

Санкт-Петербургский государственный университет

Кафедра информатики

Группа 24.M81-мм

# Навигация колёсного робота в незнакомой местности по лидару и цветной камере

***ВЛАДИМИРОВА Элина Вячеславовна***

Отчёт по учебной практике  
в форме «Производственное задание»

Научный руководитель:  
старший преподаватель кафедры информатики, к.ф.-м.н., С.И. Салищев

Санкт-Петербург  
2024

# Оглавление

<b>Введение</b>	<b>3</b>
<b>1. Постановка задачи</b>	<b>5</b>
<b>2. Обзор</b>	<b>7</b>
2.1. Задача автономной навигации . . . . .	7
2.2. Современные RGBD-сенсоры . . . . .	8
2.3. Алгоритмы визуального SLAM . . . . .	14
2.4. Требования к решению . . . . .	14
<b>3. Визуализация RGBD-данных с лидар-камеры</b>	<b>16</b>
3.1. Подключение камеры к компьютеру . . . . .	16
3.2. Подключение камеры к Raspberry Pi . . . . .	19
3.3. Подключение камеры к Raspberry Pi с экраном . . . . .	25
3.4. Получение данных с камеры по беспроводному соединению	26
<b>Заключение</b>	<b>32</b>
<b>Список литературы</b>	<b>33</b>

# Введение

Автономная навигация является одним из ключевых направлений развития мобильной робототехники. Роботы, перемещающиеся в пространстве по самостоятельно собираемым и обрабатываемым данным, находят применение во множестве сфер жизни: от науки и промышленности до бытовых задач и досуга. В последние десятилетия пилотируемые удалённо и программируемые беспилотные колёсные и летательные аппараты популярны в качестве инструмента образования.

Колёсный робот «Геобот» — образовательный проект ГК «Геоскан», представляющий собой объект разработки и тестирования программного обеспечения для управления роботом, получения и обработки визуальных и одометрических данных с его помощью. Его основная целевая аудитория — интересующиеся спортивной робототехникой и программированием школьники, студенты и взрослые. На примере «Геобота» энтузиасты учатся сборке и настройке оборудования на базе Raspberry Pi, алгоритмам управления таковым и навигации для такового.

Основной локацией использования «Геобота» является так называемая «арена» — размеченный полигон с плоским гладким полом и перемещаемыми вертикальными препятствиями для испытаний малых роботов различных типов и устраиваемых ГК «Геоскан» выездных соревнований. С объёмной виртуальной моделью арены с препятствиями стали бы возможны постановка экспериментов на полигоне в симуляции и визуализация проведённых в реальности, что сняло бы географические ограничения с соревнований по робототехнике, моделированию и программированию и открыло бы новые задачи в их рамках.

Учебная практическая работа выполняется в рамках стажировки в ГК «Геоскан». Главная цель практики — создание алгоритма навигации и картографирования для колёсного робота «Геобот» в неизвестных заранее локациях по данным механически зафиксированного на роботе и подключаемого к нему внешнего RGBD-сенсора — лидар-камеры Intel RealSense L515.

В результате стажировки и практической работы были визуализи-

рованы RGBD-данные с лидар-камеры через проводное подключение, создан алгоритм визуализации RGBD-данных с лидар-камеры в виде карты глубин через беспроводное подключение через Raspberry Pi и проведён обзор современных стереокамер с выбором из них альтернатив используемой лидар-камеры с учётом специфики задачи и технических требований проекта. Для реализации задачи SLAM выбран алгоритм ORB-SLAM3.

# 1. Постановка задачи

Целями практической работы являются:

1. Реализация алгоритма для построения и прохождения маршрута от текущего положения до заданной точки в неизвестной заранее местности с препятствиями колёсным роботом «Геобот» по данным с лидар-камеры Intel RealSense L515 с одновременным построением трёхмерной карты пространства;
2. Проверка гипотезы об улучшении качества работы алгоритма внедрением технологий глубокого машинного обучения.

Для достижения указанных целей ставились следующие задачи:

1. Задачи в рамках первой цели подразумевают проверку работоспособности лидар-камеры и её совместимости с устройствами и программным обеспечением, а также подбор подходящего под технические условия алгоритма SLAM:
  - (а) Визуализировать RGBD-данные с лидар-камеры на компьютере через проводное подключение;
  - (б) Визуализировать RGBD-данные с лидар-камеры на компьютере через беспроводное подключение к Raspberry Pi;
  - (с) Выбрать алгоритм SLAM, подходящий техническим требованиям проекта;
  - (д) Внедрить выбранный алгоритм SLAM в логику взаимодействия с камерой.
2. Задачи в рамках второй цели являются личной профессиональной инициативой и призваны улучшить результат практической работы путём внедрения нейросетевых технологий для распознавания объектов, видимых камерой:
  - (а) Выбрать подходящую предобученную нейронную сеть для семантической сегментации изображений с камеры;

- (b) Собрать датасет препятствий в целевом пространстве перемещения робота;
- (c) Дообучить нейронную сеть на собранных данных;
- (d) Сравнить качество алгоритма с использованием нейросети с качеством алгоритма без нейросети.

## 2. Обзор

### 2.1. Задача автономной навигации

Автономная навигация мобильного робота в незнакомой или динамически изменяющейся среде подразумевает решение следующих задач [3, 11]:

1. Одновременная локализация и картирование (simultaneous localization and mapping, SLAM) — определение роботом собственного положения в пространстве, построение или обновление и сохранение в памяти карты местности.
2. Планирование пути — построение роботом маршрута для достижения целевой позиции в пространстве с учётом препятствий. В усложнённой формулировке требуется оптимальность (кратчайшая возможная длина) пути.
3. Планирование движения — разбиение роботом глобального пути на короткие участки и обеспечение их прохождения с минимальным отклонением от запланированного маршрута.

Задача SLAM решается разными подходами в зависимости от используемых данных. В контексте практической работы рассматривается визуальный SLAM, входные данные для которого могут иметь вид серии «плоских» изображений, карт глубин или сочетание таковых в совокупности с показаниями одометрии и глобальной локационной системы. Лидар-камера Intel RealSense L515 [8] является RGBD-сенсором, результат работы которого — видеопоток в цветном (RGB), инфракрасном и «глубинном» (depth, D) форматах, поэтому в работе рассматриваются алгоритмы визуального SLAM, использующие RGBD-данные.

Задача планирования пути в нагруженной препятствиями среде может быть обобщена до задачи прохождения лабиринта [7, 17]. Приведённые исследования полезны демонстрацией применимости базовых алгоритмов машинного зрения в задаче картографирования и построения маршрута движения по лабиринту. Ввиду предсказуемости внеш-

него вида препятствий — стен — на пути робота достаточно монокулярного датчика и геометрических подсчетов расстояний [6].

Решение задачи планирования движения в контексте практической работы делегируется разработанной ГК «Геоскан» для «Геобота» системе автопилота, являющейся частью коммерческого продукта с закрытым исходным кодом. Задачу усложняет потенциальное возникновение в поле зрения RGBD-сенсора движущихся объектов: людей, перемещаемых ими предметов, других роботов, — обработку которого предполагается проводить на более поздних этапах работы в зависимости от результатов тестирования и доступа к автопилоту.

## 2.2. Современные RGBD-сенсоры

Визуальные данные в формате RGBD объединяют данные глубины ( $D$ , depth) и цветное изображение в стандартном формате RGB [5]. Их наиболее информативные визуальные представления — два «плоских» изображения или объёмное облако точек с наложенным на него элементы цветным фотоснимком. Наиболее распространённые современные RGBD-сенсоры — камеры глубины и лидар-камеры [12].

### 2.2.1. Основные технологии RGBD-сенсоров

Лидар-камеры [14] совмещают лазерный датчик для измерения удалённости объектов и цветную камеру для наложения на получаемую лидаром карту глубин цветов и фактур. Преимущества лидаров: устойчивость к отражениям на поверхностях, высокие точность и скорость работы, простая калибровка, надёжность — обеспечивают им высокую стоимость. Отдельной математической задачей является наложение полученного цветной камерой изображения на генерируемое лидаром, в случае современных лидар-камер решаемое предоставляемым разработчиком оборудования набором инструментов разработки (Software Development Kit, SDK).

Камеры глубины, как правило, существенно доступнее лидар-камер, т.к. используют только оптические сенсоры видимого излучения. В за-

висимости от способа работы выделяются два основных типа RGBD-камер: Time-of-Flight- (ToF-) и стереокамеры. ToF-камеры измеряют расстояние по скорости прохождения светового потока через единственный оптический сенсор, т.е. являются монокулярными камерами, в связи с чем наименее точны и устойчивы к дефектам внешней среды: за светкам, отражениям, движениям.

Стереокамеры являются более качественным вариантом благодаря двум (или более) оптическим сенсорам, зафиксированным на расстоянии друг от друга. Измерение дальности объектов производится корреляционным анализом фотографий, полученных с различных точек зрения [6], и требует точной настройки нескольких параметров обоих сенсоров: разрешения, фокусного расстояния и др. Преимуществом над лидар-камерами, помимо стоимости, является устойчивость стереокамер к солнечному свету, что расширяет возможности применения помещениями с большими окнами и локациями на открытом воздухе.

### **2.2.2. Обзор современных стереокамер**

В ходе практической работы был проведён подробный обзор современных аналогов лидар-камеры Intel RealSense L515 в качестве её потенциальной замены. Основное внимание из соображений специфики локаций использования, требований к решению и планов по приобретению устройства было удалено стереокамерам. Полный обзор вариантов со ссылками на возможные ресурсы для покупки приведён в таблице 1.

Из найденных стереокамер были выбраны и предложены как альтернативы имеющейся на данный момент в распоряжении модели следующие камеры глубины:

- Intel RealSense D435f;
- Luxonis OAK-D Lite;
- Orbbec Gemini 336L.

Таблица 1: Сравнительный анализ наиболее популярных современных стереокамер

Компания	Поддержка ROS2	Сравнение моделей. Выводы	Есть ли доставка в Россию	Цены <sup>1</sup>	Модели для покупки
ELP Dual Lens Stereo Camera	Драйвер и документация есть для ROS 1, на сайтах ROS 2 камеры не упоминаются.		Есть в офисе ГК «Геоскан»	Есть в офисе ГК «Геоскан»	— (не поддерживаются ROS2)
Intel RealSense	Поддерживается всеми версиями ROS2. SDK поддерживает всё семейство глубинных камер D400 (кроме 420, но её нет в продаже).	<ul style="list-style-type: none"> <li>- рабочий диапазон: 0.3–6м. В задаче SLAM на территории, на которой препятствия не располагаются дальше 3.5м и ориентация требует рассмотрения близких объектов, лучше всего подходят D415 (0.5–3м) и D435 (0.3–3м, шире угол обзора для глубины, есть комплектации с IMU),</li> <li>- модели с i в конце названия включают IMU (inertial measurement unit) и собирают информацию о наклонах и поворотах камеры. Рекомендуются Intel для подвижных устройств, но в задаче с плоским полом избыточны,</li> <li>- модели с f в конце названия снабжены near-infrared IR Pass Filter и меньше реагируют на отражения и преломления света. Тоже избыточная надстройка для задачи: отражающих поверхностей в офисе и на арене нет, стеклянные двери с камерой глубины по лидару и камере не отловим.</li> </ul>	Intel не поставляет в Россию. Есть предложения на Ozon и Aliexpress	Ozon: D435f (39K), D415 (30.9K, 40.8K)  Aliexpress: D415 (40.3K)	D435f

<sup>1</sup>Цены указаны в тысячах российских рублей (обозн.: К). Рассматриваются предложения на 13 ноября 2024 г. Ссылки приведены в оригинальном отчёте: <https://docs.google.com/spreadsheets/d/1cNuTN9cztnxaT0iAsxxx2HDXW0nK8R5LxLHem4N00g/edit?usp=sharing>. Их актуальность на момент защиты практической работы не гарантируется.

	Leopard Imaging Hawk	NVIDIA поддерживает свой Isaac ROS, но в практической работе используются вычислительные элементы не под NVIDIA			— (опираются на CUDA)
	StereoLabs	Поддерживается Ubuntu 22.04 и ROS2 Humble, для Jazzy пока не вышел. SDK поддерживает все модели (их всего 3). Но требует CUDA.	2 доступные модели (ZED X, ZED X Mini) ориентируются на промышленных работающих на улице роботов («agriculture, logistics and construction»), оснащены хорошим IMU, делают ставку на высокое разрешение (что может быть критично для обработки без GPU).		— (опираются на CUDA)
11	Microsoft KinectV2	Выпуск завершился в 2017. Драйвер и обертку перевезли на ROS2 Humble в двух вариантах: обновлены год назад и полгода назад, — поддержка для Humble есть, но неофициальная. Учитывая распространённость камеры, вполне возможно, что энтузиасты и дальше продолжат её обновлять.		Т.к. камера выпускалась как игровая, есть много б/у вариантов на Авито, Ozon и Aliexpress при полном отсутствии на офиц.сайте.	2-10K на разных источниках. — (EOL)

		<ul style="list-style-type: none"> <li>- для разных назначений с диапазоном хорошего качества от 0.3—1м до 8—30м (с минимальной ошибкой; Luxonis практически не указывают строгих диапазонов). Есть оптимальная Lite с диапазоном 0.5—12м,</li> <li>- включают CPU, адаптированный к нейросетевым вычислениям для отслеживания объектов и человеческих поз, запуска пользовательских нейросетей,</li> <li>- Lite включает 3 камеры: измеряющие глубину по краям и цветная 4К по центру,</li> <li>- включают IMU,</li> <li>- не так требовательны к USB-3, могут работать и с USB-2,</li> <li>- в статьях-сравнениях моделей RealSense и OAK-D показывается преимущество вторых, а Luxonis называется молодой перспективной компанией.</li> </ul>	<p>Luxonis не поставляет в Россию. На Ozon тоже ничего нет, но есть на Aliexpress и в сторонних магазинах</p>	<p>Aliexpress: OAK-D Lite (20K, 23K)</p> <p>Другие магазины: OAK-D Lite (37.3K)</p>	OAK-D Lite
Luxonis	Активно развиваются, официально поддерживаются ROS2 Humble, неофициально Jazzy (OAK-D-Lite).				

		<p>- все работают в диапазоне 0.2—6м — идеальная дальность в условиях офиса,</p> <p>- у многих из них ограничение на солнечный свет.</p> <p>Наиболее оптимальны Gemini 335L/336L (диапазон 0.25—6м с минимальной ошибкой, до 20м в целом; отличаются IR Pass Filter для обработки отражающих поверхностей у 336L),</p> <p>- включают IMU,</p> <p>- почти все передают данные только по USB-3,</p> <p>- в сравнении моделей Orbbec Gemini 2 L и RealSense D455 из блога OpenCV делается вывод о большей гладкости и стабильности карты глубин Orbbec. У Orbbec больше угол захвата на 4-5 градусов в ширину и высоту и меньше реакция на отражения (должно быть исправлено в f-версиях RealSense). RealSense лучше распознает стекло и справляется со светом. В условиях офиса и аренды из перечисленного важен только угол обзора.</p>	<p>Orbbec поставляет в Россию. Также есть предложения на Ozon и Aliexpress, но или устаревшие, или дороже оригинала.</p>	<p>Официальный сайт: Gemini 335L (36K), Gemini 336L (38K)</p> <p>Ozon: Gemini E (устар.) (20K)</p> <p>Aliexpress: Gemini 335L (51K)</p>	Gemini 335L, Gemini 336L
	Orbbec	<p>Активно развиваются, официально поддерживаются ROS2 от Foxy до Jazzy через обертку для SDK, актуальный для всех (продаваемых) камер.</p>			

## **2.3. Алгоритмы визуального SLAM**

Далее под SLAM-алгоритмами понимаются алгоритмы визуального SLAM на основе RGBD-данных и только они.

Ввиду актуальности задачи визуального SLAM с развитием технологий сбора и обработки необходимых для него данных одновременно активно развиваются и различные решения, отличающиеся скоростью и точностью работы, требованиям к данным и вычислительной аппаратуре, широтой применения. Из них к внедрению запланирован алгоритм ORB-SLAM3 [13] — выбор из ряда SLAM-алгоритмов с поддержкой RGBD-камер сделан в соответствии с требованиями к решению (2.4), совместимостью с лидар-камерой Intel RealSense L515, наличием специализированной реализации для не включающих ROS 2 систем и выводами работ [4, 16].

## **2.4. Требования к решению**

### **2.4.1. Инструменты**

Оборудование предоставлено ГК «Геоскан» (пункты 1—4) или является личной рабочей техникой, выбранной из соображений удобства и производительности (пункты 5—6):

1. колёсный робот «Геобот»;
2. компьютер Raspberry Pi 4;
3. лидар-камера Intel RealSense L515 [8];
4. провод USB type-A to type-C 2.1 длиной 1 м;
5. ноутбук ASUS Vivobook, процессор AMD Ryzen 7 5800H with Radeon Graphics;
6. ОС Windows 11 Pro, WSL2.

Камера закреплена на роботе неподвижно.

## **2.4.2. Дополнительные требования**

Ориентация компании на долгосрочное использование разработанного в ходе стажировки решения устанавливает следующие требования по версиям программного обеспечения:

- система управления роботом — ROS 2 [1];
- операционная система для Raspberry Pi поддерживается минимум до 2026 года — следовательно, из дистрибутивов Linux доступны Ubuntu не младше версии 22.04 Focal Fossa и Debian 12 Bookworm.

Специфика задачи подразумевает следующие характеристики решения:

- решение работает в режиме реального времени;
- решение ориентировано на перемещение по ровной плоскости на 2 локациях: полигон и офис студенческого конструкторского бюро — с соответствующими вертикальными препятствиями, предполагающими их обезд роботом;
- результат работы решения — субоптимальный маршрут передвижения робота в заданную точку плоскости и трёхмерная карта пространства.

### 3. Визуализация RGBD-данных с лидар-камеры

#### 3.1. Подключение камеры к компьютеру

Получение данных с лидар-камеры Intel RealSense L515 [8] производится через её подключение проводом USB type-A to type-C 3.1 к устройству, имеющему систему визуализации или способному подключиться к устройству с таковой. Для проверки работоспособности лидар-камеры и корректности получаемых с неё данных она была подключена к компьютеру проводом USB type-A to type-C 2.1 ввиду отсутствия провода требуемого поколения.

Визуализация производится в программе RealSense Viewer, элемента разработанных и поддерживаемых компанией Intel комплекта программ и библиотеки для работы с камерами Intel RealSense Intel RealSense SDK 2.0 [9]. Поскольку L515 признана Intel устаревшим оборудованием<sup>2</sup>, требовался подбор подходящей версии SDK, одновременно поддерживаемой Windows 11 и поддерживающей лидар-камеру.

Несмотря на документацию версий, среди которых последней поддерживающей L515 указана v2.54.2<sup>3</sup>, в результате последовательных сборок версий SDK из исходных файлов и тестирования с камерой выбрана наиболее современная из удовлетворяющих требованиям совместимости (запускающаяся без ошибки<sup>4</sup>) и оптимизированная под последнюю версию прошивки L515 версия — v2.47.0<sup>5</sup>. Результаты подключения показаны на скриншотах программы RealSense Viewer (рис. 1, рис. 2, рис. 3).

---

<sup>2</sup><https://www.intelrealsense.com/message-to-customers/>

<sup>3</sup>На момент сентября–ноября 2024 г.; <https://github.com/IntelRealSense/librealsense/releases>

<sup>4</sup><https://github.com/IntelRealSense/librealsense/issues/13037>

<sup>5</sup><https://github.com/IntelRealSense/librealsense/releases/tag/v2.47.0>

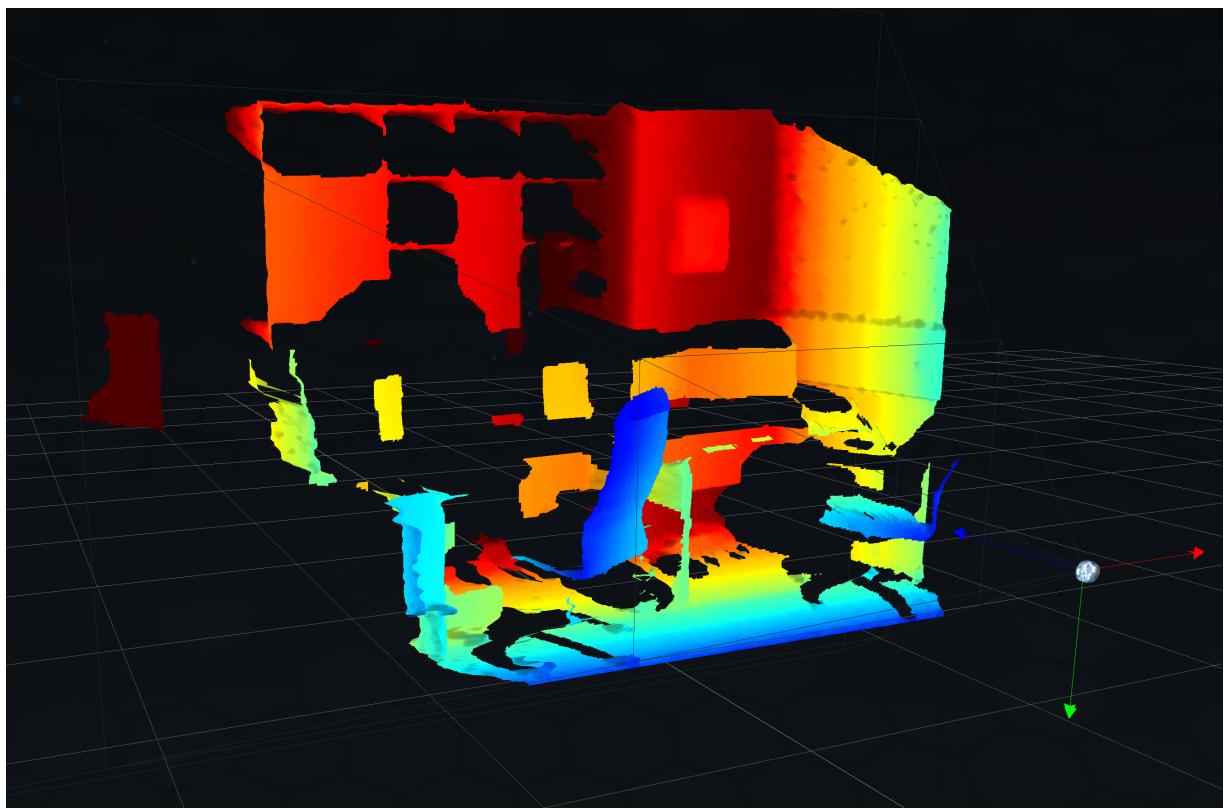


Рис. 1: Облако точек, полученное через проводное подключение L515 к компьютеру

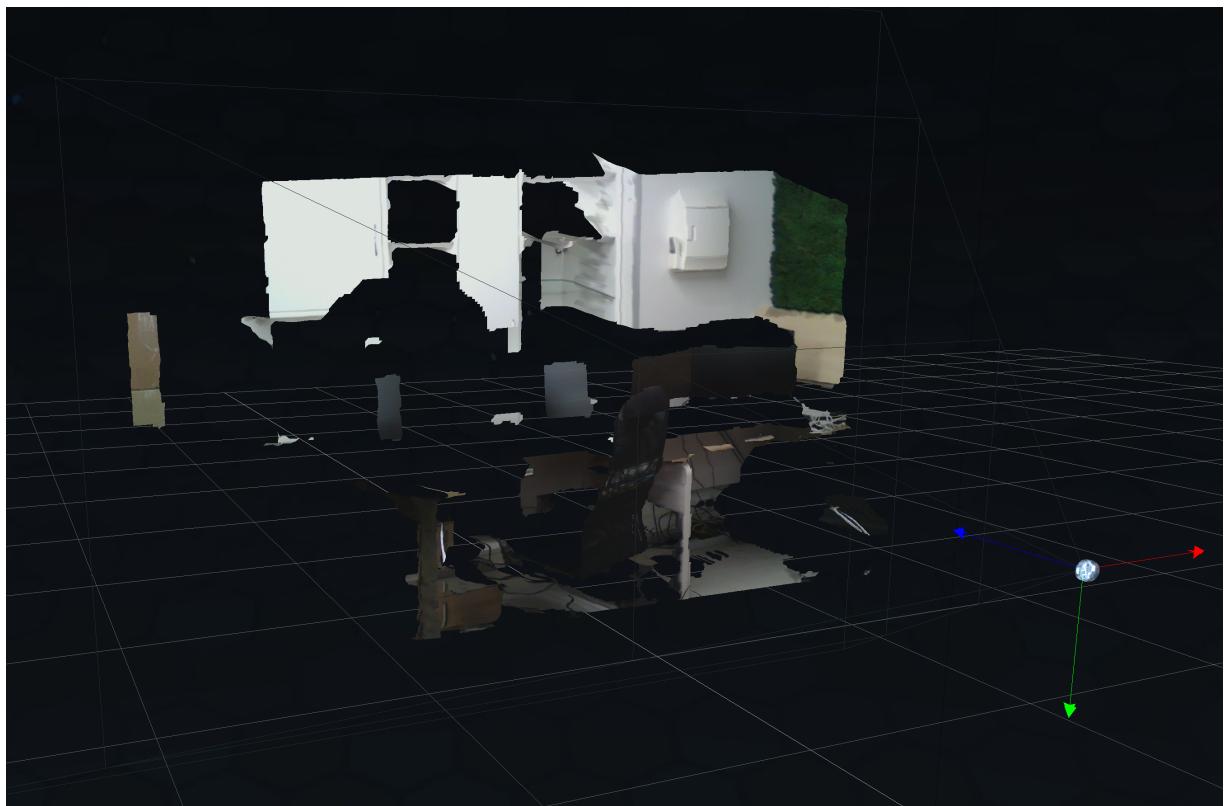


Рис. 2: Цветное облако точек, полученное через проводное подключение L515 к компьютеру

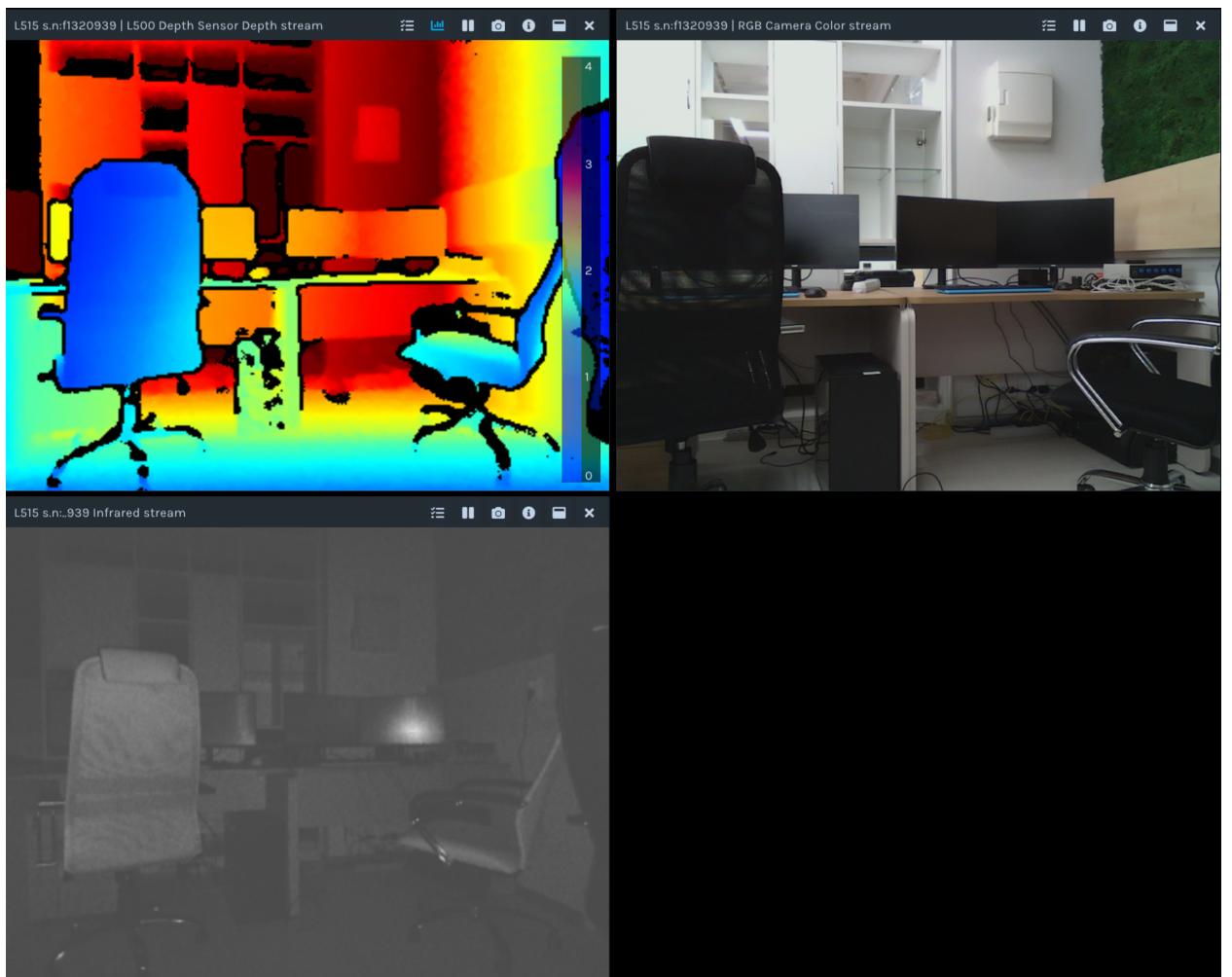


Рис. 3: Карта глубины, цветное и инфракрасное плоские изображения, полученные через проводное подключение L515 к компьютеру

## **3.2. Подключение камеры к Raspberry Pi**

Следующий после подключения лидар-камеры к компьютеру этап — адаптация камеры к Raspberry Pi 4 (далее RPi), ввиду особенностей корпуса робота не подлежащего изъятию и подключению внешнего экрана. Для этого предварительно необходимо настроить RPi:

1. с помощью программы-установщика Raspberry Pi Imager установить на sd-карту RPi Ubuntu Server 22.04.5 (64-bit);
2. в ходе установки настроить подключение RPi к сети Wi-Fi;
3. Подключиться к RPi по протоколу ssh и установить подходящую версию ROS 2 — Humble;
4. установить программу для работы с камерой RealSense с обёрткой для ROS 2;
5. проверить качество установки: передать изображение или карту глубины с лидар-камеры на компьютер.

### **3.2.1. Установка операционной системы на Raspberry Pi**

Установка операционной системы на RPi производится через установку таковой на sd-карту, подключённую через проводной порт к компьютеру с программой Raspberry Pi Imager (далее: Imager). Для установки требуется полное форматирование карты, исполняемое Imager автоматически.

Для первой карты, полученной вместе с RPi 4, автоматическое форматирование Imager исполнялось с ошибкой, в связи с чем было проведено вручную. Так как на ошибки установки форматирование не повлияло, карта была заменена на корректно работающую из другого RPi 4 — и установка адаптированной для RPi 4 операционной системы Ubuntu Server 22.04.5 (64-bit) через Imager прошла успешно.

В дальнейшем в результате многочисленных экспериментов с Ubuntu 22.04, Ubuntu 20.04 и Debian 12 будет оставлен последний дистрибутив.

### 3.2.2. Подключение Raspberry Pi к сети Wi-Fi

Для взаимодействия с роботом по протоколу ssh необходимо подключение RPi и управляющего устройства к единой сети Wi-Fi — в качестве таковой была выбрана глобальная специализированная сеть компании для управления роботами. После временного снятия оборудования, в т.ч. роутера, настройки подключения были изменены на локальную сеть Wi-Fi, раздаваемую с мобильного устройства.

### 3.2.3. Установка ROS 2 на Raspberry Pi

Для ОС Ubuntu 22.04, установленной на RPi, выпущены 2 версии ROS 2: «Humble Hawksbill» (Humble) и «Iron Irwini» (Iron) — из них была выбрана первая как более стабильная, легковесная и рекомендованная для новичков. Установка проводилась, ввиду ошибок в процессе рекомендованной официальной документацией ROS 2 Humble<sup>6</sup> установкой через бинарные файлы, способом сборки из исходных файлов<sup>7</sup> в несколько этапов, т.к. заряда аккумулятора робота не хватало для завершения процесса в рамках одного подхода. Сборка производилась с помощью `colcon`.

Поскольку в системе коммуницирующих через ROS 2 устройств одинаковые версии ROS 2 должны быть установлены на каждом из них, на компьютер были установлены дистрибутив Ubuntu 22.04.5 LTS и в его рамках версия ROS 2 Humble — по стандартной инструкции<sup>8</sup>.

Проверка корректности установки осуществлялась в 2 этапа: через возможность войти в среду разработки ROS 2 и результат запуска виртуальной модели «сервер-клиент»-коммуникации между компьютером и достижимым с него через ssh RPi. Первый этап потребовал дополнительного разбора файлов, особенно содержащих bash-скрипты; самой частой ошибкой в нём была неспособность системы обнаружить присутствовавший в директории поиска файл настройки. Указанная ошиб-

---

<sup>6</sup><https://docs.ros.org/en/humble/How-To-Guides/Installing-on-Raspberry-Pi.html>

<sup>7</sup><https://docs.ros.org/en/humble/Installation/Alternatives/Ubuntu-Development-Setup.html>

<sup>1</sup>

<sup>8</sup><https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debs.html>

ка была признана багом и устранена повторной полной сборкой ROS 2 на RPi. Второй этап служил критерием успеха и вопросов при условии корректности первого не вызвал.

### 3.2.4. Установка обёртки RealSense для ROS 2 на Raspberry Pi

Для взаимодействия камер Intel RealSense с разными аппаратурой и программным обеспечением предусмотрены специальные обёртки, в числе которых `realsense-ros` для ROS 2 [10]. Каждая обёртка устанавливается только дополнительно к основному SDK.

**Установка RealSense SDK.** Было опробовано 2 способа установки 3 версий SDK на RPi<sup>9</sup>: в качестве модуля `apt` и со сборкой из исходниковых файлов — со следующими результатами:

- Через `apt` устанавливается по умолчанию v2.55.1, новейшая стабильная версия, согласно документации, не поддерживающая L515. С ней камера действительно не воспринимается как устройство RealSense (ошибка «`No RealSense devices were found!`»), несмотря на идентификацию её RPi через `lsusb` и различный вывод тестировочной команды<sup>10</sup> при подключённой и отключенной камере. Решения этой проблемы для версии SDK 2.55.1 найдено не было: все найденные ресурсы указывают на несовместимости ядра Ubuntu, версии SDK и модели лидара.
- Установка v2.50.0, поддерживающей камеру L515, через исходные файлы также не была успешной ввиду несоответствия ядра Ubuntu ограничениям версии SDK: на RPi поставлен дистрибутив с ядром 5.15, v2.50.0 поддерживается до 5.8. Ошибки о неподдерживаемой версии ядра возникают в конце установки по инструкции для Ubuntu<sup>11</sup>. Сборка производилась с помощью `CMake`.

---

<sup>9</sup><https://github.com/IntelRealSense/realsense-ros/blob/ros2-master/README.md#installation-on-ubuntu>

<sup>10</sup>[https://github.com/IntelRealSense/librealsense/blob/master/doc/distribution\\_linux.md#installing-the-packages](https://github.com/IntelRealSense/librealsense/blob/master/doc/distribution_linux.md#installing-the-packages)

<sup>11</sup><https://github.com/IntelRealSense/librealsense/blob/master/doc/installation.md>

- С помощью пакета `linux-image-generic` была собрана версия SDK v2.47.0. Для неё ошибка распознавания камеры изменилась на «`RealSense error calling [название функции]: No device connected.`»

**Смена операционной системы.** Для устранения сохраняющейся ошибки восприятия камеры, противоречащей документации версии, была предпринята переустановка Ubuntu на 20.04.5, результатом которой стали длительные проблемы с подключением RPi 4 к заданной сети Wi-Fi (вне зависимости от сети и локации). Возврат к Ubuntu 22.04 и переход на Raspbian (Debian 12) не оказали влияния на ситуацию. Задачу было решено рассматривать на дополнительном RPi 4 с Raspbian (Debian 12) и возможностью подключения клавиатуры и экрана для непосредственного программирования, чтобы повторить успешную последовательность действий на RPi в роботе или перенести на него sd-карту с новыми настройками.

Настройка подключения производилась через графический интерфейс `nmtui`: были создана и добавлена в конфигурацию новая сеть с данными необходимой сети Wi-Fi, произведена перезагрузка системы, вручную активировано подключение к сети и в качестве тестирования связи через `ping` достигнут сайт Google. По завершении настройки подключения sd-карта из свободного RPi была перемещена в RPi в роботе — беспроводной контакт с роботом был восстановлен.

**Переустановка RealSense SDK.** После изменения операционной системы установка SDK проводилась заново по специализированной для Raspbian инструкции<sup>12</sup>. В процессе ввиду исключений `apt` («`Unable to locate package`») часть пакетов загружена из внешних архивов или заменена на новые версии, часть по совету коллег проигнорирована<sup>13</sup>, ввиду низкой скорости скачивания по локальной сети

---

<sup>12</sup>[https://github.com/IntelRealSense/librealsense/blob/master/doc/installation\\_raspbian.md](https://github.com/IntelRealSense/librealsense/blob/master/doc/installation_raspbian.md)

<sup>13</sup>Из указанных в инструкции 3 наборов пакетов используется только последний, т.е. следующие пакеты: `ibglu1-mesa`, `libglu1-mesa-dev`, `mesa-utils`, `mesa-utils-extra`, `xorg-dev`, `libgtk-3-dev`, `libusb-1.0-0-dev`

Wi-Fi репозиторий SDK загружен в несколько подходов.

Результатом этапа стал подбор успешной комбинации ОС–SDK — Raspbian (Debian 12) и Intel RealSense SDK v2.47.0 соответственно, — дающей стабильные подключения компьютера к роботу по протоколу ssh и робота к лидар-камере по проводу USB 2.1.

Для установки всех нужных функций SDK в дальнейшем потребовалась повторные полные сборки версии v2.47.0, поскольку наличие пакетов регулировалось флагами CMake.

**Переустановка ROS 2.** Коллегами была установлена и протестирована на базовом примере взаимодействия «сервер-клиент» между роботом и компьютером версия ROS 2 Jazzy для Raspbian [2] — новейшей стабильной версии на середину октября 2024 — из соображений совместимости Debian 12 и ROS 2.

Со сменой предполагаемой рабочей версии ROS 2 на RPi была сменина таковая и на личном рабочем компьютере, для чего повышена версия Ubuntu до 24.04.1 LTS.

**Установка обёртки `realsense-ros`.** Обёртка `realsense-ros` с поддержкой версии SDK v2.47.0 отсутствует, ближайшие обозначенные в списке выпусков обёртки версии — v2.45.0 и v2.48.0. В предположении наличия в коде выпуска обёртки, поддерживающего более позднюю версию SDK, существенных изменений для адаптации к таковым в более поздней версии SDK для работы с v2.47.0 была взята последняя из поддерживающих v2.45.0 — `realsense-ros` 3.2.0. Ограничения по поддерживаемым указанными выпусками `realsense-ros` устройствам включают лидар-камеру L515, но по версиям ROS 2 — только признанные устаревшими версии ROS 2.

Установка зависимостей `rosdep` для сборки `realsense-ros` 3.2.0 потребовала дополнительной загрузки избранных библиотек из внешних архивов и через `apt`. Вместе с тем, завершить сборку не удалось ввиду ошибок `colcon` на трёх существенных для работы ROS 2 библиотеках:

- `realsense2_camera_msgs` — сборка завершается с ошибкой, вы-

званной устареванием модуля `em`, и не реагирует на переустановку `empty` (импортируемой как `em` в соответствующих файлах), очистку файлов логгирования и обновление связанных с данной библиотек. Ручной разбор зависимостей в `package.xml` и отчетов о ходе сборки в `CMakeOutput.log` библиотеки `realsense2_camera_msgs` не дали информации об ошибке. Проверка прав доступа SDK к порту, куда подключена камера, не возымела результата.

- `xacro` — сборка завершается с предупреждением об устаревании версий используемых ей библиотек и зависящим от очередности относительно `realsense2_camera_msgs` исходом: успех в случае раннего завершения сборки и ошибка иначе.
- `opencv_tests` — аналогично `xacro`.

В процессе решения вопроса было предпринято прохождение пропущенного и потенциально связанного с описанной задержкой сборки шага инструкции — настройка модуля `protobuf`. При его установке по инструкции<sup>12</sup> была найдена несостыковка версий: сборка `protobuf` для старых версий SDK нереализуема без функции, убранной из библиотеки `setup-tools` с версии 58<sup>14</sup>, тогда как для Debian 12 доступны только версии `setup-tools` 66. Поскольку до обнаружения несостыковки `setup-tools` была удалена, проблема с ROS 2 решилась самоустраниением: и ROS 2, и обёртка SDK для него также оказались удалены.

Восстановление ROS 2 и `realsense-ros` с учётом нерешённых вышеуказанных трудностей прогнозировало затруднения, преодоление которых было оценено мной и коллегами как непозволительно времязатратные усилия, в связи с чем идея использовать камеру в качестве узла системы ROS 2 для автоматической навигации робота была отложена, и дальнейшие усилия были сосредоточены на получении данных с камеры с помощью RealSense SDK, их передачи на компьютер и обработки через обёртку для Python — `pyrealsense2`, — и дальнейшей реализации автоматического алгоритма SLAM с роботом, управляемым вручную.

---

<sup>14</sup><https://stackoverflow.com/a/75085788/27244963>

### 3.3. Подключение камеры к Raspberry Pi с экраном

Для демонстрации реализуемости задачи Intel RealSense SDK v2.47.0 была восстановлена на свободном RPi, аналогичном встроенному в «Геобот», с клавиатурой и экраном<sup>15</sup>, что позволило получить с лидар-камеры RGBD-данные и отобразить соответствующее цветное облако точек в среде RealSense Viewer, установленной на RPi (рис. 4).

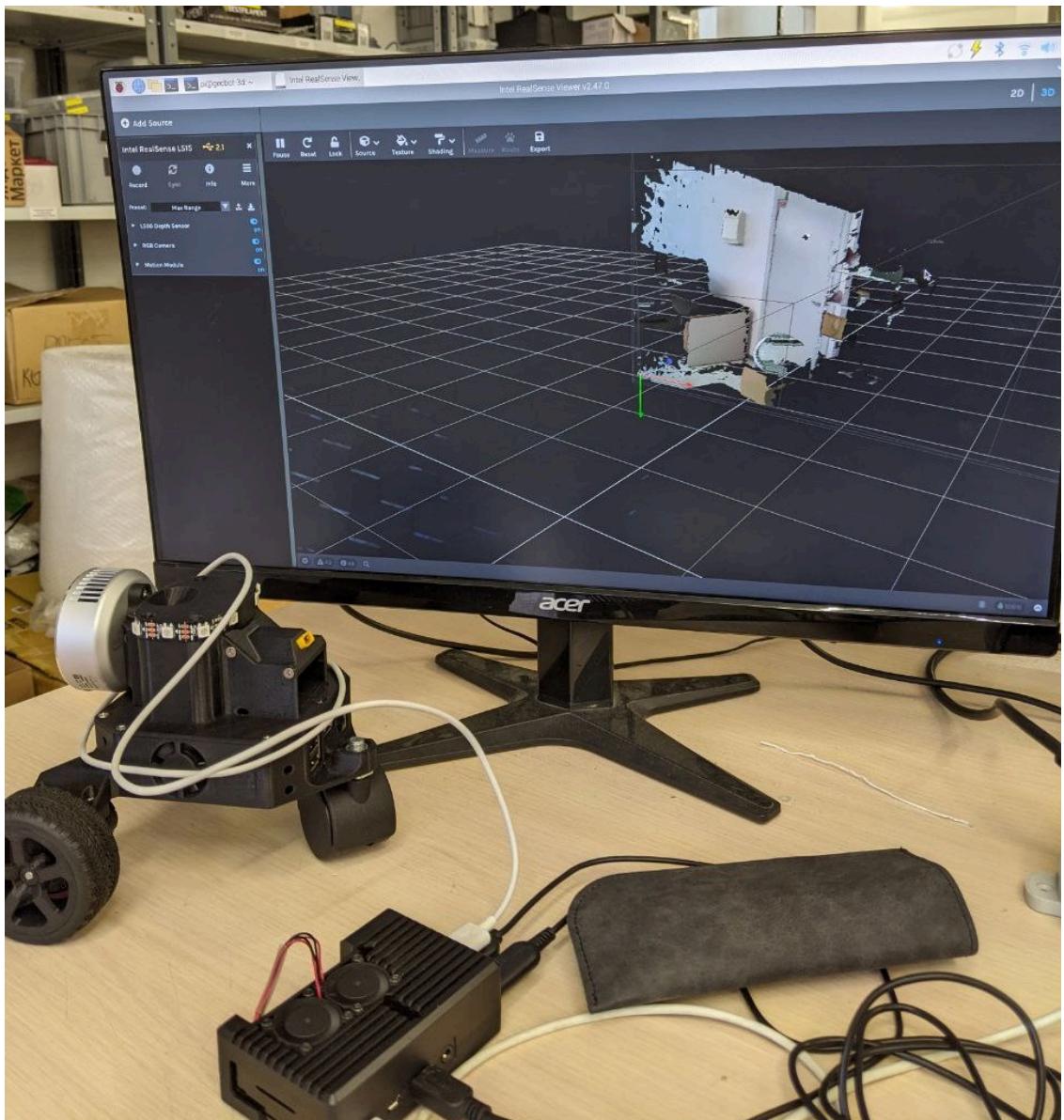


Рис. 4: Цветное облако точек в RealSense Viewer с Raspberry Pi с экраном

<sup>15</sup>[https://github.com/datasith/Ai\\_Demos\\_RPi/wiki/Raspberry-Pi-4-and-Intel-RealSense-D435](https://github.com/datasith/Ai_Demos_RPi/wiki/Raspberry-Pi-4-and-Intel-RealSense-D435)

### 3.4. Получение данных с камеры по беспроводному соединению

RealSense SDK v2.47.0 и ROS 2 Jazzy были восстановлены на RPi в работе, также для удобства программирования были настроено подключение к нему через ssh-клиент в VS Code. В качестве основного языка разработки из соображений скорости установки и написания кода был выбран Python и соответствующая ему обёртка RealSense SDK `pyrealsense2` (в пункте 3.4.2 используется сокращение названия библиотеки «`rs`», стандартное при импортировании). Предполагается следующий план взаимодействия с лидар-камерой:

1. транслировать RGBD-поток с лидар-камеры на компьютер через ssh-соединение с RPi;
2. принимать и отображать RGBD-поток на компьютере (желательно в Realsense Viewer — интерактивной среде с удобным графическим интерфейсом);
3. сохранять RGBD-поток в обновляемое облако точек окружающего пространства для получения в результате проезда роботом местности её 3D-карты.

Последний пункт представляет собой сложную алгоритмическую задачу о разрешении перекрытий на соседних кадрах. Планируется решать её готовыми SLAM-алгоритмами, из которых наиболее репрезентативен и реализован для не включающих ROS 2 систем ORB-SLAM3 [13].

#### 3.4.1. Карта глубин (изображение)

Для трансляции данных с лидар-камеры используется модифицированный пример из репозитория `pyrealsense2`<sup>16</sup>. Пусть он основан на устаревающей библиотеке `asyncore` [15], обеспечивающей асинхронность сервера и потенциально нескольких подключающихся к нему клиентов, внедрение в код новой версии библиотеки, `asyncio`, отложено из

---

<sup>16</sup>[https://github.com/IntelRealSense/librealsense/tree/master/wrappers/python/examples/etherenet\\_client\\_server](https://github.com/IntelRealSense/librealsense/tree/master/wrappers/python/examples/etherenet_client_server)

соображений временных ограничений и сохранения работоспособности и скорости работы старым вариантом.

Пример демонстрирует передачу сервером карты глубин в виде строки битов через модуль `pickle` и получение, расшифровку и отображение клиентом этих данных в виде плоского изображения фиксированного разрешения в режиме реального времени. В код примера внесены изменения по передаче и отображению данных: формат изображения подогнан под ограничения кабеля USB 2.1<sup>17</sup>, окно (исходно формата 320×240) расширено в 4 раза, цветовая схема изображения изменена с монохромной на «Jet» для удобства восприятия (рис. 5).

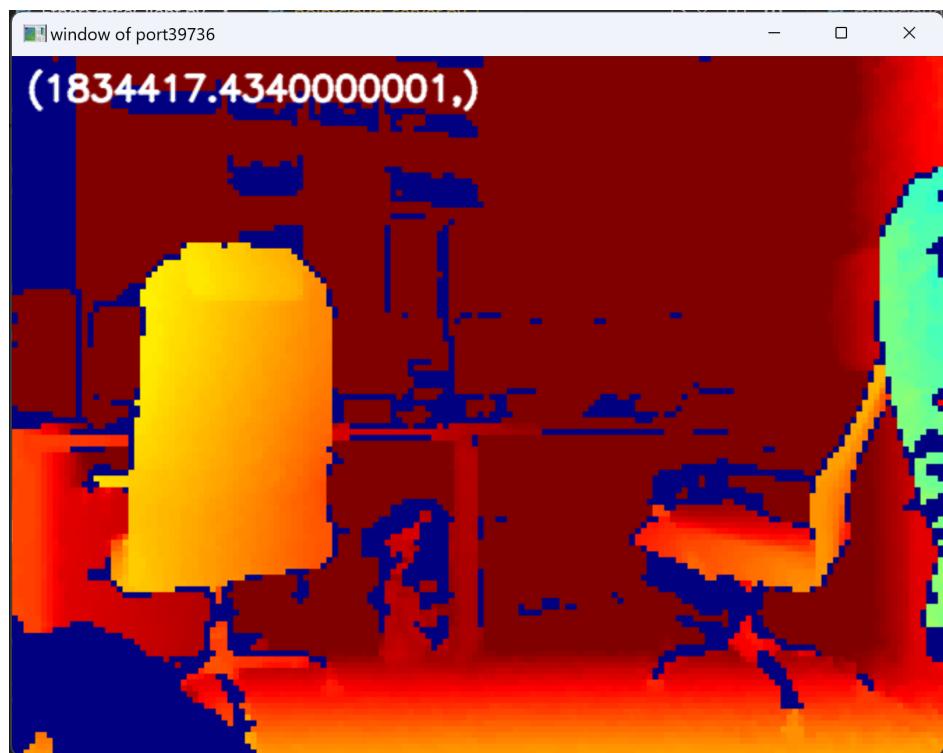


Рис. 5: Карта глубины (плоское изображение), полученная через беспроводное подключение L515 к компьютеру через Raspberry Pi 4

### 3.4.2. Облако точек

Визуализация объёмных данных является одной из ключевых задач работы, поэтому стала основным объектом внимания на последнем эта-

---

<sup>17</sup>Оригинально рекомендуется USB 3.1/3.2 ввиду существенного объёма передаваемых данных и требований по скорости передачи: <https://github.com/intelrealsense/librealsense/issues/9979#issuecomment-975706710>

пе практической работы. Были опробованы три способа визуализации облака точек, описанные в подразделах данного раздела.

**Визуализация средствами RealSense: rs-server + RealSense Viewer.** Визуализация с помощью программы RealSense Viewer — части SDK, имеющей функциональность отображения данных с подключённой по беспроводному соединению камеры, — на компьютере стала стартом и завершением работы над визуализацией облака точек в рамках практической работы.

Программа Realsense Viewer не распознавала L515 на RPi как доступное для трансляции устройство RealSense, т.к. в базовую сборку SDK на RPi не включается `rs-server`<sup>18</sup> — встроенный инструмент для беспроводной трансляции данных с камеры. Он активируется отдельными флагами `CMake`, для чего требуется повторная сборка SDK командами из инструкции<sup>15</sup> с дополнительными флагами.

Запуск Realsense Viewer с активированным на RPi `rs-server` изменил ошибку отсутствия RealSense-устройств на ошибку восприятия формата с обнаруженного RealSense-устройства:

```
[RsRTSPClient::getStreams] error: 404 File Not Found, Or  
In Incorrect Format - 404.
```

В issue<sup>19</sup> с аналогичным вопросом разработчики Intel рекомендуют в точности следовать инструкции<sup>20</sup>, в т.ч. устанавливать на RPi статический IP-адрес и в Windows подключаться к RPi и запускать `rs-server` строго через PuTTY. В Debian Bookworm изменена система регулирования IP-адресов, в связи с чем статический IP для RPi был установлен, в отличие от рекомендаций в инструкции, через `nmtui`. Поскольку на ошибку это не повлияло, были проведены экспериментальные запуски трансляции через `rs-server`:

- с RPi на Viewer, установленный на Windows 11;

---

<sup>18</sup><https://dev.intelrealsense.com/docs/open-source-ethernet-networking-for-intel-realsense-depth-cameras>

<sup>19</sup><https://github.com/IntelRealSense/librealsense/issues/10548>

<sup>20</sup><https://dev.intelrealsense.com/docs/open-source-ethernet-networking-for-intel-realsense-depth-cameras#2-quick-start>

- с RPi на Viewer, установленный на Ubuntu 22.04 LTS;
- с Ubuntu 22.04 LTS на Viewer, установленный на Ubuntu 22.04 LTS;
- с Ubuntu 22.04 LTS на GStreamer, установленный на Ubuntu 22.04 LTS.

В процессе экспериментов вскрыты для отладки исходные файлы `rs-server`. На безрезультатности этого шага гипотезы о способах устранения ошибки были исчерпаны.

**Визуализация с собственной реализацией класса `rs.pointcloud`.** Для трансляции и визуализации облака точек скомпонованы вместе образцы кода `pyrealsense2`: передача данных с сервера на клиент<sup>16</sup> и простая интерактивная среда для отображения<sup>21</sup>.

Беспроводная трансляция данных с камеры как объектов RealSense не перенесена в обёртку `pyrealsense2` (и ни в какую иную обёртку RealSense<sup>22</sup>) и реализуется фактически только RealSense Viewer. Поэтому в демонстрациях кода обёртки транслируется исключительно карта глубины, плоская проекция объёмного облака точек, как численный массив через строки битов средствами сжатия `pickle` — и расшифровывается в аналогичный численный массив, наивно преобразуемый в изображение. Объект `rs.depth_frame` (далее «фрейм»), из которого подсчитывается объект `rs.pointcloud` (далее «облако точек»), и само облако точек не поддаются операциям `pickle` и не перемещаются с устройства на устройство встроенными методами `pyrealsense2`.

Т.к. объекты RealSense конструируются только через опирающиеся на камеру методы, получить на клиенте (компьютер) фрейм или облако точек в том виде, в котором они существовали на сервере (роботе с камерой), в текущих условиях не представляется возможным.

Простой вариант разрешения этого затруднения — воссоздание фрейма через аналогичный замещающий объект путём трансляции с

---

<sup>21</sup>[https://github.com/IntelRealSense/librealsense/blob/master/wrappers/python/examples/opencv\\_pointcloud\\_viewer.py](https://github.com/IntelRealSense/librealsense/blob/master/wrappers/python/examples/opencv_pointcloud_viewer.py)

<sup>22</sup><https://dev.intelrealsense.com/docs/open-source-ethernet-networking-for-intel-realsense-depth-cameras#32-building-from-source>

сервера всех численных полей фрейма в виде словаря. Но реализация простой среды визуализации из `pyrealsense2` использует существенное количество методов оригинального класса, реализованных на C/C++ и недостижимых для замещающего. Перенос их на Python займет много времени и снизит скорость работы программы, в связи с чем кажется нежелательным решением.

Путь не признан тупиковым, но временно оставлен — или до зелёного света от старших коллег, или до внедрения нужной функциональности в обёртку `pyrealsense2` (Intel планирует это с 2018 года<sup>23</sup>).

**Визуализация с воссозданием `rs.depth_frame` средствами RealSense.** Данное направление представляет собой ответвление описанной в предыдущем пункте идеи. Т.к. на клиенте используется множество функций SDK, реализованных на C/C++, был найден способ получить имеющий к ним доступ фрейм как объект `pyrealsense2` эмуляцией сенсора (камеры) через `rs.software_device` — класс, позволяющий создавать виртуальные «устройства» и передавать им сохранённые данные для конструкции фреймов с их помощью. Класс `rs.software_device` не реализован в `pyrealsense2`, но на GitHub встречаются попытки его использования напрямую<sup>24</sup> (все найденные issues закрыты без решения, репозитории отсутствуют).

Внесение класса в код клиентской части приводит к ошибке на этапе создания объекта `rs.software_device`:

```
RuntimeError: failed to convert special folder: errno=42
```

В ходе отладки был проверен внутренний файл инструментов отладки VS Code, на котором появляется ошибка: место появления связано с проверкой на асинхронность. Предполагаю одной из возможных причин устаревший модуль `asyncore`. Его замена на новую версию, `asyncio`, требует времени, но кажется мне более перспективной по сравнению с полноценным переносом класса в Python.

---

<sup>23</sup><https://github.com/IntelRealSense/librealsense/issues/2551>

<sup>24</sup><https://github.com/IntelRealSense/librealsense/issues/7057>, <https://support.intelrealsense.com/hc/en-us/community/posts/1500000934242-exploring-equivalent-API-in-python-for-rs2-software-device-create-matcher-RS2-MATCHER-DLR-C-defined-in-c>

На текущий момент одновременная трансляция цветного и глубинного потоков работает с ошибкой, указанной в предыдущем абзаце. Гипотеза о провоцирующих таковые ограничениях проводного канала передачи данных сформулирована мной из эмпирических соображений и нуждается в проверке информацией из иных источников. Вместе с тем, код проекта предусматривает возможность передачи и обработки RGBD-данных, а также наложения цветов на облако точек.

**Выводы из этапа.** На основании результатов работы по отображению облака точек был сделан следующий вывод: для отображения данных с лидар-камеры Intel RealSense L515 через Raspberry Pi 4 в RealSense Viewer необходимо писать собственное программное обеспечение на C/C++.

# Заключение

В ходе учебной практической работы в течение I семестра были получены следующие результаты:

1. Подобрана конфигурация версий ОС, лидар-камеры и SDK для визуализации RGBD-данных с лидар-камеры на компьютере через проводное подключение;
2. Подобрана конфигурация версий ОС, лидар-камеры и SDK для визуализации RGBD-данных с лидар-камеры на Raspberry Pi 4 через проводное подключение;
3. Создан алгоритм визуализации RGBD-данных с лидар-камеры в виде карты глубин на компьютере через беспроводное подключение к Raspberry Pi 4;
4. Составлен обзор стереокамер, поддерживаемых используемыми версиями дистрибутивов Linux и совместимых с современными версиями ROS 2, от 7 производителей. Предложены альтернативные Intel RealSense L515 варианты;
5. Выбран и предложен к реализации SLAM-алгоритм ORB-SLAM3.

Ссылка на GitHub-репозиторий с исходным кодом основной части практической работы: <https://github.com/Ali-AliAli-Ali/mapping-with-L515>.

На текущий момент продолжается работа над задачей визуализации RGBD-данных с лидар-камеры в виде облаков точек на компьютере через беспроводное подключение к Raspberry Pi. Во II семестре будет осуществлена работа над задачами внесения выбранного алгоритма SLAM в логику взаимодействия с лидар-камерой, внедрения в полученную программу нейросетевого алгоритма обработки RGBD-данных и тестирования полученного решения на результативность.

## Список литературы

- [1] Al-Batati A.S., Koubaa A., Abdelkader M. ROS 2 in a Nutshell: A Survey. — 2024.
- [2] Ar-Ray-code. Raspberry Pi OS ROS2. — URL: <https://github.com/Ar-Ray-code/rpi-bullseye-ros2> (дата обращения: 17 декабря 2024 г.).
- [3] Arvanitakis I., Giannousakis K., Tzes A. Mobile robot navigation in unknown environment based on exploration principles // Conference on Control Applications (CCA). — 2016.
- [4] Comparative Evaluation of RGB-D SLAM Methods for Humanoid Robot Localization and Mapping / A. Vedadi, A. Yousefi-Koma, P. Yazdankhah, A. Mozayyan. — arXiv, 2023. — P. 807–812.
- [5] Firman M. RGBD Datasets: Past, Present and Future // Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). — 2016.
- [6] Hartley R., Zisserman A. Multiple View Geometry in Computer Vision. — 2003.
- [7] Implementation of NAO Robot Maze Navigation Based on Computer Vision and Collaborative Learning / D. Magallan-Ramirez, J. Martinez-Aguilar, A. Rodriguez-Tirado et al. // Frontiers in Robotics and AI. — 2022. — Vol. 9.
- [8] Intel RealSense. Intel RealSense LiDAR Camera L515. — URL: <https://www.intelrealsense.com/lidar-camera-l515/> (дата обращения: 17 декабря 2024 г.).
- [9] Intel RealSense. Intel RealSense SDK 2.0. — URL: <https://github.com/IntelRealSense/librealsense> (дата обращения: 17 декабря 2024 г.).

- [10] Intel RealSense. Intel RealSense realsense-ros. — URL: <https://github.com/IntelRealSense/realsense-ros> (дата обращения: 17 декабря 2024 г.).
- [11] Kumar N., Takacs M., Vamossy Z. Robot navigation in unknown environment using fuzzy logic // International Symposium on Applied Machine Intelligence and Informatics. — 2017.
- [12] Levine M. Towards Safe, Real-Time Systems: Stereo vs Images and LiDAR for 3D Object Detection. — 2022.
- [13] ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM / C. Campos, R. Elvira, J.J.G. Rodriguez et al. // IEEE Transactions on Robotics. — 2021. — P. 1874–1890.
- [14] Pyroistech. LiDAR: working principle and main applications. — URL: <https://www.pyroistech.com/lidar/> (дата обращения: 19 декабря 2021 г.).
- [15] Python. asyncio — Asynchronous socket handler. — URL: <https://docs.python.org/3/library/asyncio.html> (дата обращения: 17 декабря 2024 г.).
- [16] Zhang S., Zheng L., Tao W. Survey and Evaluation of RGB-D SLAM // IEEE Access. — 2021.
- [17] A pipeline framework for robot maze navigation using computer vision, path planning and communication protocols / A. Rodriguez-Tirado, D. Magallan-Ramirez, J.D. Martinez-Aguilar et al. // Developments in eSystems Engineering International Conference (DeSe). — 2020.