

**«Санкт-Петербургский государственный электротехнический университет  
«ЛЭТИ» им. В.И.Ульянова (Ленина)»  
(СПбГЭТУ «ЛЭТИ»)**

<b>Направление</b>	01.03.02 Прикладная математика и информатика
<b>Профиль</b>	Математическое обеспечение программно-информационных систем
<b>Факультет</b>	КТИ
<b>Кафедра</b>	МОЭВМ

*К защите допустить*

Зав. кафедрой

А.А. Лисс

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА  
БАКАЛАВРА**

**Тема: КОМПЛЕКСИРОВАНИЕ ДАННЫХ ВИЗУАЛЬНОЙ ОДОМЕТРИИ И НАВИГАЦИОННОЙ СИСТЕМЫ ДЛЯ БЕСПИЛОТНЫХ УСТРОЙСТВ**

Студент		<hr/>	М.С. Азаров
		<i>подпись</i>	
Руководитель	к. т. н., доцент	<hr/>	Д.А. Черниченко
		<i>подпись</i>	
Консультанты	к.э.н.	<hr/>	О.С. Артамонова
		<i>подпись</i>	
	к.т.н.	<hr/>	М.М. Заславский
		<i>подпись</i>	

Санкт-Петербург  
2024

## ЗАДАНИЕ

### НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю

Зав. кафедрой МОЭВМ

\_\_\_\_\_ А.А. Лисс

«      »      2024 г.

Студент                      Азаров М.С

Группа 0382

Тема работы: Комплексирование данных визуальной одометрии и навигационной системы для беспилотных устройств

Место выполнения ВКР: СПбГЭТУ «ЛЭТИ» кафедра МО ЭВМ

Исходные данные (технические требования):

# Исследовать требования к датчикам при использовании алгоритмов слияния данных визуальной одометрии и навигационной системы

## Содержание ВКР:

Введение, Обзор аналогов, Подготовительная часть, Практическая часть, Обеспечение качества разработки, Заключение.

Перечень отчетных материалов: пояснительная записка, иллюстративный материал

Дополнительные разделы: Обеспечение качества разработки, продукции, программного продукта

Дата выдачи задания

Дата представления ВКР к защите

«\_02\_»\_\_апреля\_\_\_\_2024\_ г

«\_11\_»\_\_июня\_\_\_\_2024\_ г

Студент

М.С. Азаров

Руководитель к. т. н., доцент

Д.А. Черниченко

## КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю

Зав. кафедрой МОЭВМ

\_\_\_\_\_ А.А. Лисс

«      » 2024 г.

Студент                      М.С. Азаров

Группа 0382

Тема работы: Комплексирование данных визуальной одометрии и навигационной системы для беспилотных устройств

№ п/п	Наименование работ	Срок выполнения
1	Обзор литературы по теме работы	02.04 – 03.05
2	Аналитическая часть	04.05 – 06.05
3	Разработка системы	06.05 – 18.05
4	Оформление пояснительной записки	18.05 – 26.05
7	Оформление доп. раздела	26.05 – 31.05
8	Оформление иллюстративного материала	01.06 – 04.06
9	Предзащита	05.06.2024

Студент \_\_\_\_\_ М.С. Азаров

Руководитель к. т. н., доцент \_\_\_\_\_ Д.А. Черниченко

## РЕФЕРАТ

Пояснительная записка 60 стр., 23 рис., 2 табл., 23 ист.

viSLAM, Гауссовский шум, БПЛА, локализация, IMU, визуальная одометрия, стерео-камеры.

**Объектом исследования** является алгоритмы оценки положения БПЛА на основе датчиков IMU и камер (далее viSLAM)

**Предметом исследования** является влияние параметров датчиков и получаемых данных на работу алгоритмов viSLAM

**Цель работы:** Используя различные методы моделирования разработать систему определяющую зависимость точности работы алгоритма viSLAM от различных параметров датчиков.

В данной работе рассматривается вопрос о влиянии характеристик датчиков на точность оценки положения БПЛА, полученную с помощью алгоритмов SOTA(state of the art). Для изучения этого вопроса было принято использовать алгоритм VINS-Fusion из-за его доступности и высокой точности среди других алгоритмов SOTA. В результате работы, для VINS-Fusion была разработана система оценивания точности локализации на выбранном наборе данных, с возможностью моделирования следующих параметров датчиков: разрешение камер, частота IMU, белый шум гироскопа и акселерометра.

## **ABSTRACT**

In this paper, the question of the influence of sensor characteristics on the accuracy of the UAV position estimation obtained using SOTA (state of the art) algorithms is considered. To study this issue, it was decided to use the VINS-Fusion algorithm because of its accessibility and high accuracy among other SOTA algorithms. As a result, a system for estimating localization accuracy on a selected data set was developed for VINS-Fusion, with the possibility of modeling the following sensor parameters: camera resolution, IMU frequency, white noise of the gyroscope and accelerometer.

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	9
1. ОБЗОР АНАЛОГОВ .....	14
1.1 Исследование влияния шума камеры на работу ORB-SLAM2 .....	14
1.2 Исследование влияния шума камеры глубины на модифицированный RBPF-SLAM .....	15
1.3 Исследование влияния сжатия изображения на точность ORB-SLAM3 .....	15
1.4 Исследование влияния изменения разрешения и частоты потокового видео на SVO Pro .....	16
1.5 Вывод .....	17
2. ПОДГОТОВИТЕЛЬНАЯ ЧАСТЬ .....	18
2.1 Выбор алгоритма SLAM .....	18
2.2 Выбор набора данных .....	20
2.3 Выбор исследуемых параметров датчиков .....	23
3. ПРАКТИЧЕСКАЯ ЧАСТЬ .....	24
3.1 Исследование влияния параметров камеры .....	24
3.2 Исследование влияния параметров IMU .....	37
4. ОБЕСПЕЧЕНИЕ КАЧЕСТВА РАЗРАБОТКИ .....	54
4.1 Определение потребителей .....	54
4.2 Формулировка требований .....	54
4.3 Операциональные определения .....	55
ЗАКЛЮЧЕНИЕ .....	57
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	58
ПРИЛОЖЕНИЕ А .....	61

## ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

**БПЛА** – беспилотный летательный аппарат.

**IMU** (inertial measurement units) – измерительный блок, который обычно содержит акселерометры, гироскопы (иногда еще магнитометр и барометр) и используется для определения ускорения тела и угловой скорости тела в пространстве. В данной работе подразумевается использование 6-осевого IMU (3-осевой гироскоп + 3-осевой акселерометр)

**МЭМС** - микроэлектромеханические системы, устройства, объединяющие в себе взаимосвязанные механические и электрические компоненты микронных размеров. В данной работе под МЭМС понимают IMU МЭМС.

**SLAM** (simultaneous localization and mapping) – метод построения карты в неизвестном пространстве с одновременным контролем текущего местоположения, ориентации и пройденного пути;

**viSLAM** (visual-inertial SLAM) – визуально-инерциальный SLAM. SLAM который использует данные с камер и IMU.

**Одометрия** - использование данных с датчиков движения для оценки изменения положения с течением времени

**Визуальная одометрия (VO - visual odometry)**- процесс определения положения и ориентации робота путем анализа соответствующих изображений с камер.

**Визуально инерциальная одометрия (VIO)** - процесс определения положения и ориентации робота путем анализа соответствующих изображений с камер и данных IMU датчика.

**Особые точки** – точки на изображении с уникальными свойствами, благодаря которым их можно детектировать на других изображениях и с других ракурсов.

**Loop Closure** – дословно «закрытие петли», элемент алгоритма SLAM, который уточняет локализацию устройства, используя генерируемую карту местности и повторно встречаемые *особые точки*.

**Местоположение (положение)** – определяется координатами XYZ.

**Ориентация** – определяется тремя углами или кватернионом.

**Поза объекта** – местоположение и ориентация объекта.

**Локализация** – определение *местоположения* и *ориентации* (позы) устройства.

**ATE** (Absolute Trajectory Error)- абсолютная ошибка траектории, метрика определяющая точность работы SLAM алгоритма. Тоже самое что и **APE**(Absolute Pose Error).

**RSME** (root mean square error) – это среднее квадратичное значение различий между наблюдаемыми значениями и прогнозируемыми

**ROS** - а набор программных фреймворков для разработки программного обеспечения для роботов.

**Нода** - это процесс в ROS, который выполняет вычисления. Ноды объединяются в граф и взаимодействуют друг с другом.

**Топик** - это поток данных, используемый для обмена информацией между нодами.



## **ВВЕДЕНИЕ**

### **Актуальность**

Беспилотные летательные аппараты (БПЛА) активно развиваются в последнее время. По итогам 2022 года объем российского рынка беспилотных авиационных систем и услуг с их применением составил около 50 млрд. рублей. А рост с 2018 по 2022 год рынка в среднегодовом выражении увеличился на 27 % [1]. БПЛА имеют широкое применение в различных задачах, таких как: геодезия и картографирование, спасательные операции при стихийных бедствиях, сбор пространственной информации, инспекция зданий, мониторинг дорожного движения и много других задач [2].

### **Локализация и датчики БПЛА**

Для качественного выполнения таких задач в автономном режиме, БПЛА необходимо решать проблему навигации. Данная проблема заключается в перемещении из точки А в точку В.

Для решения, данную задачу разбивают на составляющие:

1. Локализация - определение текущего местоположения устройства и его ориентацию;
2. Построение маршрута из А в В;
3. Определение действий чтобы следовать этому маршруту.

Для решения этой задачи используются GPS-трекеры и различные датчики движения: IMU , камеры , event-камеры, лидары , энкодеры, ультразвуковой дальномер и т.д.

Для локализации с помощью датчиков которые не сообщают напрямую позу объекта, используются различные алгоритмы одометрии , такие как визуальная одометрия (использует камеры) , Лидарная одометрия и т.д. .

### **Проблема GPS**

GPS широко применим в задачах локализации БПЛА, но к сожалению все еще уязвим к подделке, помехам и воздействию окружающей среды [3]. В

ситуациях, когда GPS перестает корректно функционировать, БПЛА может оценить свою позу с помощью относительной локализации, получив позу относительно последней глобальной позы, полученной с GPS. Затем, учитывая эти данные вычислить текущую глобальную позу.

Данная работа концентрирует свое внимание на проблеме относительной локализации в условиях отсутствия GPS.

## SLAM

Для решения задачи относительной локализации широкое применение получили алгоритмы SLAM, которые оценивают позу робота и одновременно строят карту местности на основе полученных данных с различных датчиков.

В последнее время алгоритмы SLAM активно разрабатываются и изучаются в научных кругах.

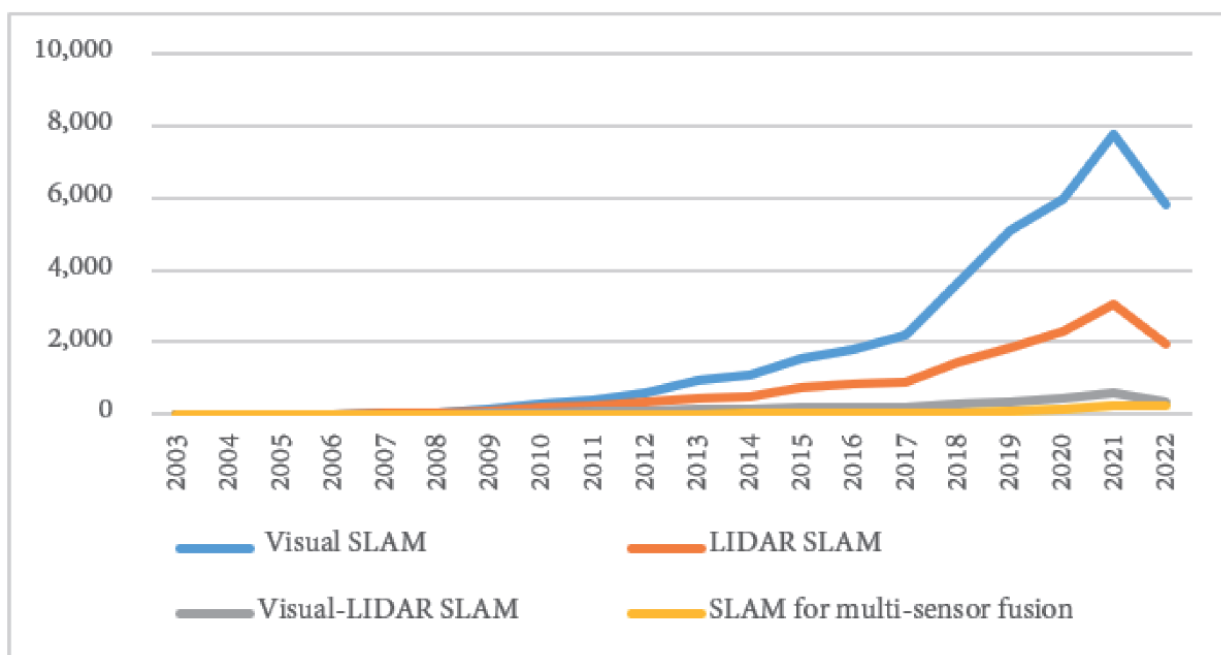


Рисунок 1 – Кол-во цитирований статей связанных с методами SLAM с 2003 по 2022 год в Web of Science

Как видно из рис. 1, количество цитируемых статей, связанных с SLAM методами значительно увеличилось и имеет внушительный масштаб. Спад в

2022 связан с тем что данная статистика собралась в тот же год, и говорит о том что не все статьи еще были опубликованы. [4]

### **Проблема выбора датчиков**

Учитывая сказанное, алгоритмы SLAM являются активно исследуемой областью и имеет широкое применение в практических целях так как используются в навигации БПЛА. В связи с этим перед промышленностью встает вопрос, не только в выборе нужного алгоритма SLAM, но и выборе датчиков, для достижения необходимой точности локализации для успешного выполнения задач. С одной стороны, слишком дешёвые датчики вызовут уменьшение точности локализации устройства, в связи с чем выполнение задач будет невозможно. С другой слишком дорогие повлекут избыточные финансовые траты, что не выгодно для бизнеса.

Для помощи в этом вопросе данная работа ставит следующие задачи и цель.

**Цель работы:** Используя различные методы моделирования разработать систему определяющую зависимость точности работы алгоритма viSLAM от различных параметров датчиков.

**Объектом исследования** является алгоритмы оценки положения БПЛА на основе датчиков IMU и камер (далее viSLAM)

**Предметом исследования** является влияние параметров датчиков и получаемых данных на работу алгоритмов viSLAM

### **Задачи**

В данной работе предпринята попытка помочь с поиском компромисса между качеством датчиков и точностью работы алгоритма. Для достижения цели необходимо решить следующие задачи:

- Подготовительная часть
  - Выбрать алгоритм SLAM

- Выбрать набор данных
- Выбрать исследуемые параметры датчиков
- Практическая часть
  - Разработать систему позволяющую, моделировать различные данные в соответствии с выбранными параметрам датчиков на основе оригинального набор данных.
  - Продемонстрировать применение полученных зависимостей при выборе характеристик датчиков.

Данная работа фокусируется на исследовании факторов влияющих на точность алгоритма SLAM. Вопрос связанный с вычислительными нагрузкой в данной работе не рассматривается. В дальнейшем, алгоритму будет предоставлено достаточно вычислительной мощности, чтобы производить максимально точную оценку.

Метод исследования и результаты моделирования, описанные в этой статье, могут быть в дальнейшем использованы при проектировании мобильной роботизированной навигационной системы.

## **Структура**

В текущем разделе «ВВЕДЕНИЕ» рассказывается об актуальности проблемы рассматриваемой в данной работе и проводится небольшое поверхностное введение в научную область для лучшего дальнейшего понимания содержания работы. Также ставятся задачи работы, которые в дальнейшем будут выполнены в ходе исследования.

В разделе «1. ОБЗОР АНАЛОГОВ» рассматриваются схожие по теме исследования работы, для выяснения проработанность рассматриваемой области и оценки новизны данной работы.

В разделе «2. ПОДГОТОВИТЕЛЬНАЯ ЧАСТЬ» описывается решение задач связанных с выбором SLAM алгоритма, базы данных и параметрами датчиков и аргументируется почему было принято именно такое решение.

В разделе «3. ПРАКТИЧЕСКАЯ ЧАСТЬ» в соответствии с названием производится разработка системы для моделирование данных для SLAM алгоритма на предоставленных оригинальных данных .

В разделе «4.ОБЕСПЕЧЕНИЕ КАЧЕСТВА РАЗРАБОТКИ » демонстрируется способ применения разработанной системы для оценки качества алгоритма viSLAM .

Наконец в разделе «5. ЗАКЛЮЧЕНИЕ» формулируются заключительные замечания и рекомендации относительно дальнейшего развития.

## 1. ОБЗОР АНАЛОГОВ

Несмотря на активное развитие SLAM в научной литературе, подавляющее большинство статей либо разрабатывают новую систему SLAM, либо создают набор данных для конкретной проблемы (EuRoC, KITTI, TUM-VI и др. [5, 6, 7]), либо фреймворки для удобного создания алгоритмов SLAM (GTSAM, OpenVINS, [8, 9]) или их удобной оценки (EVO[10]). Об этом говорит существования множество даже viSLAM-ов с *открытым* исходным кодом (ORB-SLAM3, SVO, VINS, Okvis, Rovio и несколько других [11, 12, 13, 14, 15]) и различные их вариации и улучшения, созданные за последние 10 лет.

Возможно, в связи с тем, что область бурно развивается, и новые алгоритмы регулярно появляются, влиянию датчиков, на которых запускаются данные алгоритмы, уделяется меньшее время. В данном разделе рассматриваться те немногие статьи, которые все-таки фокусируются на этой проблеме.

### 1.1 Исследование влияния шума камеры на работу ORB-SLAM2

В работе [16], изучается влияние различных уровней шумов камеры на работу visual-SLAM. В качестве образца visual-SLAM берётся ORB-SLAM2. Также авторы статьи представляют синтетически сгенерированный набор данных с различными уровнями шума изображения. В ходе создания набора моделируются три вида шума: независимый от освещенности, зависящий от освещенности и смесь этих двух. Также авторы предлагают сравнить работу ORB-SLAM2 на зашумленных данных и на шумоподавленных данных используя быструю и гибкую сверточную нейронную сеть FFDNet[17].

В ходе экспериментов были определены граничные значения шума до которых АТЕ оставалась стабильной. Что касается шумоподавленных данных, АТЕ почти во всех экспериментах оставалась стабильной. Другими словами, последовательности с шумоподавлением более надежны и показывали большую точность, чем исходные зашумленные последовательности. Так же было установлено, что для чистой последовательности (без добавления каких

либо шумов) среднеквадратичная ошибка (RMSE) АТЕ после шумоподавления снизилась на 16,75%;

## **1.2 Исследование влияния шума камеры глубины на модифицированный RBPF-SLAM**

Авторы статьи [18] изучают влияния случайных шумов, создаваемых камерой глубины съемки ToF, на точность оценки позы мобильного робота на основе усовершенствованного алгоритма RBPF-SLAM. В ходе работы был усовершенствован алгоритм RBPF-SLAM и создана Гауссова модель шумов глубинной камеры, используя которую были сгенерированы зашумленные данные.

В результате были построены графики RMSE траекторий для различных уровней шума. Случайные помехи, создаваемые глубинной камерой ToF, влияют на точность позиционирования мобильного робота. Чем больше шум, тем больше среднеквадратичная погрешность траектории робота. Таким образом когда параметр шума был равен  $\mu_x = \mu_y = 0$ ,  $\sigma_x = \sigma_y = 0.09$  максимальная среднеквадратичная погрешность траектории робота достигла 0.075м.

## **1.3 Исследование влияния сжатия изображения на точность ORB-SLAM3**

В этой работе [19] исследуется влияние сжатия изображений на точность локализации ORB-SLAM3. Для увеличения пропускной способности, данные полученные с БПЛА предлагается сжимать и затем отправлять на внешний сервер, где будет выполняться SLAM. В связи с этим также предлагаются методы по улучшению обработки ORB-SLAM3 сжатых видео и исследуется влияние сжатия на точность модифицированного алгоритма и оригинального.

В результате построены box-диаграммы APE для оригинального ORB-SLAM3 и 4х модификаций на трех сжатых последовательностях, а также для

каждого случая были вычислены основные параметры распределения APE такие как: среднее, медиана, минимум/максимум, дисперсия. Опираясь на полученные данные, был сделан вывод, что предложенные модификации повышают устойчивость ORB-SLAM3 к сжатию изображений, обеспечивая более высокую степень сжатия при сохранении желаемой точности локализации.

#### **1.4 Исследование влияния изменения разрешения и частоты потокового видео на SVO Pro**

В этой статье [20] исследуются характеристики визуальной одометрии при различных сочетаниях разрешения изображения и частоты получения изображений. За пример визуальной одометрии берётся алгоритм визуально-инерциальной одометрии SVO Pro Open. В ходе исследования тестируют работу алгоритма на трех различных разрешениях:  $848 \times 800$  пикселей (исходное),  $636 \times 600$  пикселей и  $424 \times 400$  пикселей; и на 4 частотах: 30 (исходная), 25, 20 и 15 Гц. В статье анализируются точность SVO Pro и загрузку ЦП при работе на различных сгенерированных последовательностях.

В результате было выяснено, что для проанализированного промежуточного разрешения ( $636 \times 600$  пикселей) может быть достигнут оптимальный компромисс с точки зрения точности локализации, загрузки ЦП и надежности системы. Исследование с изменением частоты показало, что на высоких частотах (25 и 30 Гц) получаются лучшие результаты с точки зрения локализации. Кроме того, из вычислительного анализа было выяснено, что частота 25 Гц обеспечивает значительную экономию вычислительных ресурсов по сравнению с 30 Гц для этого промежуточного разрешения, хотя и с несколько более высокими ошибками перевода. Таким образом, как утверждают авторы, пользователь может найти подходящий компромисс между точностью и ресурсами ЦП в зависимости от доступных вычислительных возможностей.



## **1.5 Вывод**

Как можно наблюдать, проблема влияния различных характеристик датчиков на разные алгоритмы SLAM мало рассмотрена в литературе и имеет множество пробелов. Большинство рассмотренных публикаций в основном фокусируется на модификации существующего алгоритма и при экспериментальном исследовании предложенного решения, заодно предоставляют информацию о работе оригинального алгоритма.

Данная работа будет целенаправленно концентрироваться не на исследовании новой модификации, а на свойствах оригинальной модели. Результаты этого исследования могут в дальнейшем применяться в промышленности, при разработке навигационных систем. В ходе работы, будет учтено какие алгоритмы уже были частично исследованы, а какие остались без внимания и учтет это при выборе исследуемого SLAM.

Забегая вперед, для исследований будет выбран алгоритм VINS-Fusion. Как можем видеть, данный алгоритм еще не исследовался с данного ракурса. Другими словами, данная работа действительно представляет научную новизну. Почему был выбран именно VINS-Fusion будет подробно сказано в разделе «3. ПОДГОТОВИТЕЛЬНАЯ ЧАСТЬ».

## **2. ПОДГОТОВИТЕЛЬНАЯ ЧАСТЬ**

В данном разделе решаются задачи связанные с выбором алгоритма SLAM, точность которого в дальнейшем будет в различных ситуациях; с выбором набора данных, который ляжет в основу генерируемых данных; и с выбором параметров датчиков. В следующем разделе будет исследоваться, как эти параметры влияют на точность выбранного алгоритма.

### **2.1 Выбор алгоритма SLAM**

SLAM может опираться на один датчик , например : vSLAM – visual-SLAM который использует визуальную одометрию для оценки позы, LIDAR SLAM – использует лидар. Либо на множество датчиков : viSLAM – visual-inertial SLAM(IMU+camera), RTAB-MAP (лидар + камера).

Обычно применяется несколько датчиков, для достижения большей точности и компенсирования недостатков каждого.

Так как данная работа нацелена исследованию влиянию параметров датчиков в области локализацию БПЛА, БПЛА имеют характерные особенности связанные с компактными размерами и вычислительной ограниченностью. Поэтому использование лидаров неуместно из-за громоздкости и большой вычислительной нагрузки. Лидары зачастую используются в наземных автопилотах, но на данный момент не применимы в большинстве БПЛА(за исключением очень огромных).

В условиях наружного освещения или при съемке больших сцен использование камер глубины также затруднено.

Благодаря последним достижениям в области проектирования и производства аппаратного обеспечения, появились недорогие и легкие датчики IMU - МЭМС. Благодаря компактности и дешевизне МЭМС датчики стали широко применяться в задачах локализации БПЛА . К сожалению эти датчики имеют большую погрешность и дрейф нуля, в связи с чем зачастую их используют в связке с другими датчиками для компенсирования ошибки.

Таковыми датчиками может выступить камера. Камера, обладает преимуществами небольшого размера, малого веса, малой мощности и низкой стоимости, а также может предоставлять обширную информацию и проста в использовании. Методы визуального SLAM с использованием камер значительно повысили точность, надежность и эффективность и в последние годы приобретают все большую популярность [22]. На устройстве может быть одна камера – моно или несколько стерео. Для получения большей информации о 3D окружении а также для более точной триангуляции, желательно использовать stereo.

Таким образом, эффективным комплексом датчиков для БПЛА является визуально-инерциальная система, состоящая из недорогого инерциального измерительного блока (IMU) и стерео-камер, которая сочетает в себе преимущества визуальной одометрии (VO) и IMU.

Опираясь на результаты статьи [21], в которой рассматривается и сравниваются множество широко используемых открытых алгоритмов моно и стерео VO/VIO. Результат сравнения можно увидеть на рисунке 2. Под наши требования подходят VINS-Fusion (VINS-Fusion-imu), Stereo-MSCKF(S-MSCKF), Kimera. Было принято взять **VINS-Fusion** так как он показал лучшую точность среди прочих. (VINS имел наименьшую среднюю RSME по всем видео)

	Xavier								
	VINS-Mono	ALVIO	ROVIO	Kimera	VINS-Fusion	VINS-Fusion-gpu	VINS-Fusion-imu	ORB-SLAM2	S-MSCKF
cir-n	0.12	0.12	×	0.08	<b>0.07</b>	0.09	0.08	0.08	0.11
cir-f	0.14	0.12	0.80	<b>0.08</b>	0.16	0.13	0.13	0.12	0.23
cir-h	0.41	0.49	2.11	0.26	<b>0.06</b>	0.11	0.07	0.15	0.20
inf-n	0.24	0.12	1.19	0.09	<b>0.07</b>	0.09	<b>0.07</b>	<b>0.07</b>	0.09
inf-f	0.10	0.09	0.44	0.13	0.07	<b>0.05</b>	0.07	0.07	0.12
inf-h	0.57	0.48	×	1.09	<b>0.09</b>	0.14	0.12	<b>0.09</b>	0.87
squ-n	0.11	<b>0.10</b>	0.47	0.13	0.17	<b>0.10</b>	0.15	0.12	0.15
squ-f	0.12	0.10	0.56	0.14	<b>0.08</b>	0.11	0.10	0.14	0.17
squ-h	0.30	0.36	×	1.50	0.18	<b>0.15</b>	0.18	0.17	0.50
rot-n	×	0.81	×	0.18	0.11	0.12	0.11	0.16	<b>0.07</b>
rot-f	0.89	0.72	×	0.90	0.26	0.11	<b>0.07</b>	0.18	0.19

Рисунок 2 – Результат сравнению алгоритмов VO\VIO в статье[21]. По горизонтали располагаются рассматриваемые алгоритмы, по вертикали названия видео данных, на которых тестировались алгоритмы. В ячейках указана RMSE ATE для каждой последовательности.

Также стоит обратить внимания что из раздела «2. ОБЗОР АНАЛОГОВ» видно, что до сих пор, исследования влияние качества датчиков на VINS-Fusion не проводились.

## 2.2 Выбор набора данных

Благодаря активному развитию SLAM алгоритмов, в настоящее время существует множество готовых наборов данных различных датчиков для задач SLAM. Основные из них представлены в рисунке 3.

Dataset	Sensor	Environment	Availability
KITTI	RGB-D+LIDAR+GPS+IMU	Outdoor	[75]
Oxford	RGB-D+LIDAR+GPS+IMU	Outdoor	[76]
ASL Kinect	RGB-D	Indoor	[77]
ASL RGB-D	RGB-D+LIDAR	Indoor	[78]
TUM RGB-D	RGB-D	Indoor	[72]
ICL-NUIM	RGB-D	Indoor	[79]
VaFRIC	RGB-D	Indoor	[80]
EuRoC	Binocular+IMU	Indoor	[81]
TUM VI	Binocular+IMU	Indoor/Outdoor	[74]
TUM monoVO	Monocular	Indoor/Outdoor	[73]

Рисунок 3 – Широко распространенные наборы данных для задач SLAM алгоритмов. Изображение взято из [4].

В связи с тем, что был выбран VINS-Fusion который является стерео viSLAM, из наборов данных представленных в рисунке 3 нам подходят только EuRoC и TUM VI. Было принято решения взять набор **EuRoC**, для моделирования различных параметров датчиков. Это связано с тем, что по личному опыту он более распространен в литературе, связанной с viSLAM, чем TUM VI.

EuRoC содержит 11 видео последовательностей, снятых в трех помещениях(V1, V2, МН). Характерные особенности каждой видеопоследовательности показаны на рисунке 4.

Name	Length / duration	Avg. vel. / angular vel.	Note
MH_01_easy	80.6m 182s	$0.44\text{ms}^{-1}$ $0.22\text{rad s}^{-1}$	Good texture, bright scene
MH_02_easy	73.5m 150s	$0.49\text{ms}^{-1}$ $0.21\text{rad s}^{-1}$	Good texture, bright scene
MH_03_medium	130.9m 132s	$0.99\text{ms}^{-1}$ $0.29\text{rad s}^{-1}$	Fast motion, bright scene
MH_04_difficult	91.7m 99s	$0.93\text{ms}^{-1}$ $0.24\text{rad s}^{-1}$	Fast motion, dark scene
MH_05_difficult	97.6m 111s	$0.88\text{ms}^{-1}$ $0.21\text{rad s}^{-1}$	Fast motion, dark scene
V1_01_easy	58.6m 144s	$0.41\text{ms}^{-1}$ $0.28\text{rad s}^{-1}$	Slow motion, bright scene
V1_02_medium	75.9m 83.5s	$0.91\text{ms}^{-1}$ $0.56\text{rad s}^{-1}$	Fast motion, bright scene
V1_03_difficult	79.0m 105s	$0.75\text{ms}^{-1}$ $0.62\text{rad s}^{-1}$	Fast motion, motion blur
V2_01_easy	36.5m 112s	$0.33\text{ms}^{-1}$ $0.28\text{rad s}^{-1}$	slow motion, bright scene
V2_02_medium	83.2m 115s	$0.72\text{ms}^{-1}$ $0.59\text{rad s}^{-1}$	fast motion, bright scene
V2_03_difficult	86.1m 115s	$0.75\text{ms}^{-1}$ $0.66\text{rad s}^{-1}$	fast motion, motion blur

Рисунок 4 – Краткие характеристики видеопоследовательностей EuRoC. Изображение взято из [5].

Также в наборе для каждого видео представлена истинная траектория БПЛА хранящаяся в .csv формате. Данная траектория получена путем использования системы захвата движения **Vicon** или лазерного трекера **Leica MS50**. Истинная траектория необходима для вычисления ошибки траектории построенной алгоритмом SLAM.

Также стоит упомянуть что разработчиками OpenVINS[9] было обнаружено, что истинная траектория V1\_01\_easy была неточно оценена. В итоге они предоставили более корректный вариант истинной траектории. Подроб-

нее [23]. В связи с этим в дальнейшем в ходе работы будет использоваться обновленный вариант истинной траектории для V1\_01\_easy.

### **2.3 Выбор исследуемых параметров датчиков**

VINS-Fusion использует два типа датчиков: IMU и стерео камеры. В связи с этим, будут моделироваться следующие параметры для этих датчиков.

Для стереокамер:

- Разрешение

Для IMU:

- Частота
- Белый шум гироскопа
- Белый шум акселерометра

### **3. ПРАКТИЧЕСКАЯ ЧАСТЬ**

В данном разделе производится разработка системы моделирование данных соответствующих различным параметрам датчиков камер и IMU. Исследуется зависимость между параметрами датчиков и точностью VINS-Fusion. На основе проведенных экспериментов и полученных результатов делаются соответствующие выводы.

#### **3.1 Исследование влияния параметров камеры**

##### **3.1.1 Разрешение**

###### **Моделирование**

Для моделирования различного разрешение потокового видео было решено смоделировать следующие разрешения:

- 480 x 752 – оригинальное
- 432 x 677
- 389 x 609
- 350 x 548
- 315 x 493

Данные значения разрешений были получены уменьшение предыдущего разрешение на 0.9.(Пример:  $480 * 0.9 = 432$ ,  $752 * 0.9 = 676,8$  и т.д.)

Для примера данных было взято три последовательности из набора данных EuRoC (в скобках указаны сокращения принятые для этого подраздела):

- V1\_01\_easy (V1)
- MH\_03\_medium (MH)
- V2\_03\_difficult (V2)

Данный выбор был обусловлен необходимостью максимально покрыть набор данных, поэтому были взяты последовательности из всех трех комнат, с различными уровнями сложности.



В результате, необходимо разработать программу, которая смоделирует 12 видео данных для последующего исследования точности VINS-Fusion.

В наборе данных EuRoC данные хранятся в формате bag . Bag это формат файла ROS в котором сохраняются сообщения публикуемые в топиках различными нодами, генерируемый специальными инструментами ROS. Bag файл имеет расширение .bag . В EuRoC в bag файле хранятся данные получаемые из датчиков IMU, камер. Для каждой видеопоследовательности существует свой bag файл.

Поэтому для моделирования, необходимо создать новый bag файл для каждой последовательности изменив в нем только разрешение изображений, получаемых с камер.

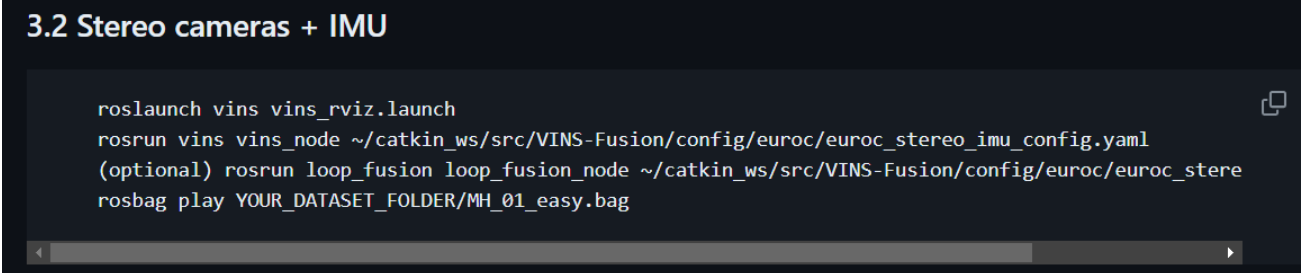
Для этого была написана программа *genbag\_resolution.py* с использование языка Python и следующих библиотек: rosbag, CvBridge, openCV.

Данная программа создает новые bag файл с различными разрешениями видео внутри, в следующие этапы:

1. С помощью функций библиотеки rosbag, происходит последовательное считывание всех сообщений хранимых в bag файле
2. Если сообщение не принадлежит камерам, то данное сообщение без изменений сохраняется в новых bag файл. В противном случае сообщение передается в функцию *shrink\_image()* и результат сохраняется в новых bag файл.
3. В функции *shrink\_image()* :
  - a. происходит конвертация изображение хранимого в сообщении ROS в формат openCV используя библиотеку CvBridge.
  - b. Изменение разрешения инструментами библиотеки openCV.
  - c. Обратная конвертация измененного изображения с помощью CvBridge.



В соответствии с представленной информацией в GitHub репозитории VINS-Fusion , чтобы запустить алгоритм на данных EuRoC, необходимо выполнить 4 команды в различных терминалах. Команды представлены на рисунке 7.



```
3.2 Stereo cameras + IMU

roslaunch vins vins_rviz.launch
roslaunch vins vins_node ~/catkin_ws/src/VINS-Fusion/config/euroc/euroc_stereo_imu_config.yaml
(optional) roslaunch loop_fusion loop_fusion_node ~/catkin_ws/src/VINS-Fusion/config/euroc/euroc_stereo_imu_config.yaml
rosbag play YOUR_DATASET_FOLDER/MH_01_easy.bag
```

Рисунок 7 – Команды необходимые для запуска VINS\_Fusion

Команда *roslaunch vins vins\_rviz.launch* запускает ros master и визуализатор rviz в котором отображается построение траектории в ходе работы VINS.

Команда *roslaunch vins vins\_node <config.yaml>* запускает основные ноды связанные с визуальной одометрией и построением траектории на основе VO и IMU. В *config.yaml* описываются все переменные алгоритма и параметры используемых датчиков.

Команда *roslaunch loop\_fusion loop\_fusion\_node <config.yaml>* запускает ноды связанные выполнением Loop Closure. В противном случае алгоритм не будет выполнять уточнение траектории используя метод «закрывания петли». *config.yaml* – совпадает с конфигурационным файлом из предыдущей команды.

*rosbag play ./MH\_01\_easy.bag* – воспроизводит сохранённые данные в bag, имитируя получение данных алгоритмом в реальном времени.

Для запуска VINS использует не только данные из конфигурационного файла *euroc\_stereo\_imu\_config.yaml* но и из зависимых файлов лежащих в папке /config/euroc/, список файлов показан на рисунке 8.

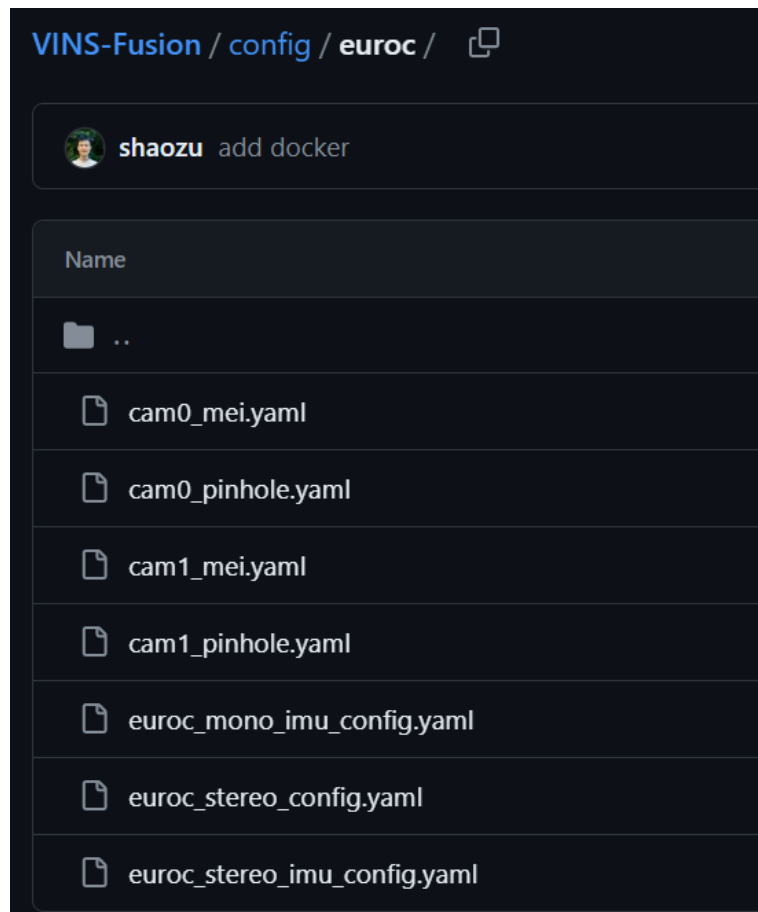


Рисунок 8 – Список конфигурационных файлов для запуска VINS-Fusion на наборе данных EuRoC.

Так как в данных конфигурационных файлах содержит информацию о параметрах камеры, то для корректного выполнения VINS на различных разрешениях были созданы соответствующие папки с измененными конфигурационными файлами.

Для получения траектории БПЛА в файловом формате из запущенного VINS, была создана программа *sub.py*. Данная программа создает ноду-подписчика и подписывается на топик, в котором публикуется траектория для отображения в rviz. (Название данного топика: «/loop\_fusion/pose\_graph\_path»). Полученные данные сохраняются в файле в формате TUM по пути переданному в программу.

Формат траектории TUM выглядит следующим образом, как показано на рисунке 9. Каждая строка содержит 8 записей, содержащих временную метку (в секундах), положение и ориентацию (в виде кватерниона), причем каждое значение разделено пробелом. Символ # обозначает комментарий.

```
# timestamp(s) tx ty tz qx qy qz qw
1403636580.83856 4.688319 -1.786938 0.783338 -0.153029 -0.827383 -0.082152 0.534108
1403636580.84356 4.688177 -1.786770 0.787350 -0.152990 -0.826976 -0.082863 0.534640
1403636580.84856 4.688028 -1.786598 0.791382 -0.152945 -0.826562 -0.083605 0.535178
1403636580.85356 4.687878 -1.786421 0.795429 -0.152884 -0.826146 -0.084391 0.535715
1403636580.85856 4.687727 -1.786240 0.799484 -0.152821 -0.825731 -0.085213 0.536244
```

Рисунок 9 – Пример того как храниться траектория в формате TUM.

В связи с тем, что VINS необходимо запустить 15 раз были предприняты методы для автоматизации процесса:

1. Для этого первая команда *roslaunch vins vins\_rviz.launch* была заменена командой *roscore*, для того чтобы запустить только *ros master*, без запуска графического визуализатора *rviz*.
2. Для того чтобы была возможность запустить все команды в одном окне консоли, все команды запускаться в фоновом режиме используя в конце символ амперсанда (&)
3. Чтобы работая в фоновом режиме команды не выводили в консоль промежуточные результаты был перенаправлен вывод используя потоки *stdout* и *stderr*. В результате команда должна выглядеть следующим образом: *command > /dev/null 2>&1 &*
4. После выполнения работы все процессы связанные с *ros* уничтожаются, для корректного последующего запуска алгоритма.
5. Написан скрипт *create\_VINS\_tum.sh* который реализует данные подходы и автоматизирует последовательный запуск алгоритма VINS а также запуск программы *sub.py* на всех 15 видео.

В результате работы скрипта *create\_VINS\_tum.sh* было создано 15 файлов .tum описывающих траектории полученные при работе VINS-Fusion на последовательностях с различными разрешениями.

### Анализ полученных результатов

Для анализа и сравнения полученных данных использовался такой инструмент как EVO [10].

EVO имеет следующие консольные команды:

- *evo\_ape* – вычисляет APE(absolute pose error), то же самое что и ATE(absolute trajectory error) между траекторией вычисленной SLAM и истинной траекторией
- *evo\_rpe* - вместо прямого сравнения абсолютных поз, относительная погрешность поз сравнивает движения ("дельта поз")
- *evo\_traj* – инструмент для анализа, построения или экспорта одного или нескольких траектории
- *evo\_res* – инструмент для сравнения одного или нескольких файлов результатов из *evo\_ape* или *evo\_rpe*

В наборе EuRoC истинная траектория храниться в формате .csv как было сказано в подразделе «3.2 Выбор набора данных». Траектория построенная VINS храниться в формате .tum . Для согласования форматов был применен инструмент *evo\_traj*, в результате чего все истинные траектории были конвертированы из формата .csv в формат .tum

Для описания точности работы VINS-Fusion будет использоваться метрика APE. APE для времени  $t$  вычисляется следующим образом:

$$E_t = P_{gt,t}^{-1} P_{est,t}$$

Где  $P_{gt,t}$  – поза истинной траектории в момент  $t$  ,  $P_{est,t}$  - поза вычисленной SLAM траектории в момент  $t$ .

После получения APE для каждого момента времени, можно вычислить различные статистики численно описывающие конкретную вычисленную траекторию алгоритмом SLAM.

Примером такой статистики может выступить RMSE:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N APE_i^2}$$

Для вычисления APE между истинной траекторией и полученной применяется команда *evo\_ape*. Для нее необходимо указать формат сравниваемых данных, нахождение файлов истинной и вычисленной траектории и способ геометрического сопоставления.

Последнее является очень важным, под сопоставлением понимается какие преобразования можно использовать при наложения вычисленной траектории на истинную. Пример различных сопоставлений показан на рисунке 10.

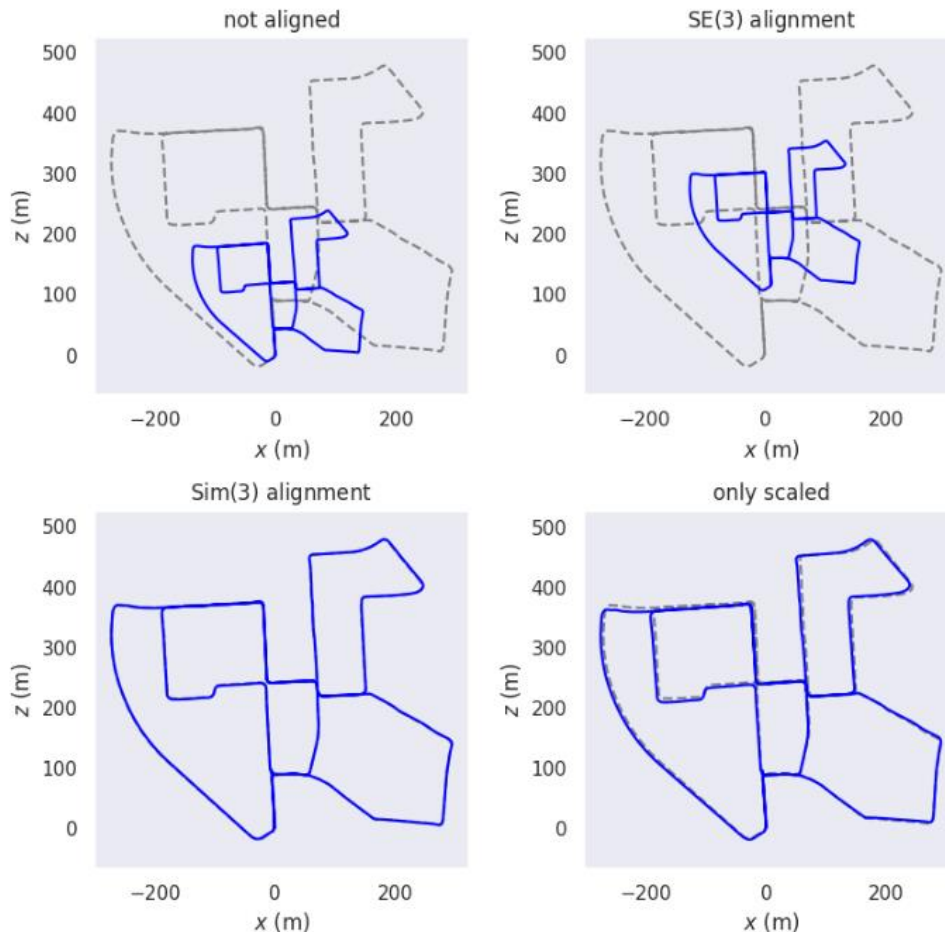


Рисунок 10 – Различные способы сопоставления траекторий. Сверху-вниз , слева-направо: без сопоставления, вращение и перемещение, вращение перемещение и масштаб , только масштаб.

Может казаться, что лучше всего сравнивать траектории без сопоставления, но тогда для оригинального набора данных мы получаем следующий результат , рисунок 11 . Можем видеть, что ошибка достаточно большая, хотя очевидно, что фигуры похожи с точностью до сдвига. На рисунке 12 показаны те же траектории, но с применением сопоставления и как видно траектории почти совпадают.

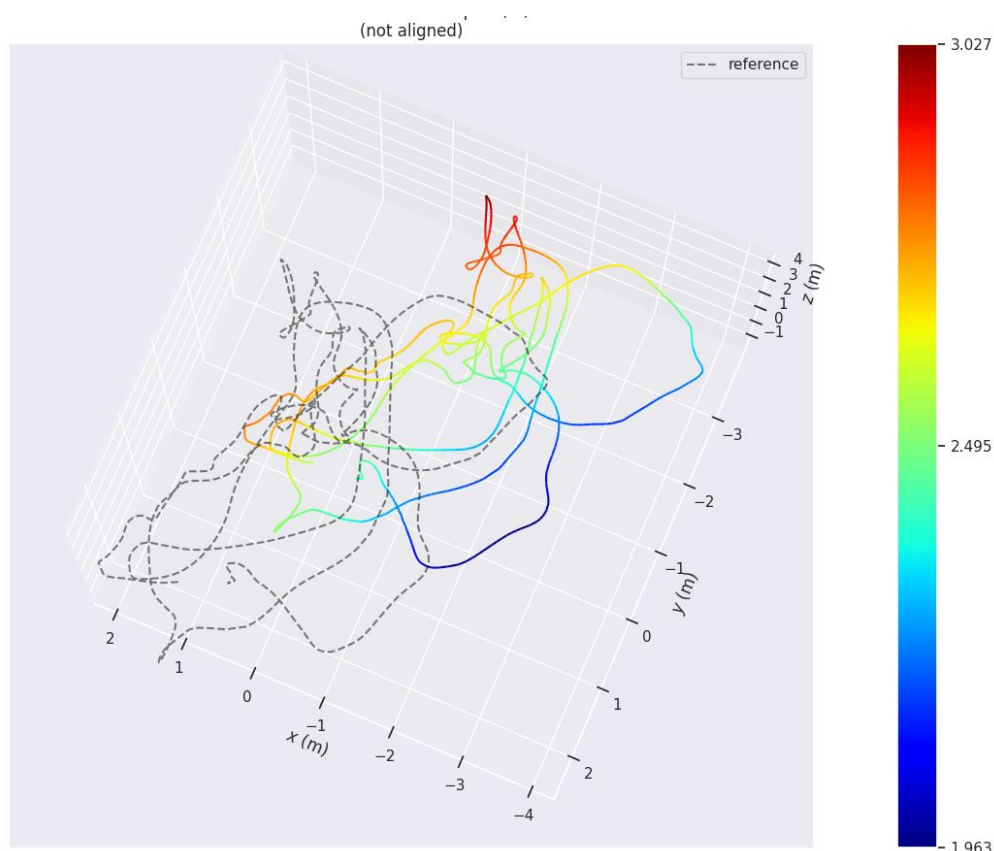


Рисунок 11 – Результат сравнения истинной траекторий и вычисленной VINS без сопоставления. Пунктиром обозначена истинная траектория, цветной линией траектория полученная из VINS. Последовательность V1\_01\_easy.



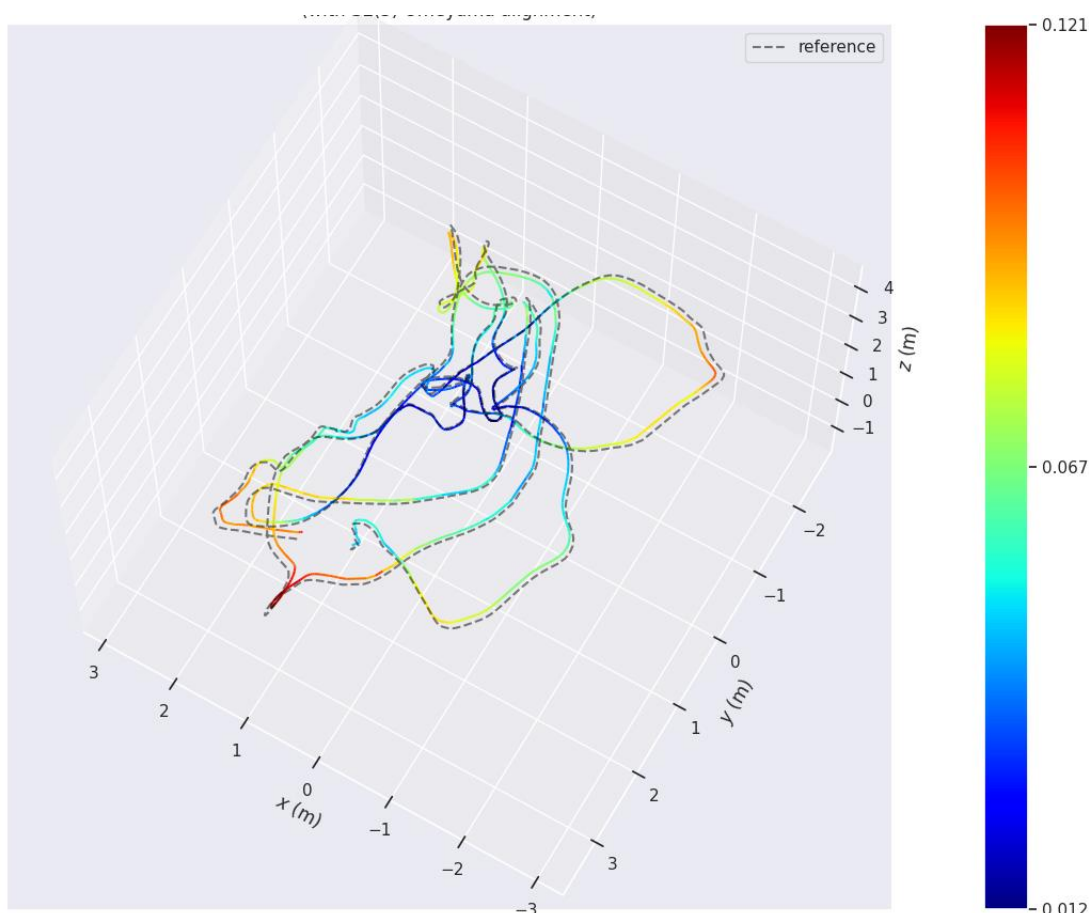


Рисунок 12 – Результат сравнения истинной траекторий и вычисленной VINS с сопоставлением. Пунктиром обозначена истинная траектория, цветной линией траектория полученная из VINS. Последовательность V1\_01\_easy.

При использовании сдвига возникает другая проблема, если для каждой пары траекторий производить различное сопоставление, итоговые результаты будет невозможно согласовать между собой. Для этого команда *evo\_ape* будет использоваться с флагом *--align\_origin*, который отвечает за простое сопоставление по исходной точке. В данном случае, сопоставление у всех будет одинаковое и можно будет удобно сравнивать дрейф оценки траектории.

В результате, итоговая команда для вычисления APE выглядит следующим образом:

```
evo_ape tum <путь до ист. тр.> <путь до выч. тр.> -p --align_origin --
save_results results.zip
```

Результат которой сохраняется в файл *results.zip*

После выполнения сравнения истинных траекторий с вычисленными VINS, с использованием инструмента *evo\_ape*, было получено множество файлов *results.zip*. Для обработки и сравнения этих файлов используется инструмент EVO *evo\_res*, который соединяет полученные APE из разных последовательностей в один график.

Таким образом был получен график APE для всех разрешений для последовательности V1\_01\_easy, см. рисунок 13.

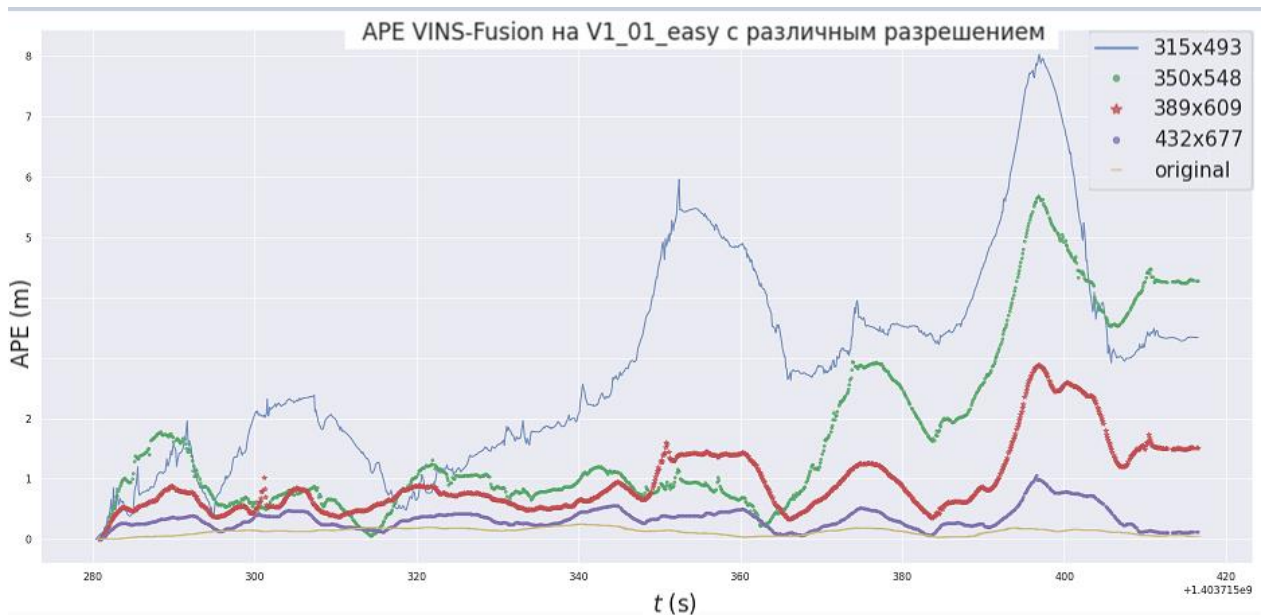


Рисунок 13 – График APE VINS-Fusion для всех разрешений для последовательности V1\_01\_easy.

Также *evo\_res* предоставляет самые популярные статистики APE для каждой последовательности и сохраняет их в csv файл. Для лучшего отображения и восприятия, была написана программа *csv\_to\_latex.py* для перевода таблицы из csv в latex стиль. С помощью этой программы в таблицу были собраны все полученные статистики для всех 15 последовательностей, см. рисунок 14.

	rmse	mean	median	std	min	max
V1 original	0.14	0.13	0.145	0.058	0.0	0.25
V1 432x677	0.38	0.34	0.32	0.17	0.0	1.06
V1 389x609	1.12	0.97	0.77	0.57	0.0	2.88
V1 350x548	2.21	1.70	1.05	1.41	0.0	5.68
V1 315x493	3.41	2.92	2.78	1.76	0.0	8.03
MH original	0.44	0.39	0.41	0.20	0.0	0.96
MH 432x677	2.58	1.75	1.11	1.89	0.0	11.34
MH 389x609	2.08	1.80	1.96	1.03	0.0	3.75
MH 350x548	21.76	12.87	6.64	17.55	0.0	109.86
MH 315x493	x	x	x	x	x	x
V2 original	0.21	0.19	0.16	0.08	0.0	0.43
V2 432x677	12.7	11.39	10.42	5.64	0.0	20.18
V2 389x609	10.59	8.50	9.01	6.31	0.0	21.17
V2 350x548	x	x	x	x	x	x
V2 315x493	x	x	x	x	x	x

Рисунок 14 – Таблица статистик APE VINS-Fusion для каждой рассматриваемой последовательности данных в этом подразделе. x – обозначено, что алгоритм потерял траекторию и ошибка превысила 100м.(V1 - V1\_01\_easy, V2 - V2\_03\_difficult, MH- MH\_03\_medddium)

В ходе разработки, был создана система, позволяющий исследовать точность VINS-Fusion на различных разрешениях камеры. В ходе практического использования были получены:

1. График APE VINS-Fusion для различных разрешений V1\_01\_easy, рисунок 13.
2. Таблица статистик APE VINS-Fusion для каждой рассматриваемой последовательности V1\_01\_easy, V2\_03\_difficult, MH\_03\_medddium, рисунок 14.

## Интерпретация результатов

Как и ожидалось из рисунка 13 видно, что чем меньше разрешение, тем выше получаемая APE, что говорит об ухудшении точности работы алгоритма. Колебания на графике скорее всего связаны с сложными участками траектории такие как повороты.

Из таблицы из рис.14 можно сделать вывод что чем сложнее данные тем лучше нужны датчики, об этом говорит то что на простой последовательности V1\_01\_easy VINS справлялся на всех разрешениях несмотря на растущую ошибку. На сложной V2\_03\_difficult VINS уже не смог корректно строить траекторию на разрешении 350 x 548 и 315 x 493.

Если говорить о дальнейшем применении полученных данных, то можно разобрать пример выбора разрешения датчиков таким, чтобы RMSE APE была не больше 1 м:

- для задач с медленным движением БПЛА и яркой сценой , другими словами условиями близкими к V1\_01\_easy если средняя ошибка должна быть меньше метра тогда лучше использовать камеры с разрешением >389x609 и тд. С другой стороны если средняя ошибка 3 метра а максимальная 8метров на протяжении 80 метров допустима , то можно использовать камеры и с разрешением >315 x 493.
- для задач с быстрым движением как в MH\_03\_medium использование камер с разрешением меньшим или равным 350 x 548 неприемлемо. Для того чтобы достичь средней точности меньше 1 метра , рекомендуется использовать разрешение в районе 752x480.
- Для сложных задач с быстрым движением и размытием подвижных объектов как в последовательности V2\_03\_difficult желательно использовать разрешение камер около 752x480 но никак не ниже , так как ошибка значительно растет с уменьшением разрешения на сложных данных.

## **3.2 Исследование влияния параметров IMU**

### **3.2.1 Частота**

В данном подразделе будет разрабатываться система оценивающая влияние частоты получаемых данных IMU на точность VINS. Процесс будет проходить аналогичным образом, поэтому будет описан более кратко.

#### **Моделирование**

Для моделирования различной частоты данных IMU было решено смоделировать следующие частоты:

- 200 Гц – оригинальное
- 100 Гц
- 50 Гц
- 25 Гц

Данные значения частот были получены путем делением предыдущего на 2. В связи с тем, что данные IMU генерируются равномерно во времени, нет возможности рассмотреть промежуточные частоты между выбранными. Так, например, если убрать не каждое второе получаемое значение IMU из данных, а каждое третье, то данные IMU перестанут быть равномерными, что не соответствует реальности.

Для примера данных было взято три последовательности из набора данных EuRoC (в скобках указаны сокращения принятые для этого подраздела):

- МН\_01\_easy (МН\_01)
- МН\_03\_medium (МН\_03)
- МН\_04\_difficult (МН\_04)

Данный выбор был обусловлен необходимостью наличия различного уровня сложности данных.

Для генерации данных использовалась программа *genbag\_freq\_imu.py*, которая работает аналогично *genbag\_resolution.py* программе описанной подробно в предыдущем подразделе. Отличием является отсутствие функции *shrink\_image* и пропуском каждого второго сообщения IMU.

При использовании данной функции было сгенерировано 9 последовательностей, по 3 уровня частоты для 3-х оригинальных последовательностей. В сумме с оригинальными данными, набор для исследования влияния шума включает 12 последовательностей.

### **Запуск VINS-Fusion**

Для запуска алгоритма VINS на полученных данных, была расширена программа *create\_VINS\_tum.sh* и добавлена возможность запустить работу алгоритма на данных с различной частотой IMU. Внутри выполнение VINS производилось аналогичным образом, за исключением только того, что для данной ситуации не изменялись оригинальные файлы конфигурации. Это связано с тем что в VINS не передаются никаких априорных данных об частоте IMU при запуске.

В результате работы скрипта *create\_VINS\_tum.sh* было создано 12 файлов .tum описывающих траектории полученные при работе VINS-Fusion на последовательностях с различным уровнем частот IMU.

### **Анализ полученных результатов**

Для анализа и сравнения полученных данных также использовался аналогичным образом инструмент EVO [10].

В результате чего был получен график APE для всех уровней частоты IMU для последовательности MH\_01\_easy, см. рисунок 15.

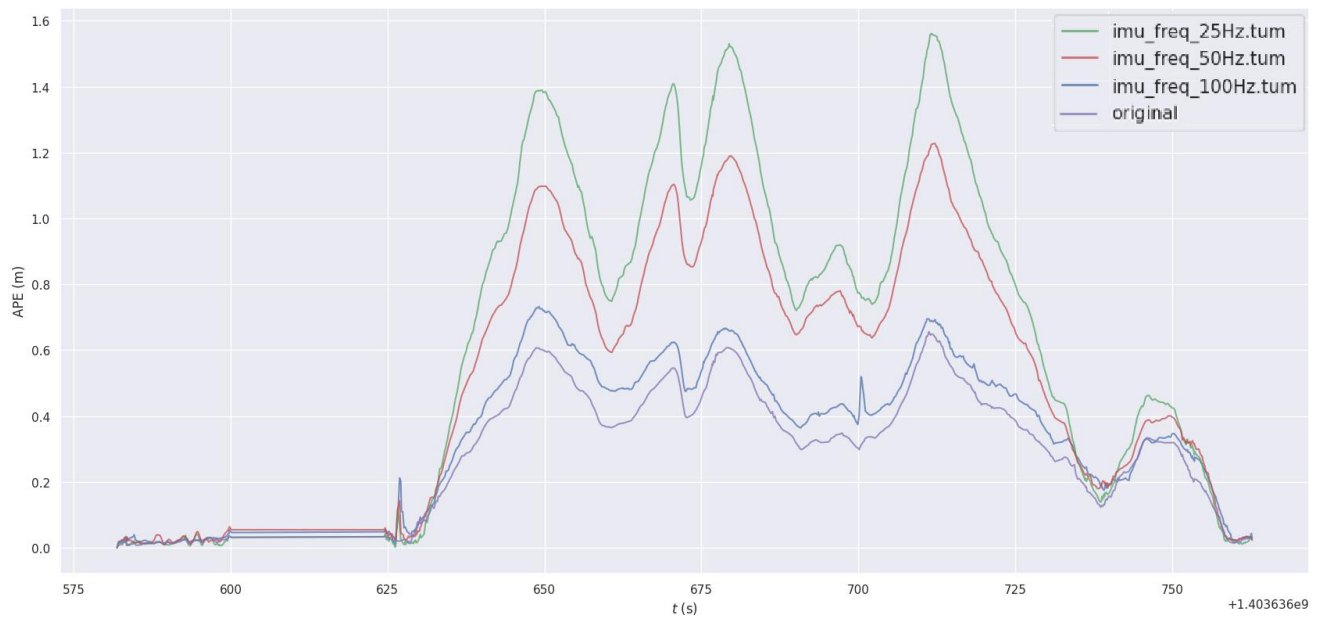


Рисунок 15 – График APE VINS-Fusion для всех уровней частоты IMU для последовательности МН\_01\_easy.

Для большей визуальной наглядности, данная информация была представлена в виде box-графика, см. рисунок 16.

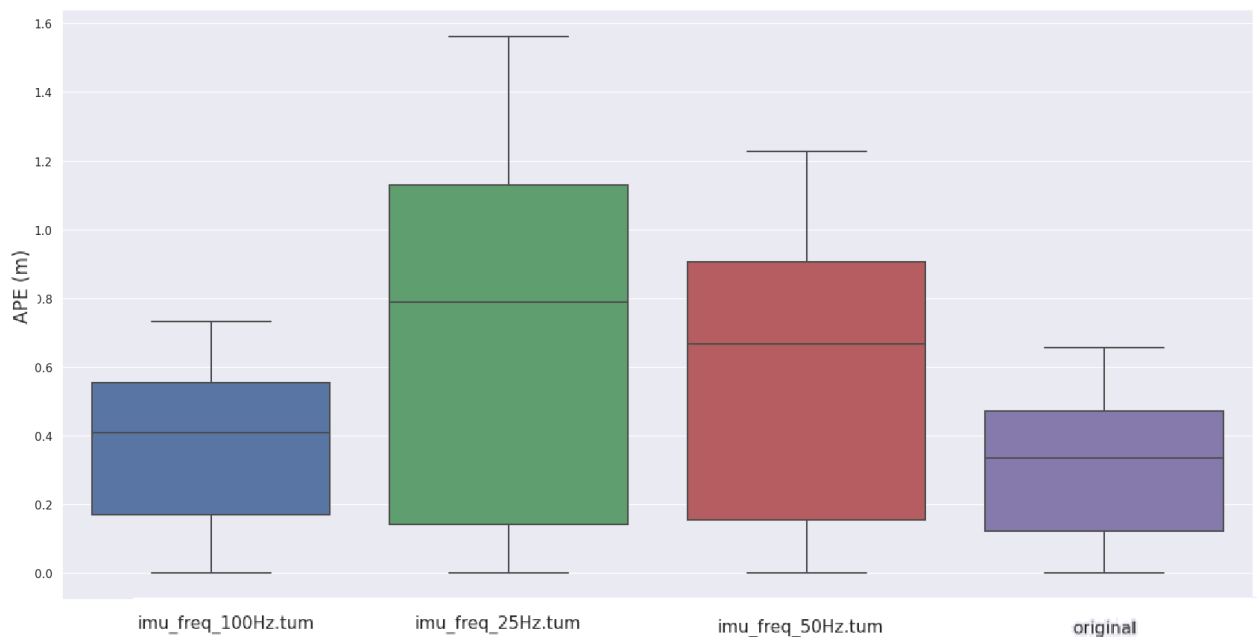


Рисунок 16 – Box-график APE VINS-Fusion для всех уровней частоты IMU для последовательности МН\_01\_easy

Также с помощью команды `evo_res` и программы `cvs_to_latex.py` была сгенерирована таблица, в которую были собраны все полученные статистики для всех 12 последовательностей, см. рисунок 17.

	rmse	mean	median	std	min	max
MH_01 org	0.37	0.31	0.34	0.2	0.0	0.66
MH_01 100Hz	0.43	0.37	0.41	0.23	0.0	0.73
MH_01 50Hz	0.69	0.57	0.67	0.4	0.0	1.23
MH_01 25Hz	0.86	0.69	0.79	0.51	0.0	1.56
MH_03 org	0.44	0.39	0.42	0.2	0.0	0.96
MH_03 100Hz	0.41	0.36	0.35	0.2	0.0	0.96
MH_03 50Hz	0.92	0.79	0.81	0.48	0.0	1.94
MH_03 25Hz	0.79	0.69	0.7	0.39	0.0	1.69
MH_04 org	0.71	0.59	0.58	0.39	0.0	1.37
MH_04 100Hz	0.68	0.59	0.6	0.35	0.0	1.35
MH_04 50Hz	0.63	0.54	0.56	0.33	0.0	1.16
MH_04 25Hz	1.73	1.45	1.46	0.94	0.0	2.79

Рисунок 17 – Таблица статистик APE VINS-Fusion для каждой рассматриваемой последовательности данных с различными уровнями частоты IMU.

В ходе разработки, был создана система, позволяющий исследовать точность VINS-Fusion на различных уровнях частот IMU датчика. В ходе практического использования были получены:

1. График APE VINS-Fusion для различных уровней частот MH\_01\_easy, рисунок 15 и 16.
2. Таблица статистик APE VINS-Fusion для всех рассматриваемых последовательностей: MH\_01\_easy, MH\_03\_medium, MH\_04\_difficult



## Интерпретация результатов

Как и ожидалось из рисунка 15 видно, что чем меньше частота IMU тем меньше поступает информации об окружении в алгоритм и тем хуже будет работать алгоритм. Так же можно обратить внимание на то что на последовательности МН\_01\_easy при изменении частоты IMU с 100Гц на 50Гц наблюдается значительный скачок в ошибке алгоритма.

Из таблицы из рис.17 можно сделать вывод что чем сложнее данные тем меньше точность, об этом говорит то что на простой последовательности МН\_01\_easy средне квадратическая ошибка была 0.37м , а уже на МН\_04\_difficult 0.71 м.

Имея эти данные потенциальный разработчик навигационной системы может определить необходимые требования к частоте датчика IMU для работы на этих последовательностях , либо запустить разработанную систему на пользовательских данных для получения информации об влиянии частоты IMU на точность VINS-Fusion на своих данных.

### 3.2.2 Белый шум акселерометра.

В данном подразделе будет разрабатываться система оценивающая влияние белого шума акселерометра на точность VINS. Процесс будет проходить аналогичным образом как в разделе «3.1.1 Разрешение», поэтому будет описан более кратко.

## Моделирование

Шум акселерометра описывается двумя составляющими, смещением нуля *bias* и гауссовским (белым) шумом:

$$\begin{aligned} m &= m_t + bias + \eta_w & \eta_w &\sim N(0, \sigma_w^2) \\ bias &= b + \eta_b & \eta_b &\sim N(0, \sigma_b^2) \end{aligned}$$

Где,

$m_t$  – истинное значение ускорения

$m$  – зашумленное значение акселерометра

$bias$  – смещением нуля

$\eta_w$  – гауссовский(белый) шум

$b$  – фиксированная ошибка смещения

$\eta_b$  – гауссовский шум смещения нуля

$N(\mu, \sigma^2)$  – нормальное распределение

Для моделирования различного уровня белого шума акселерометра было решено смоделировать шум со следующим значением среднеквадратического отклонения  $\sigma$ :

- $\sigma = 0.1$  – оригинальное
- $\sigma = 0.2$
- $\sigma = 0.4$
- $\sigma = 0.8$
- $\sigma = 1.6$
- $\sigma = 3.2$

Данные значения  $\sigma$  были получены путем умножением предыдущего на 2.

Для примера данных было взято три последовательности из набора данных EuRoC (в скобках указаны сокращения принятые для этого подраздела):

- V1\_01\_easy (V1\_01)
- V1\_02\_medium (V1\_02)
- V1\_03\_difficult (V1\_03)

Данный выбор был обусловлен необходимостью наличия различного уровня сложности данных.

Для моделирования различного уровня белого шума акселерометра необходимо учесть, что в оригинальных данных IMU уже присутствует определенный белый шум.

Поэтому при добавлении белого шума  $\eta_a$  к уже присутствующему шуму  $\eta_w$  в результате получится уровень белого шума  $\eta'_w$ .

Из теории вероятности известен закон сложения нормально распределенных случайных величин:

$$\begin{aligned} X &\sim N(\mu_x, \sigma_x^2) \\ Y &\sim N(\mu_y, \sigma_y^2) \\ Z &= X + Y \\ Z &\sim N(\mu_x + \mu_y, \sigma_x^2 + \sigma_y^2) \end{aligned}$$

В нашем случае,

$$\begin{aligned} \eta_w &\sim N(0, \sigma_w^2) \\ \eta_a &\sim N(0, \sigma_a^2) \\ \eta'_w &= \eta_w + \eta_a \\ \eta'_w &\sim N(0, \sigma_w^2 + \sigma_a^2) \end{aligned}$$

Таким образом если мы хотим получить новый шум  $\eta'_w \sim N(0, \sigma_w'^2)$  зная изначальный уровень шума  $\eta_w \sim N(0, \sigma_w^2)$  нам необходимо добавить  $\eta_a$  со следующим значением СКО  $\sigma_a$ :

$$\begin{aligned} \sigma_w'^2 &= \sigma_w^2 + \sigma_a^2 \\ \sigma_a &= \sqrt{\sigma_w'^2 - \sigma_w^2} \end{aligned}$$

Данный подход реализуется в программе ***genbag\_wnoise\_imu.py***, которая работает аналогично *genbag\_resolution.py*. Отличием является замена функции *shrink\_image* на *add\_noise\_acc*, которая вызывается для каждого значения акселерометра и в которой реализуется описанный выше способ моделирования белого шума.

В результате использования данной функции было сгенерировано 15 последовательностей, по 5 различных уровней белого шума для 3-х оригинальных последовательностей. В сумме с оригинальными данными, набор для исследования влияния шума включает 18 последовательностей.

### Запуск VINS-Fusion

Для запуска алгоритма VINS на полученных данных, была расширена программа *create\_VINS\_tum.sh* и добавлена возможность запустить работу алгоритма на данных с различным уровнем белого шума акселерометра. Были созданы соответствующие конфигурационные файлы для каждого уровня белого шума. Внутри программы выполнение VINS производилось аналогичным образом, как и до этого.

В результате работы скрипта *create\_VINS\_tum.sh* было создано 18 файлов .tum описывающих траектории полученные при работе VINS-Fusion на последовательностях с различным уровнем белого шума акселерометра.

### Анализ полученных результатов

Для анализа и сравнения полученных данных также использовался аналогичным образом инструмент EVO [10].

В результате чего был получен график APE для всех уровней белого шума акселерометра для последовательности V1\_01\_easy, см. рисунок 18.

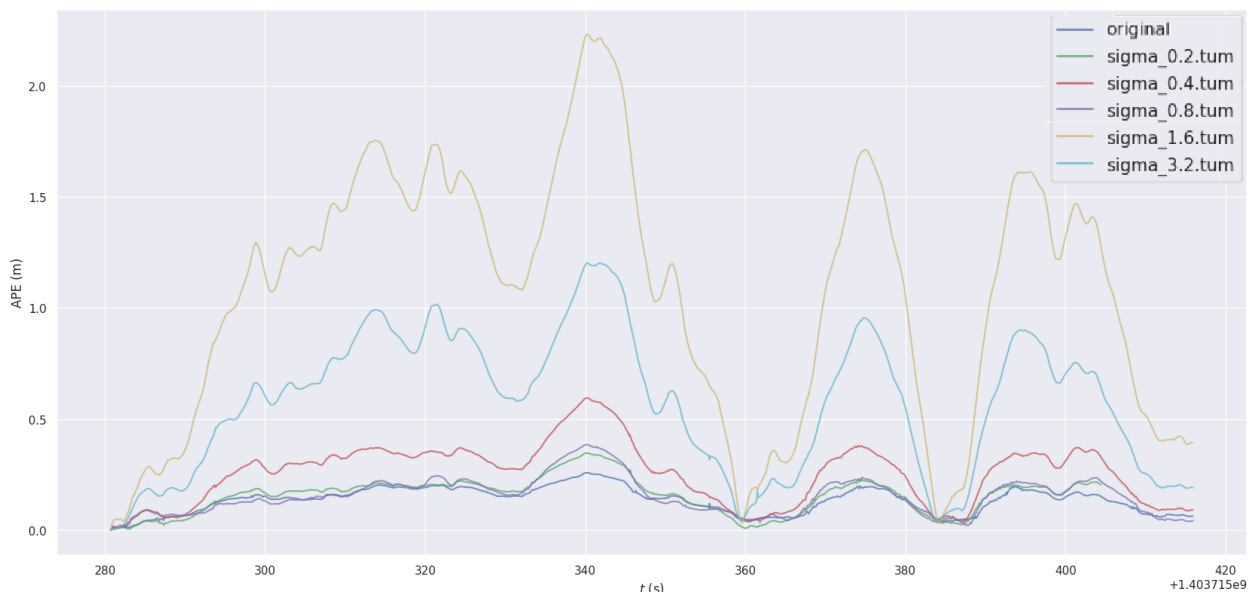


Рисунок 18 – График APE VINS-Fusion для всех уровней белого шума акселерометра для последовательности V1\_01\_easy.

Для большей визуальной наглядности, данная информация была представлена в виде box-графика, см. рисунок 19.

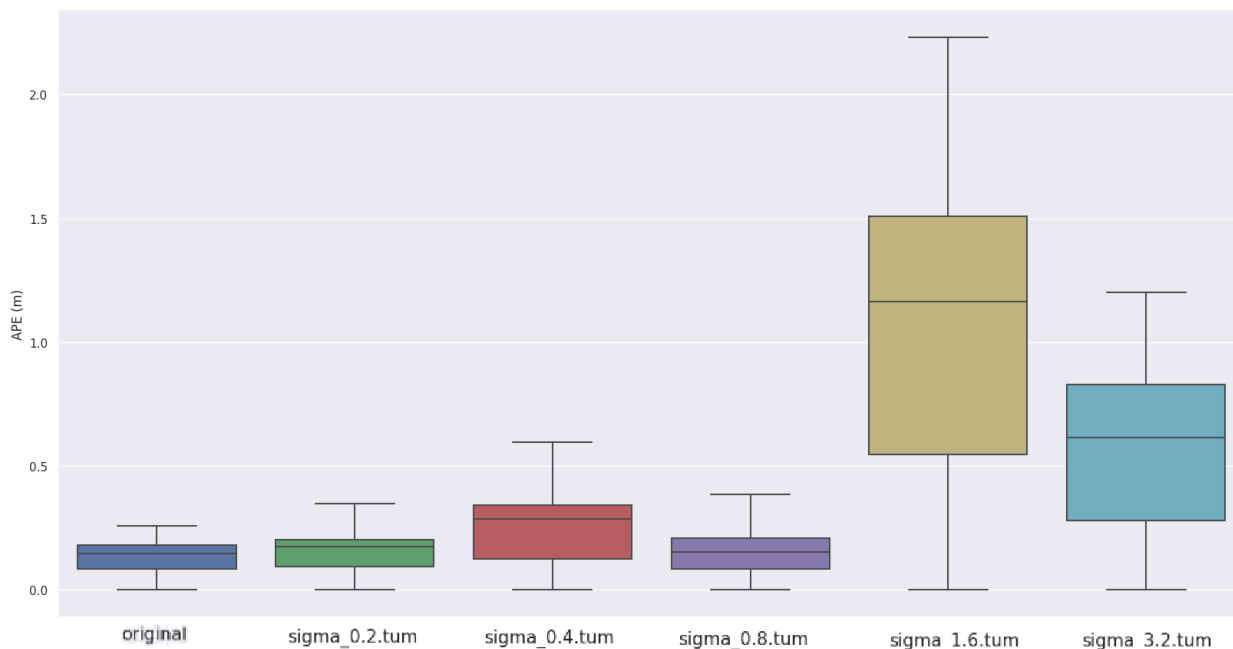


Рисунок 19 – Box-график APE VINS-Fusion для всех уровней белого шума для последовательности V1\_01\_easy

Также с помощью команды `evo_res` и программы `cvs_to_latex.py` была сгенерирована таблица, в которую были собраны все полученные статистики для всех 18 последовательностей, см. рисунок 20.

	rmse	mean	median	std	min	max
V1_01 org	0.15	0.14	0.15	0.06	0.0	0.26
V1_01 sigma 0.2	0.18	0.16	0.18	0.08	0.0	0.35
V1_01 sigma 0.4	0.29	0.26	0.28	0.14	0.0	0.59
V1_01 sigma 0.8	0.18	0.15	0.15	0.08	0.0	0.38
V1_01 sigma 1.6	1.21	1.07	1.16	0.57	0.0	2.23
V1_01 sigma 3.2	0.65	0.58	0.61	0.31	0.0	1.2
V1_02 org	0.22	0.2	0.21	0.07	0.0	0.37
V1_02 sigma 0.2	0.15	0.14	0.14	0.04	0.0	0.26
V1_02 sigma 0.4	0.24	0.22	0.23	0.11	0.0	0.45
V1_02 sigma 0.8	0.2	0.19	0.19	0.06	0.0	0.32
V1_02 sigma 1.6	0.17	0.16	0.17	0.05	0.0	0.24
V1_02 sigma 3.2	0.24	0.21	0.22	0.11	0.0	0.57
V1_03 org	0.35	0.33	0.3	0.11	0.0	0.65
V1_03 sigma 0.2	0.58	0.56	0.55	0.16	0.0	1.01
V1_03 sigma 0.4	0.17	0.16	0.15	0.06	0.0	0.33
V1_03 sigma 0.8	0.18	0.17	0.17	0.07	0.0	0.32
V1_03 sigma 1.6	0.27	0.24	0.23	0.11	0.0	0.54
V1_03 sigma 3.2	1.45	1.41	1.46	0.34	0.0	1.92

Рисунок 20 – Таблица статистик APE VINS-Fusion для каждой рассматриваемой последовательности данных с различными уровнями белого шума акселерометра.

В ходе разработки, был создана система, позволяющий исследовать точность VINS-Fusion на различных уровнях белого шума акселерометра. В ходе практического использования были получены:

1. График APE VINS-Fusion для различных уровней частот V1\_01\_easy, рисунок 18 и 19.
2. Таблица статистик APE VINS-Fusion для всех рассматриваемых последовательностей с различными уровнями белого шума акселерометра: V1\_01\_easy, V1\_02\_medium, V1\_03\_difficult. Рисунок 20.

### **Интерпретация результатов**

Как и ожидалось из рисунка 18 видно, что чем больше уровень белого шума акселерометра, тем хуже точность VINS-Fusion, так как данные из датчика подвержены более сильному искажению. Так же из рисунка 19 можно видеть на последовательности V1\_01\_easy при изменении шума с  $\sigma = 0.8$  до  $\sigma = 1.6$  наблюдается значительный скачок в ошибке алгоритма.

Из таблицы из рис.20 можно видеть, что на наборе V1\_02\_medium увеличение белого шума акселерометра почти не повлияло на точность алгоритма, и на такой последовательности имеет смысл использовать более дешевые датчики с более сильным уровнем белого шума.

Имея эти данные потенциальный разработчик навигационной системы может определить необходимые требования к уровню шума акселерометра для работы в этих помещениях, либо запустить разработанную систему на пользовательских данных для получения информации об влиянии уровня шума акселерометра на точность VINS-Fusion на своих данных.

### **3.2.3 Белый шум гироскопа.**

В данном подразделе будет разрабатываться система оценивающая влияние белого шума гироскопа на точность VINS. Процесс будет проходить аналогичным образом как в разделе «3.1.1 Разрешение», поэтому будет описан более кратко.

## Моделирование

Шум гироскопа описывается , аналогично шуму акселерометра. И включает две составляющие: смещение нуля  $bias$  и гауссовский (белый) шум:

$$\begin{aligned} m &= m_t + bias + \eta_w & \eta_w &\sim N(0, \sigma_w^2) \\ bias &= b + \eta_b & \eta_b &\sim N(0, \sigma_b^2) \end{aligned}$$

Где,

$m_t$  – истинное значение ускорения

$m$  – зашумленное значение акселерометра

$bias$  – смещением нуля

$\eta_w$  – гауссовский(белый) шум

$b$  – фиксированная ошибка смещения

$\eta_b$  – гауссовский шум смещения нуля

$N(\mu, \sigma^2)$  – нормальное распределение

Для моделирования различного уровня белого шума гироскопа было решено смоделировать шум со следующим значением среднеквадратического отклонения  $\sigma$ :

- $\sigma = 0.01$  – оригинальное
- $\sigma = 0.02$
- $\sigma = 0.04$
- $\sigma = 0.08$
- $\sigma = 0.16$
- $\sigma = 0.32$

Данные значения  $\sigma$  были получены путем умножением предыдущего на 2.



Для примера данных было взято три последовательности из набора данных EuRoC (в скобках указаны сокращения принятые для этого подраздела):

- V2\_01\_easy (V2\_01)
- V2\_02\_medium (V2\_02)
- V2\_03\_difficult (V2\_03)

Данный выбор был обусловлен необходимостью наличия данных различного уровня сложности.

Для моделирования различного уровня белого шума гироскопа также необходимо учесть, что в оригинальных данных IMU уже присутствует определенный белый шум.

Поэтому, аналогично случаю с акселерометром, если мы хотим получить новый шум  $\eta'_w \sim N(0, \sigma'^2_w)$  зная изначальный уровень шума  $\eta_w \sim N(0, \sigma^2_w)$  нам необходимо добавить  $\eta_a$  со следующим значением СКО  $\sigma_a$ :

$$\begin{aligned}\sigma'^2_w &= \sigma^2_w + \sigma^2_a \\ \sigma_a &= \sqrt{\sigma'^2_w - \sigma^2_w}\end{aligned}$$

Данный подход реализуется в программе ***genbag\_wnoise\_imu.py***, которая работает аналогично ***genbag\_resolution.py***. Отличием является замена функции *shrink\_image* на *add\_noise\_gyr*, которая вызывается для каждого значения гироскопа и в которой реализуется описанный выше способ моделирования белого шума.

В результате использования данной функции было сгенерировано 15 последовательностей, по 5 различных уровней белого шума для 3-х оригинальных последовательностей. В сумме с оригинальными данными, набор для исследования влияния шума включает 18 последовательностей.

## Запуск VINS-Fusion

Для запуска алгоритма VINS на полученных данных, была расширена программа *create\_VINS\_tum.sh* и добавлена возможность запустить работу алгоритма на данных с различным уровнем белого шума гироскопа. Были созданы соответствующие конфигурационные файлы для каждого уровня белого шума. Внутри программы выполнение VINS производилось аналогичным образом, как и до этого.

В результате работы скрипта *create\_VINS\_tum.sh* было создано 18 файлов .tum описывающих траектории полученные при работе VINS-Fusion на последовательностях с различным уровнем белого шума гироскопа.

### Анализ полученных результатов

Для анализа и сравнения полученных данных также использовался аналогичным образом инструмент EVO [10].

В результате чего был получен график APE для всех уровней белого шума гироскопа для последовательности V2\_01\_easy, см. рисунок 21.

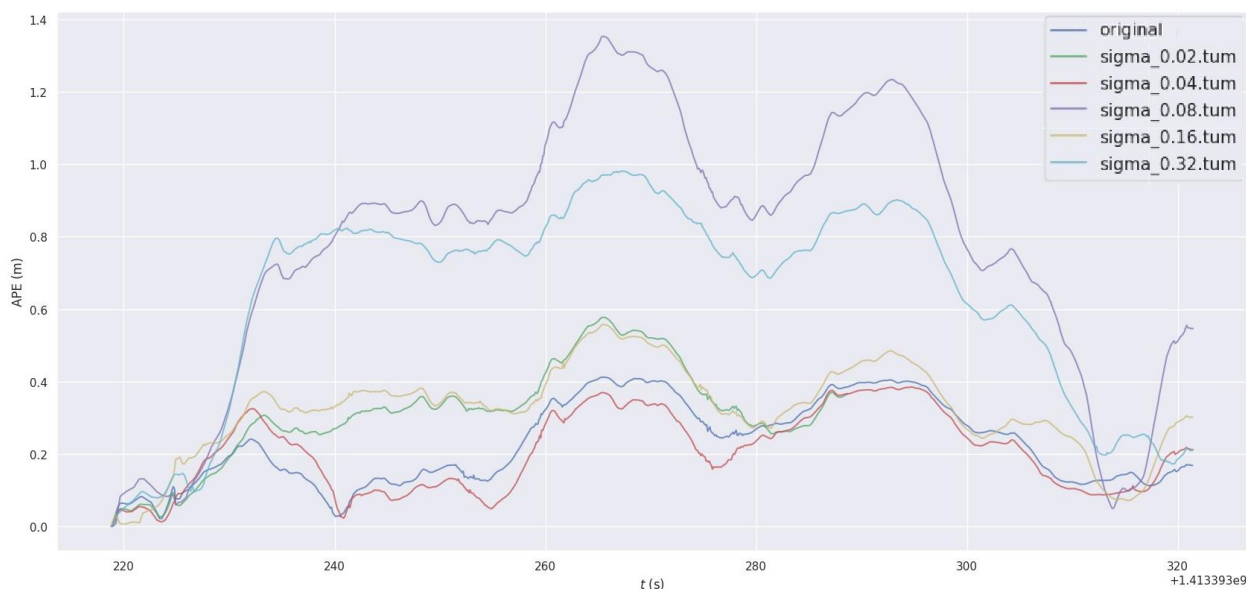


Рисунок 21 – График APE VINS-Fusion для всех уровней белого шума гироскопа для последовательности V2\_01\_easy.

Для большей визуальной наглядности, данная информация была представлена в виде box-графика, см. рисунок 22.

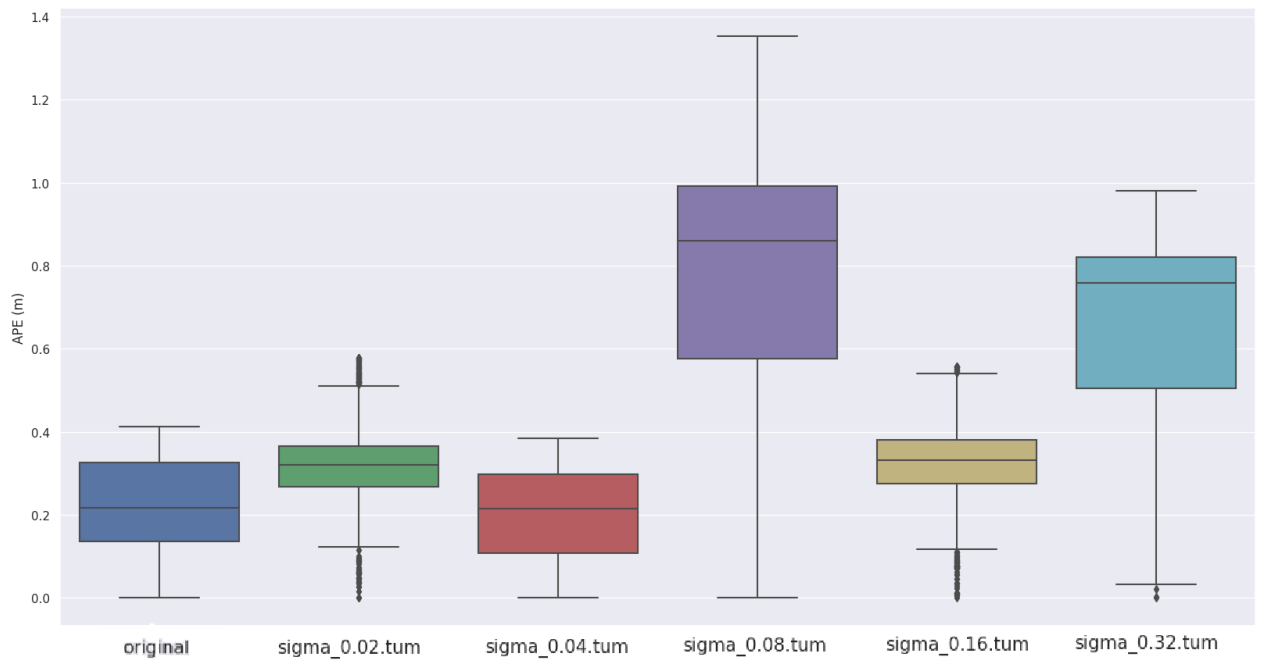


Рисунок 22 – Вох-график APE VINS-Fusion для всех уровней белого шума гироскопа для последовательности V2\_01\_easy

Также с помощью команды `evo_res` и программы `cvs_to_latex.py` была сгенерирована таблица, в которую были собраны все полученные статистики для всех 18 последовательностей, см. рисунок 23.

	rmse	mean	median	std	min	max
V2_01 org	0.25	0.23	0.22	0.11	0.0	0.41
V2_01 sigma 0.02	0.35	0.32	0.32	0.12	0.0	0.58
V2_01 sigma 0.04	0.23	0.21	0.21	0.1	0.0	0.38
V2_01 sigma 0.08	0.85	0.77	0.86	0.35	0.0	1.35
V2_01 sigma 0.16	0.34	0.32	0.33	0.12	0.0	0.56
V2_01 sigma 0.32	0.7	0.64	0.76	0.26	0.0	0.98
V2_02 org	0.3	0.28	0.27	0.11	0.0	0.54
V2_02 sigma 0.02	0.47	0.45	0.47	0.13	0.0	0.67
V2_02 sigma 0.04	0.5	0.49	0.51	0.13	0.0	0.68
V2_02 sigma 0.08	0.76	0.71	0.72	0.26	0.0	1.19
V2_02 sigma 0.16	0.41	0.35	0.32	0.22	0.0	0.8
V2_02 sigma 0.32	0.54	0.52	0.53	0.16	0.0	0.77
V2_03 org	0.23	0.22	0.21	0.07	0.0	0.37
V2_03 sigma 0.02	0.32	0.3	0.28	0.13	0.0	0.62
V2_03 sigma 0.04	0.46	0.41	0.4	0.21	0.0	0.99
V2_03 sigma 0.08	0.31	0.28	0.22	0.13	0.0	0.58
V2_03 sigma 0.16	0.39	0.36	0.37	0.16	0.0	0.85
V2_03 sigma 0.32	x	x	x	x	x	x

Рисунок 23 – Таблица статистик APE VINS-Fusion для каждой рассматриваемой последовательности данных с различными уровнями белого шума гироскопа. x – обозначено, что алгоритм потерял траекторию и ошибка превысила 100.

В ходе разработки, была создана система, позволяющая исследовать точность VINS-Fusion на различных уровнях белого шума гироскопа. В ходе практического использования были получены:

3. График APE VINS-Fusion для различных уровней частот V2\_01\_easy, рисунок 21 и 22.
4. Таблица статистик APE VINS-Fusion для всех рассматриваемых последовательностей с различными уровнями белого шума гироскопа: V2\_01\_easy, V2\_02\_medium, V2\_03\_difficult. Рисунок 23.

### **Интерпретация результатов**

Как и ожидалось из рисунка 22 видно, что чем больше уровень белого шума гироскопа, тем хуже точность VINS-Fusion, так как данные из датчика подвержены более сильному искажению.

Из таблицы из рисунка 23 видно, что на сложных данных V2\_03\_difficult VINS не справляется с белым шумом гироскопа  $\sigma = 0.32$ . Поэтому датчик с таким белым шумом нежелательно использовать на сложных данных.

Имея эти данные потенциальный разработчик навигационной системы может определить необходимые требования к уровню шума гироскопа для работы в этих помещениях, либо запустить разработанную систему на пользовательских данных для получения информации об влиянии уровня шума гироскопа на точность VINS-Fusion на своих данных.

## 4. ОБЕСПЕЧЕНИЕ КАЧЕСТВА РАЗРАБОТКИ

В данном разделе определяются требования потребителей к разработанной системе, преобразуются данные требования в характеристики качества разработки и проверяется соответствие разработанной системы данным характеристикам. В результате чего делается вывод о возможных дальнейших улучшениях системы.

### 4.1 Определение потребителей

В роли потребителей выступают предприятия и компании разрабатывающие системы автономной навигации на коммерческой основе и не только. Разработанная система помогает с определением необходимых параметров аппаратной части навигационной системы под выбранный алгоритм VINS-Fusion. Поэтому потенциальными потребителями более конкретно являются разработчики навигационных системы, работающие в данных компаниях.

### 4.2 Формулировка требований

В результате консультаций с экспертами в данной области были сформированы требования к итоговому решению. Требования к итоговому решению см. в таблице 1.

Таблица 1 – Требования к итоговому решению

Качество решения	Требование
Удобство использования	Наличие обратной связи при выполнении длительных процедур
Корректность	Результат программы на общеизвестных данных должен согласовываться с уже известными и проверенными результатами других исследователей
Модифицируемость	Наличие возможности изменения программы под нужды потребителя

### 4.3 Операциональные определения

Операциональное определение (ОО) - это уточнение значения того или иного термина применительно к данной системе, находящейся в конкретных условиях и для людей, в ней задействованных.

Для составления операционального определения требуются определить компоненты:

- **Критерий:** требование, относительно которого оценивается результат измерения или испытания;
- **Тест:** метод или процедура для измерения характеристик объекта;
- **Решение:** процедура определения соответствует ли результат испытания стандарту.

По требованиям, выявленным в п. «4.2 Формулировка требований» (см. таблицу 1), сформированы операциональные определения, см. таблица 2.

Таблица 2 – Операциональные определения для решения

Требование (Критерий)	Измерение и/или испытание (Тест)			Анализ (Решение)	
Формулирование требования (критерия)	Характеристика качества	Ед.из м	Процедура измерения и/или испытания характеристики	Целе- вое зна- чение	Процедура анализа и принятия решения о соответствии/ несоответствии

Наличие обратной связи при выполнении длительных процедур	Наличие индикатора выполнения	логическая единица	Проверка наличия индикатора выполнения программы для всех операций, выполнение которых занимают больше 5 сек.	Правда	Если для всех процедур, которые занимают больше 5 сек. времени есть индикатор выполнения программы, требование выполнено
Согласованность результатов программы с уже проверенными результатами	Корректность	логическая единица	Запуск программы на наборе данных EuRoC и сравнения полученных APE с результатами статьи [13]	Правда	Если ошибка APE на каждой последовательности не превышает 0.1, требование выполнено
Наличие возможности изменения программы под нужды потребителя	Открытый исходный код	логическая единица	Проверка выложенных в общий доступ исходного кода программы	Правда	Если исходный код выложен в общий доступ, требование выполнено

По итогам рассмотрения операциональных было определено, что разработанная система удовлетворяет всем описанным требованиям, что говорит о высоком уровне качества разработанной системы.



## ЗАКЛЮЧЕНИЕ

В ходе выполнения работы были решены все поставленные задачи, а именно: был выбран алгоритм SLAM – VINS-Fusion, был выбран набор данных EuRoC, были выбраны параметры датчиков влияние которых в дальнейшем моделировалось и исследовалось: разрешение камер, частота IMU, белый шум гироскопа и акселерометра. Была разработана система, которая позволяет моделировать данные соответствующие выбранным параметрам датчиков на основе оригинального набора данных, тестировать работу VINS-Fusion на сгенерированных данных и строить графики и таблицы отражающие зависимость точности алгоритма от параметров датчиков. Данные этапы были реализованы с помощью таких инструментов как: Python, bash, OpenCV, ROS, EVO.

Также в разделе «4. ОБЕСПЕЧЕНИЕ КАЧЕСТВА РАЗРАБОТКИ», были определены требования потребителей к разработанной системе, сформулированы операциональные определения этих требований. Разработанная система была проверена на соответствие этим требованиям.

Дальнейшие перспективы развития данной работы могут включать в себя исследование влияния следующих параметров на точность VINS:

- Для камеры:
  - Частота кадров в секунду
- Для IMU:
  - Уровень смещение нуля (bias) акселерометра
  - Уровень смещение нуля (bias) гироскопа

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Стратегия развития беспилотной авиации Российской Федерации на период до 2030 года и на перспективу до 2035 года. [Электронный ресурс].URL:  
<http://static.government.ru/media/files/3m4AHa9s3PrYTD316ibUtyEVUpnRT2x.pdf>
2. Al-Kaff A. et al. Survey of computer vision algorithms and applications for unmanned aerial vehicles //Expert Systems with Applications. – 2018. – Т. 92. – С. 447-463.
3. Gyagenda N. et al. A review of GNSS-independent UAV navigation techniques //Robotics and Autonomous Systems. – 2022. – Т. 152. – С. 104069.
4. Chen W. et al. SLAM overview: From single sensor to heterogeneous fusion //Remote Sensing. – 2022. – Т. 14. – №. 23. – С. 6033.
5. Burri M. et al. The EuRoC micro aerial vehicle datasets //The International Journal of Robotics Research. – 2016. – Т. 35. – №. 10. – С. 1157-1163.
6. Geiger A. et al. Vision meets robotics: The kitti dataset //The International Journal of Robotics Research. – 2013. – Т. 32. – №. 11. – С. 1231-1237.
7. Schubert D. et al. The TUM VI benchmark for evaluating visual-inertial odometry //2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). – IEEE, 2018. – С. 1680-1687.
8. Dellaert F. Factor graphs and GTSAM: A hands-on introduction //Georgia Institute of Technology, Tech. Rep. – 2012. – Т. 2. – С. 4.

9. Geneva P. et al. Opencvins: A research platform for visual-inertial estimation //2020 IEEE International Conference on Robotics and Automation (ICRA). – IEEE, 2020. – C. 4666-4672.
10. Grupp M. evo: Python package for the evaluation of odometry and slam. – 2017.
11. Campos C. et al. Orb-slam3: An accurate open-source library for visual, visual-inertial, and multimap slam //IEEE Transactions on Robotics. – 2021. – T. 37. – №. 6. – C. 1874-1890.
12. Forster C., Pizzoli M., Scaramuzza D. SVO: Fast semi-direct monocular visual odometry //2014 IEEE international conference on robotics and automation (ICRA). – IEEE, 2014. – C. 15-22.
13. Qin T., Li P., Shen S. Vins-mono: A robust and versatile monocular visual-inertial state estimator //IEEE Transactions on Robotics. – 2018. – T. 34. – №. 4. – C. 1004-1020.
14. Leutenegger S. et al. Keyframe-based visual-inertial odometry using nonlinear optimization //The International Journal of Robotics Research. – 2015. – T. 34. – №. 3. – C. 314-334.
15. Bloesch M. et al. Robust visual inertial odometry using a direct EKF-based approach //2015 IEEE/RSJ international conference on intelligent robots and systems (IROS). – IEEE, 2015. – C. 298-304.
16. Cao L., Ling J., Xiao X. Study on the influence of image noise on monocular feature-based visual slam based on ffdnet //Sensors. – 2020. – T. 20. – №. 17. – C. 4922.

17. Zhang K., Zuo W., Zhang L. FFDNet: Toward a fast and flexible solution for CNN-based image denoising //IEEE Transactions on Image Processing. – 2018. – T. 27. – №. 9. – C. 4608-4622.
18. Zhang S. The Research of RBPF-SLAM Accuracy under the Influence of Depth Camera Noises //2020 International Conference on Computing and Data Science (CDS). – IEEE, 2020. – C. 439-442.
19. Wang G. Robust visual SLAM with compressed image data: A study of ORB-SLAM3 performance under extreme image compression. – 2023.
20. Godio S. et al. Resolution and Frequency Effects on UAVs Semi-Direct Visual-Inertial Odometry (SVO) for Warehouse Logistics //Sensors. – 2022. – T. 22. – №. 24. – C. 9911.
21. Jeon J. et al. Run your visual-inertial odometry on NVIDIA Jetson: Benchmark tests on a micro aerial vehicle //IEEE robotics and automation letters. – 2021. – T. 6. – №. 3. – C. 5332-5339.
22. Yang N. et al. Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect //IEEE Robotics and Automation Letters. – 2018. – T. 3. – №. 4. – C. 2878-2885.
23. Getting Started » Supported Datasets | OpenVINS. [Электронный ресурс] URL:<https://docs.openvins.com/gs-datasets.html>

## **ПРИЛОЖЕНИЕ А**