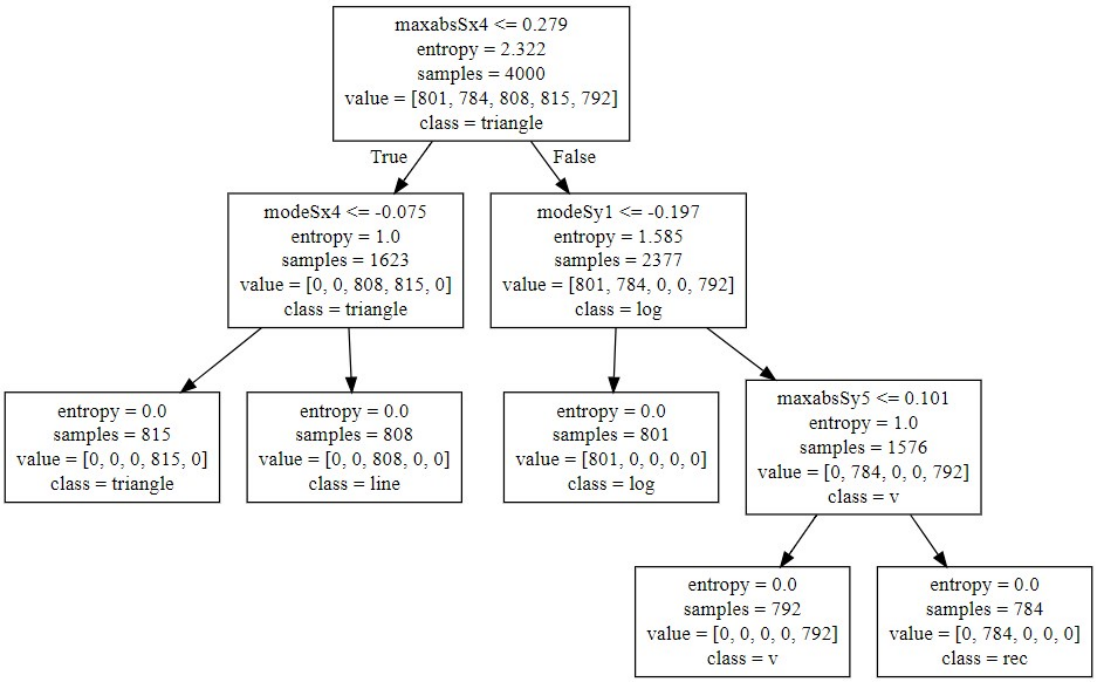

Рисунок 22 – Блок-схема решающего дерева для пяти шаблонов

Для метода ближайших соседей для каждого случая ошибка на тестовой выборке всегда была равна нулю ($e_{test} = 0$).

Из представленных выше результатов можно сделать некоторые выводы:

– Судить о характере движений по какому-либо конкретному признаку нельзя, необходимо рассматривать все пространство признаков в целом – это можно было увидеть в результате использования метода решающих деревьев (слишком большой разброс ошибки, при различных используемых шаблонах).

– Необходимое количество признаков варьируется в зависимости от количества шаблонов и для небольшого количества шаблонов их можно существенно уменьшить, для данного в работе количества шаблонов, вместо 60 признаков, вполне бы хватило 6-12.

3.3 Выводы по разделу

Полученный в результате работы алгоритм определения шаблонных движений, в итоге, алогичен алгоритму, представленному на рисунке 7, за исключением функции сравнения, которая в данной работе заменяется алгоритмом классификации.

Общий алгоритм принимает на вход с контроллера мгновенные значения ускорения $\boldsymbol{a} = [\boldsymbol{a}_x \quad \boldsymbol{a}_y \quad \boldsymbol{a}_z]$, угловой скорости $\boldsymbol{\omega} = [\boldsymbol{\omega}_x \quad \boldsymbol{\omega}_y \quad \boldsymbol{\omega}_z]$, данных об изгибах пальцев $\boldsymbol{d} = [\boldsymbol{d}_1 \quad \boldsymbol{d}_2 \quad \boldsymbol{d}_3 \quad \boldsymbol{d}_4 \quad \boldsymbol{d}_5]$ и промежутка времени с предыдущего измерения $\boldsymbol{dt}$, а также вычисленные на основе этих данных углы Эйлера. Далее, после активации шаблона, на каждой последующей итерации приема данных и до деактивации шаблона, вызывается функция дескриптор, которая сохраняет все полученные данные с момента активации, а также определяет дополнительные данные – локальные перемещения (формула (10)), совершенные от момента активации и до момента деактивации шаблона, подробнее про это написано в подразделах 2.3 и 2.6. Далее, после деактивации, шаблона, вызывается функция сравнения, в которую передаются все сохраненные и полученные в функции дескрипторе данные. В функции сравнения определяется, какое именно из загруженных шаблонных движений произошло и вызывается соответствующее событие о выполнении конкретного шаблона. В качестве функции сравнения, в данной работе, используется алгоритм классификации, который получает на вход значения перемещений вычисленных в функции дескрипторе, на основе которых производится определение – к какому конкретному шаблону относится совершенное движение, в зависимости от полученного результата во внешнюю программу вызывается то или иное событие. В качестве загружаемых пользователем шаблонов движений, в данном алгоритме используется большой набор различных шаблонных движений, на основе которых производится обучение алгоритма классификации (в работе – датасет, генерируемый ранее).

ЗАКЛЮЧЕНИЕ

В результате работы были рассмотрены возможности и особенности используемого контроллера, созданы наборы данных для исследования, а также для дальнейшего обучения алгоритма машинного обучении. Были выделены характерные признаки, по которым можно было бы предсказать характер движения, а также реализован алгоритм машинного обучения для определения характера движения, выполняемого контроллером, и отношения его к конкретному шаблону. Алгоритм был протестирован на тестовой выборке.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Черноусько Ф. Л., Болотник Н. Н., Градецкий В. Г. Мобильные роботы: проблемы управления и оптимизации движений //XII всероссийское совещание по проблемам управления ВСПУ-2014. – 2014. – С. 67 - 78.

2 IEEE Standards Association et al. ISO/IEC/IEEE 24765: 2010 Systems and software engineering-Vocabulary. Iso/Iec/Ieee 24765: 2010 25021 //Institute of Electrical and Electronics Engineers, Inc. – 2010.

3 Peregrine Glove ST [Электронный ресурс]. URL: https://peregrineglove.com/ (дата обращения: 22.12.2019).

4 MPU9250 //Документация [Электронный ресурс]. URL: https://www.invensense.com/download-pdf/mpu-9250-datasheet/ (дата обращения: 22.12.2019).

5 Madgwick S. An efficient orientation filter for inertial and inertial/magnetic sensor arrays //Report x-io and University of Bristol (UK). – 2010. – Т. 25. – С. 113 - 118.

6 Гордеев В. Н. Кватернионы и трехмерная геометрия //Киев. – 2012.

7 Нойкин Ю. М., Махно П. В. Физические основы оптической связи //Электронное учебное пособие. Ростов н/Д. – 2011.

8 Trajectory-classifier //Репозиторий проекта [Электронный ресурс]. URL: https://github.com/ArtemZaZ/trajectory-classifier (дата обращения: 5.01.2020).

9 Sklearn //Документация [Электронный ресурс]. URL: https://scikit-learn.org/stable/ (дата обращения: 22.12.2019).