Научная статья

УДК 004.89
DOI 10.25205/1818-7900-2021-19-4-36-49

Исследование SLAM-фреймворков для монокулярных мобильных роботов в проекте Duckietown

Александра Денисовна Девятовская [1]
Никита Евгеньевич Бирючков [2]
Татьяна Викторовна Лях [3]
Константин Владимирович Чайка [4]

[1–3] Новосибирский государственный университет
Новосибирск, Россия

[4] Санкт-Петербургский государственный электротехнический университет
Санкт-Петербург, Россия

[4] Лаборатория алгоритмов мобильных роботов JetBrains Research
Санкт-Петербург, Россия

[1] a.devyatovskaya@g.nsu.ru, https://orcid.org/0000-0002-0583-4202
[2] n.biryuchkov@g.nsu.ru, https://orcid.org/0000-0002-1755-9527
[3] t.liakhv@g.nsu.ru, https://orcid.org/0000-0001-9148-946X
[4] pro100kot14@gmail.com, https://orcid.org/0000-0001-5778-9266

Аннотация

Статья посвящена оценке применимости SLAM фреймворков для задачи мобильных роботов проекта Duckietown. Проведен сравнительный анализ существующих SLAM алгоритмов и фреймворков, были отобраны фреймворки с учетом всех ограничений, накладываемых роботами проекта. Приведены практические результаты апробации фреймворка OpenVSLAM как на данных реального окружения Duckietown, так и на данных симулятора Duckietown.

Ключевые слова

Duckietown, low-cost платформа, SLAM, локализация, мобильные системы

Благодарности

Работа выполнена при поддержке JetBrains Research

Для цитирования

Девятовская А. Д., *Бирючков Н. Е.*, *Лях Т. В.*, *Чайка К. В.* Исследование SLAM-фреймворков для монокулярных мобильных роботов в проекте Duckietown // Вестник НГУ. Серия: Информационные технологии. 2021. Т. 19, № 4. С. 36–49. DOI 10.25205/1818-7900-2021-19-4-36-49

SLAM in Duckietown Simulator Using the OpenVSLAM Framework

Alexandra D. Devyatovskaya [1], **Nikita E. Biryuchkov** [2]
Tatyana V. Liakh [3], **Konstantin V. Chaika** [4]

[1–3] Novosibirsk State University
Novosibirsk, Russian Federation

[4] Saint Petersburg Electrotechnical University
St. Petersburg, Russian Federation

[4] Mobile Robot Algorithms Laboratory JetBrains Research
St. Petersburg, Russian Federation

[1] a.devyatovskaya@g.nsu.ru, https://orcid.org/0000-0002-0583-4202
[2] n.biryuchkov@g.nsu.ru, https://orcid.org/0000-0002-1755-9527
[3] t.liakhv@g.nsu.ru, https://orcid.org/0000-0001-9148-946X
[4] pro100kot14@gmail.com, https://orcid.org/0000-0001-5778-9266

Abstract

The article is devoted to evaluating the applicability of SLAM frameworks for the task of mobile robots of the Duckietown project. The paper provides a comparative analysis of existing SLAM algorithms and frameworks and selects frameworks taking into account all the constraints imposed by the project robots. The practical results of testing OpenVSLAM framework on the Duckietown environment and Duckietown simulator data are presented.

Keywords

Duckietown, low-cost platforms, SLAM, localization, mobile systems

Acknowledgements

The work was carried out with the support of JetBrains Research

For citation

Devyatovskaya A. D., Biryuchkov N. E., Lyakh T. V., Chaika K. V. SLAM in Duckietown Simulator Using the OpenVSLAM Framework. *Vestnik NSU. Series: Information Technologies*, 2021, vol. 19, no. 4, p. 36–49. (in Russ.) DOI 10.25205/1818-7900-2021-19-4-36-49

Введение

На сегодняшний день активно развивается исследовательский проект под названием «Duckietown» [1]. Это платформа, которая имитирует реальный город и используется для разработки и изучения алгоритмов управления автономными автомобилями. Он состоит из двух частей – роботов (duckiebots) и города (duckietown), по которому они передвигаются. В городе есть размеченные дороги, светофоры, дорожные знаки и препятствия. Тренажер отражает особенности индустриальных автопилотируемых систем с ограниченными ресурсами: роботы оснащены только камерами, а также имеют слабую вычислительную мощность. Разработчики используют ее для отладки и тестирования SLAM алгоритмов для промышленных роботов [1] и автопилотирования.

Алгоритмы построения карты местности и одновременного позиционирования (Simultaneous Localization And Mapping – SLAM) используются для решения задач локализации объекта и построения карты окружающей территории в сфере мобильной робототехники. SLAM выполняет минимизацию ошибок определения позиции камеры и вычисление инверсной глубины ключевых точек. За счет этого выравниваются траектория движения робота (камеры) и обозреваемое пространство, на основе которого формируется карта мира. Также технологии компьютерного зрения используются во многих областях промышленности и науки [2], в том числе и в робототехнике.

[1] Duckietown. URL: https://www.duckietown.org/ (дата обращения 31.03.2021).

SLAM алгоритмы – хорошо изученная область. Множество исследований описывают различные backend (TORO [3], ORB_SLAM [4], DSO_SLAM [5]) и frontend (Fast SLAM [2], DP-SLAM [3], Loop Closure [6]) SLAM [7] алгоритмов. Было проведено множество тестов для этих алгоритмов, их комбинаций в различных условиях освещенности, сложности рельефа местности, вычислительных мощностях роботов и типах камеры. Также, существует широкий спектр SLAM-фреймворков, таких как ISam [8], GMapping [9], OpenVSLAM [4], Cartographer [10], OV2Slam [5]. Сообществом проекта Duckietown также ведется разработка SLAM-фреймворков, таких как CSlam [6] и Lane-based SLAM [7].

Duckietown накладывает свои требования на используемые алгоритмы и фреймворки. Большинство разработчиков создают фреймворки для машин, использующих большое количество сенсоров (LIDAR [8], одометрия) и камер. Однако оснащение каждой машины или робота множеством датчиков может быть дорогостоящим для предпринимателя. Duckiebots имитируют промышленные low-cost автопилотируемые системы с ограниченными ресурсами: роботы оснащаются только камерами и имеют слабую вычислительную мощность. Данные особенности симулятора и его роботов накладывают ограничение на использование большинства стандартных алгоритмов и фреймворков SLAM. В результате задача SLAM в проекте Duckietown до сих пор не решена.

В статье был проведен анализ и апробация SLAM фреймворков для использования в тренажере duckiebots с учетом всех ограничений.

Анализ SLAM алгоритмов

Существует две большие группы SLAM алгоритмов – SLAM frontend и SLAM backend.

SLAM frontend решает следующие задачи.

1. Анализа и интеграции новых данных (data association). На этом этапе производится выделение особенностей (Feature/landmarks extraction) на вновь полученных данных. Особенности – это такие характеристики, которые легко могут быть выделены в среде и использоваться в дальнейшем для ориентации в пространстве.
2. Вычисления сдвига (Local Motion Estimation). Сдвиг положения выделенных особенностей на сцене можно определить, сопоставив набор особенностей, полученных на текущем шаге, с набором особенностей, полученных на предыдущем шаге. Так как особенности сами по себе стационарны, очевидно, что этот сдвиг – результат изменения положения камеры (робота). На основе этой информации можно выразить координаты камеры через систему линейных уравнений.
3. Обновления структуры (Features Integration), хранящей историю перемещений, где каждое состояние представляет собой глобальное положение робота и взаимное расположение обозреваемых особенностей на определенном промежутке времени. На этом этапе производится анализ, и полученные ранее данные добавляются в общую структуру для хранения информации о мире за все время исследования.

Существует множество frontend SLAM алгоритмов: ICP SLAM [11], Point Cloud SLAM [9], Fast Slam, DP_SLAM, Loop Closure. Для их сравнения были выбраны критерии: сложность алгоритма, устойчивость, погрешность (табл. 1). После анализа были отобраны три frontend

[2] FAST SLAM. URL: https://github.com/bushuhui/fastslamtps (дата обращения 31.03.2021).
[3] DP-SLAM. URL: https://users.cs.duke.edu/~parr/dpslam/ (дата обращения 31.03.2021).
[4] Tutorial Openvslam. URL: https://openvslam-community.readthedocs.io/en/latest/example.html (дата обращения 31.03.2021).
[5] OV2Slam. URL: https://github.com/ov2slam/ov2slam (дата обращения 31.03.2021).
[6] CSLAM. URL: https://github.com/duckietown/duckietown-cslam (дата обращения 31.03.2021).
[7] Lane-based SLAM. URL: https://github.com/mandanasmi/lane-slam (дата обращения 31.03.2021).
[8] LIDAR. URL: https://velodynelidar.com/what-is-lidar/ (дата обращения 31.03.2021).
[9] Point Cloud SLAM Overview. URL: https://www.mathworks.com/help/vision/ug/point-cloud-registration-workflow.html (дата обращения 31.03.2021).

алгоритма, которые могут быть использованы в проекте Duckietown, и проведен их сравнительный анализ. Были отброшены алгоритм ICP SLAM, так как он требует больших вычислительных мощностей, и алгоритм Point Cloud SLAM, так как требует дополнительных сенсоров.

Таблица 1

Сравнение SLAM frontend алгоритмов

Table 1

Comparison of SLAM frontend algorithms

Критерий	Loop Closure	DP_SLAM	Fast SLAM
Сложность	Логарифмическая	Логарифмическая	Логарифмическая
Устойчивость	Не может работать длительное время из-за сложности алгоритма и накапливающегося количества ошибок	Неточность при гладкой текстуре, не имеющей углов, появление шумов, может выдавать ошибки на мелких сильно удаленных объектах	Неточность на сложной карте и при обработке большого количества данных
Погрешность, %	~ 10	~ 5–10	~ 20

1. Fast Slam [10]

Этот алгоритм основывается на идее Мерфи. Во фреймворке используется фильтр частиц Рао – Блэквелла для построения гипотез о текущем положении робота и фильтр Калмана для отслеживания положений наперед заданных меток. Одной из проблем при составлении карты является возможная идентичность встречающихся на карте объектов. Чтобы более точно определять местоположение робота и пространство вокруг него, в рассматриваемом методе вводятся ориентиры (метки), которые уникальны для каждого объекта на карте и позволяют различать местность. В Fast SLAM одна большая карта рассматривается как совокупность локальных подкарт, что позволяет убрать зависимость ориентиров друг от друга и таким образом значительно сократить время пересчета оценки состояния системы.

Сложность: O(MlogK), где M – число частиц в фильтре частиц, а K – число меток.

2. DP_Slam [11]

DP-SLAM также основан на идее Мерфи. Он тоже использует фильтры частиц, однако не требует каких-либо предопределенных меток. Проблема сопоставления реальных объектов на карте с собранными данными в этом подходе решается путем хранения в памяти множества промежуточных карт. Карта представляется в виде сетки с заполнением ячеек, занятых препятствиями. Хранить такую карту удобно в виде массива, где элементы, отражающие положение препятствий, имеют значение 1, а все остальные – 0. Таким образом, обе части задачи – локализация и составление карты – решаются одним и тем же фильтром частиц. Карты хранятся в памяти в специальном виде, называемом «распределенные частицы» (distributed particle, DP), что позволяет управлять сотнями и тысячами вероятных карт-кандидатов и вероятных позиций одновременно. Алгоритм DP-SLAM достаточно прост в реализации, поскольку он не использует дополнительных эвристических методов разрешения циклов. Кроме того, он обладает высокой производительностью. Алгоритм в наихудшем случае про-

[10] FAST SLAM. URL: https://github.com/bushuhui/fastslamtps (дата обращения 31.03.2021).
[11] DP-SLAM. URL: https://users.cs.duke.edu/~parr/dpslam/ (дата обращения 31.03.2021).

демонстрировал лого квадратичную сложность в зависимости от количества отсчетов и линейную сложность в зависимости от обозреваемой лазерным сенсором площади.

3. Loop Closure [6]

Loop closure – обнаружение петель. Обособленно от задач SLAM стоит вопрос, как отслеживать ситуации, когда робот возвращается туда, где уже побывал. Одно из решений – так называемая Bags of Binary Words. Каждому кадру ставится в соответствие дескриптор (BRIEF descriptor, вычисляемый на основе визуальных особенностей изображения). Для хранения информации об изображениях на основе данных для обучения формируется словарь в виде дерева, содержащий «слова» для представления дескрипторов и их веса (отражающих, насколько часто они встречались в наборе изображений для обучения). Для формирования «слова» производится поиск визуальных особенностей на основе данных для тренинга и их последующая группировка (с помощью метода k-среднего). Каждый последующий уровень дерева получается путем повторения данной операции с дескриптором родительского узла. В результате при анализе новых поступивших данных производится быстрое (дистанция Хэмминга) разложение идентификатора текущего кадра в линейную комбинацию узлов дерева, где в роли коэффициентов выступают их веса – вероятности.

SLAM backend решает следующие задачи.

1. Сравнения данных локального сдвига и особенностей. С одной стороны, у нас есть результаты локального сдвига, на основании которых мы можем вычислить новое положение робота. С другой – мы также можем определить позицию робота относительно особенностей сцены. В силу погрешностей датчиков и алгоритмов определения местоположения при попытке совместить подсчитанное разными методами расположение робота может возникнуть погрешность. Таким образом, существует вероятность возникновения погрешности в определении локального положения.
2. Обновления общего представления о карте мира и траектории робота (Global Optimization). На основе вновь полученных данных производится уточнение текущих представлений о мире – пересчет позиций робота (state estimation), особенностей и вероятность появления их на карте с учетом погрешности. После всех вычислений обновляется общее представление мира – местоположение робота (state update) и координаты особенностей (landmark update).

Также существует множество backend slam алгоритмов: SIFT SLAM [12], TORO, ORB_SLAM, DSO_SLAM. Для их сравнения были выбраны критерии: сложность алгоритма, устойчивость, погрешность, оптимальные условия применения и сфера применимости (табл. 2). После анализа были отобраны три backend алгоритма, которые могут быть использованы на тренажерах Duckietown, и проведен их сравнительный анализ. SIFT SLAM был отброшен, так как требует мощной видеокарты.

1. TORO [3]

Данный метод определяет конфигурация графа, при которой вероятность наблюдаемых особенностей будет максимальна. Подходы к решению подобных задач –Sparse Bundle Adjustments, Structure from motion, Visual SLAM. В основу TORO лег метод стохастического градиентного спуска – (SGD – stochastic gradient descent). Суть данного метода в последовательной оптимизации всего графа для каждого ограничения (взаимного расположения особенностей и робота). Для объединения данных оптимизации с нескольких ограничений используются специальные коэффициенты для корректировки значений остатка. Ключевое достижение метода – представление данных для SGD – в TORO используется дерево параметров в качестве индексов для линейного уравнения поз робота.

Таблица 2

Сравнение SLAM backend алгоритмов

Table 2

Comparison of SLAM backend algorithms

Критерий	TORO	ORB_SLAM	DSO_SLAM
Сложность	Линейная	Константная	Линейная
Устойчивость	Устойчивость к плохим начальным конфигурациям за счет минимизации ошибок на основе градиентного спуска	Неточность при гладкой текстуре, не имеющей углов, появление шумов, может выдавать ошибки на мелких сильно удаленных объектах	Необходимость хорошей инициализации; невысокая скорость, за счет которой идет более точная картинка
Погрешность, %	~ 5–15	~ 15	~ 10–15
Оптимальные условия применения	Хорошо работает как с большими, так и с мелкими объектами на сцене, хуже работает при плохой освещенности	Monocular, Stereo or RGB-D cameras, желательна хорошая освещенность	Желательна хорошая освещенность, но неплохо работает и при плохой освещенности за счет удаления факторов погрешности от геометрических и фотометрических перспектив
Сфера применимости	3D-реконструкция улиц, дорог, внутренней планировки домов	3D-реконструкция сцены в самых разнообразных средах, начиная от небольших лабораторных карт и заканчивая автомобилем, объезжающим несколько городских кварталов	3D-реконструкция улиц, дорог, внутренней планировки домов

2. ORB_SLAM [4]

Это непрямой метод, базирующийся на поиске ключевых точек с помощью FAST детектора и поиска ориентированных дескрипторов. Он производится в несколько этапов: отслеживание, построение локальной карты, loop closure, релокализация, глобальная оптимизация. Этот метод достаточно эффективен, однако, поскольку он использует детектор углов, то дает недостаточно точные результаты, несмотря на хорошо реализованный алгоритм замыкания петель. Плюсы: быстродействие; легко удаляются шумы. Недостатками являются неточность при гладкой текстуре, не имеющей углов; использование малой части информации изображений.

3. DSO_SLAM [5]

Метод SLAM использует прямую одометрию и создает разреженную карту местности. Создание разреженной карты делает его эффективнее (LSD SLAM), так как повышает быстродействие и уменьшает аппаратные расходы. Метод требует достаточно серьезной начальной инициализации, геометрическую и фотометрическую калибровку. Поиск ключевых то-

чек ведется по наибольшему градиенту, применяется фотометрическая коррекция изображения. К достоинствам DSO_SLAM, можно отнести использование более полной информации о изображении, в связи с чем точное нахождение ключевых точек и построение карты. К недостаткам – необходимость хорошей инициализации, невысокую скорость, однако возможность распараллеливания.

Анализ SLAM-фреймворков

SLAM алгоритмы являются базовыми схемами действия для решения определенных SLAM задач. На их базе создаются SLAM-фреймворки – готовые программы или библиотеки, которые могут включать в себя несколько SLAM алгоритмов или их улучшенные под конкретный фреймворк модификации.

Проведенный анализ сравнения SLAM backend и fronted алгоритмов повлиял на выбор SLAM-фреймворков для изучения. В конечную выборку для изучения применимости на роботах проекта Duckitown были включены только те фреймворки, которые основываются на отобранных алгоритмах. Во время составления выборки были рассмотрены такие SLAM-фреймворки, как GMapping, KartoSLAM [13], HectorSLAM [14], LagoSLAM [15], CoreSLAM [16], VSLAM, Hector_mapping.

После исследования были отобраны и проанализированы пять SLAM-фреймворков, которые подходят по используемым SLAM алгоритмам, активно использовались в исследовательских проектах и доказали свою работоспособность [17] (табл. 3).

1. GMapping [9]

Библиотека, описанная и поддерживаемая в официальной документации ROS. Использует высокоэффективный Rao-Blackwellized фильтр частиц для изучения сеточных карт по данным лазерного диапазона. Этот подход использует фильтр частиц, в котором каждая частица несет индивидуальную карту окружающей среды. Соответственно, ключевой вопрос состоит в том, как уменьшить количество частиц. В этом SLAM алгоритме используются адаптивные методы уменьшения числа частиц в фильтре частиц с Rao- Blackwellized для изучения сеточных карт. Также в нем предлагается подход к вычислению точного распределения предложений, учитывающий не только движение робота, но и самые последние наблюдения. Это резко уменьшает неопределенность относительно позы робота на этапе прогнозирования фильтра.

2. VSLAM [12]

Библиотека, поддерживаемая ROS. В официальной документации описана как экспериментальная. Компоненты библиотеки:

- обнаружение / сопоставление признаков;
- поиск ключевых точек на изображении;
- сопоставление с ключевыми точками на других изображениях;
- visual odometry;
- крупномасштабная оптимизация по 3D точечным позициям и позициям камер;
- Loop closure;
- поиск совпадений между текущим кадром и набором предыдущих кадров.

[12] VSLAM. URL: http://wiki.ros.org/vslam (дата обращения 31.03.2021).

Таблица 3

Сравнение SLAM-фреймворков

Table 3

Comparison of SLAM frameworks

Критерий	GMapping	VSLAM	Hector-Mapping	OpenVSLAM	Cartographer_ros
Требования	Linux, GCC 3.3/4.0.x, odometry sensors, laser range-finder	Camera, Linux/Unix	LIDAR systems, Linux/Unix	Linux/Unix, Camera	64-bit, modern CPU, 16 GB RAM, Ubuntu 18.04, 20.04, gcc version 7.5.0, 9.3.0
Возможность использования вместе с IMU	+	–	+	–	+
Возможность использования вместе с датчиками одометрии	+	+	–	–	+
Использует	Rao-Blackwellized фильтр частиц, visual odometry	Visual odometry, Sparse bundle adjustment, Loop closure	Gauss-Newton Approach	Gauss-Newton Approach, ORB_SLAM, Proclaim и UcoSLAM	Loop closure,Visual odometry
Где используется	Мобильные роботы – платформы, обязательно наличие камеры	Мобильные роботы – платформы, обязательно наличие камеры	Беспилотные наземные роботы, транспортные средства, портативные картографические устройства и регистрируемые данные с БПЛА	Мобильные роботы – платформы, обязательно наличие камеры	Мобильные роботы – платформы

3. Hector-mapping [13]

Библиотека, описанная и поддерживаемая в официальной документации ROS. Это подход SLAM, который может быть использован без одометрии. Он использует высокую частоту обновления современных лидарных систем, таких как Hokuyo UTM-30LX, и обеспечивает 2D-оценку позиции при высокой скорости сканирования датчиков (40 Гц для UTM-30LX). Хотя система не обеспечивает явной возможности замыкания цикла, она достаточно точна для многих реальных сценариев. Система успешно применяется на беспилотных наземных роботах, беспилотных наземных транспортных средствах, портативных картографических устройствах и регистрируемых данных с БПЛА. Hector SLAM полагается на сопоставление сканирования, использует подход Гаусса – Ньютона и достаточно точен, чтобы не требовать loop closure и визуальной одометрии.

4. OpenVSLAM [14]

Это монокулярная, стерео и визуальная система RGB-D Slam. Примечательными особенностями являются:

- фреймворк совместим с различными типами моделей камер и может быть легко настроен для других моделей камер. При необходимости пользователи могут легко реализовать компоненты для работы с различными моделями камер (например, dual fisheye, catadioptric);
- созданные карты можно хранить и загружать, а затем OpenVSLAM может локализовать новые изображения на основе готовых карт;
- система полностью модульная. OpenVSLAM разработан путем инкапсуляции нескольких функций в отдельные компоненты с помощью простых для понимания API;
- OpenVSLAM основан на indirect SLAM алгоритме с дополнительными функциями, такими как ORB_SLAM, Proclaim и UcoSLAM.

5. Cartographer_ros [10]

Cartographer – это система, которая обеспечивает одновременную локализацию и создание карт в реальном времени в формате 2D и 3D на нескольких платформах и конфигурациях датчиков. Портативные лазерные дальномеры и синхронная локализация, создание карт – эффективный метод получения готовых карт этажей зданий. Создание и визуализация поэтажных планов в реальном времени помогает оценить качество и охват данных.

Duckietown на данный момент поддерживает собственные SLAM-фреймворки, такие как CSlam и Lane-based SLAM. Они работают очень медленно, требуют специального внешнего оборудования и не подходят для использования на больших площадях, например в складах или большой площади умного города.

1. CSlam

Требуемое оборудование:

- Duckiebot;
- Watch Tower для наблюдения за полем;
- AprilTag знаки.

Цель cSLAM – объединить наблюдения сторожевых башен и duckiebots в единую систему оптимизаций, которая пытается максимально точно предсказать текущее и прошлое местоположение duckiebots. Затем это визуализируется на 3D-модели города. Плюсы: модульность, возможность расширения (больше роботов, больше территория города), относительно удобный интерфейс, функция выявления ошибок. Минусы: зависимость от освещения и погодных условий, все AprilTag должны быть хорошо различимы, погрешность 22 %.

[13] Hecor_mapping. URL: http://wiki.ros.org/hector_mapping (дата обращения 31.03.2021).

[14] Tutorial Openvslam. URL: https://openvslam-community.readthedocs.io/en/latest/example.html (дата обращения 31.03.2021).

2. Lane-based SLAM [15]

Идея – использовать семантические изображения с камеры «уткобота». Линии распознаются как часть дороги. Есть три различных типа линий: желтые линии в центре дороги, белые линии, расположенные по бокам дороги, и красные линии, которые являются стоплиниями. Затем их используют в качестве предварительного знания о «дакитауне». Используется одометрия для оценки положения позиции робота. В основе – фильтр spurious lines. Комбинация одометрии и обнаружения линий. Используемые модули: Line detector module, Line descriptor module, Ground projection module, Line sanity module, Odometry module, Show map module. Плюсы: модульность, совместимость с ROS, визуализация карты с цветом линий. Минусы: не проработан до конца, может давать большую погрешность (30 %)

По результатам проведенного анализа были отобраны те фреймворки, которые позволяют использовать в качестве датчика монокулярную камеру, расположенную на Duckiebot и не требуют дополнительных датчиков. Из приведенных выше фреймворков этим требованиям соответствуют VSLAM и OpenVSLAM. После более детального изучения этих двух фреймворков было решено тестировать только OpenVSLAM, так как разработка VSLAM-фреймворка была заброшена.

Тестирование OpenVSLAM на стандартном видео dataset Duckietown

В эксперименте фреймворк настроен в соответствии с конфигурацией камеры Raspberry pi. Были выставлены настройки: --frame-skip arg (=1) – интервал пропуска кадров, --no-sleep – не ожидать следующий кадр в режиме реального времени, частота кадров в секунду 30 fps, разрешение кадров 1920×1080. На вход подавался видеодатасет, записанный с камеры duckiebot. Робот двигался по карте спроектированного города и ориентировался по специальной разметке, нанесенной на объекты.

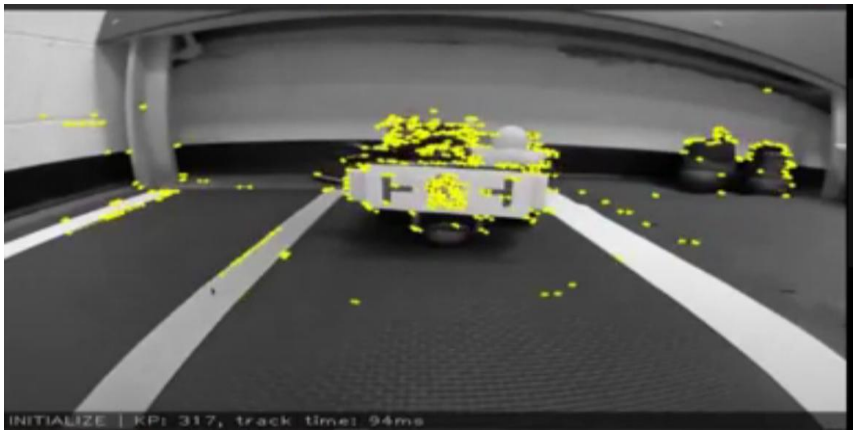

Рис. 1. Результат выставления меток на карте Duckietown
Fig. 1 Result of marking on the Duckietown map

В ходе эксперимента было установлено, что фреймворк с высокой точностью способен расставлять метки, обозначенные желтым цветом, на кадрах видеопотока. Результат работы OpenVSLAM представлен на рис. 1. При этом алгоритму не удалось построить карту местности, по которой двигался робот. При каждом резком повороте камеры Raspberry pi, фреймворк сбрасывал часть карты, которая была построена, и начинал строить маршрут сначала.

[15] Lane-based SLAM. URL: https://github.com/mandanasmi/lane-slam (дата обращения 31.03.2021).

Для устранения отказа в работе OpenVSLAM, настройки, указанные выше, варьировались в соответствии с возможными режимами. Флаг --no-sleep был отключен. В этом случае алгоритм также сбрасывал карту через каждые 3 секунды.

Тестирование OpenVSLAM на стандартном фото dataset Duckietown

В эксперименте видеодатасет был отформатирован и разделен на набор фотографий. Настройки OpenVSLAM выставлены в соответствии с камерой Raspberry pi, частота кадров 30 fps.

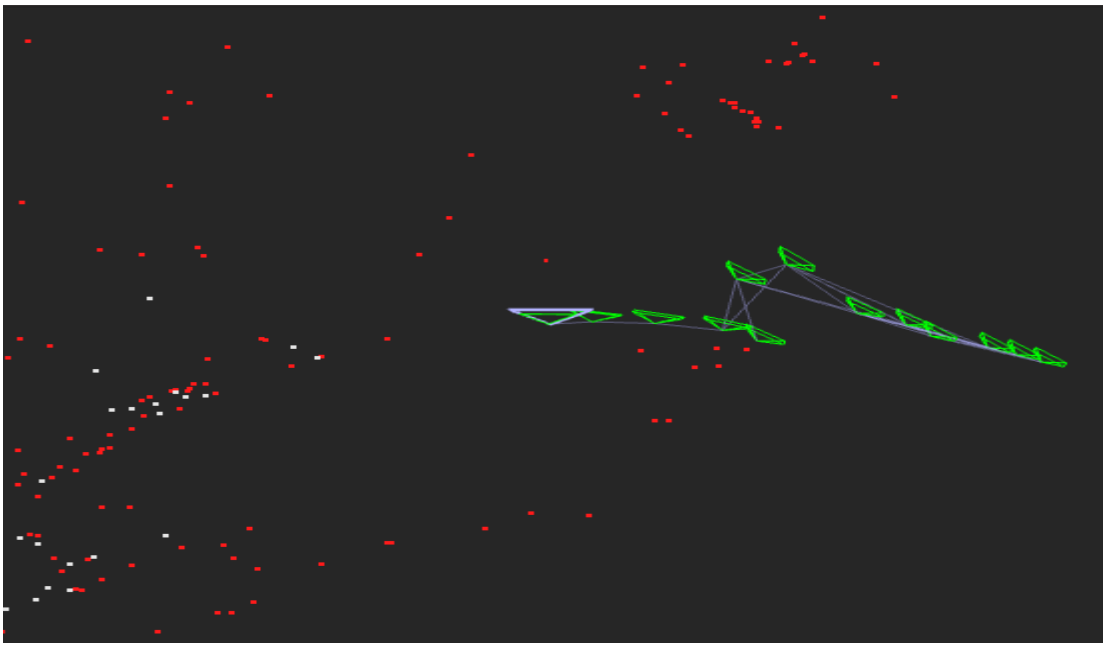

Рис. 2. Результат отрисовки карты фреймворком
Fig. 2. The result of the framework's rendering of the map

В результате работы фреймворка была построена виртуальная карта (рис. 2), повторяющая траекторию движения мобильного робота по местности Duckietown. Итоговая траектория отрисована зелеными треугольниками. Также алгоритм с высокой точностью определял свое текущее положение и выставлял соответствующую метку, обозначенную синим цветом. Каждая красная точка, выставленная OpenVSLAM, отображает, объект, находящийся перед камерой робота.

В ходе выполнения алгоритма был определен замкнутый цикл в маршруте duckiebot.

Апробация OpenVSLAM на данных симулятора Duckietown

В эксперименте был использован симулятор окружения Duckietown, который входит в перечень стандартных инструментов для работы с мобильными роботами проекта. Для работы с симулятором выбран скрипт, написанный на языке python. После запуска симулятора в стандартном режиме был записан видеопоток с виртуальной камеры duckiebot. С учетом предыдущих экспериментов Dataset разделен на набор фотографий, которые подавались на вход OpenVSLAM. Настройки, как и в предыдущих экспериментах, установлены в соответствии с конфигурацией камеры Raspberry pi.

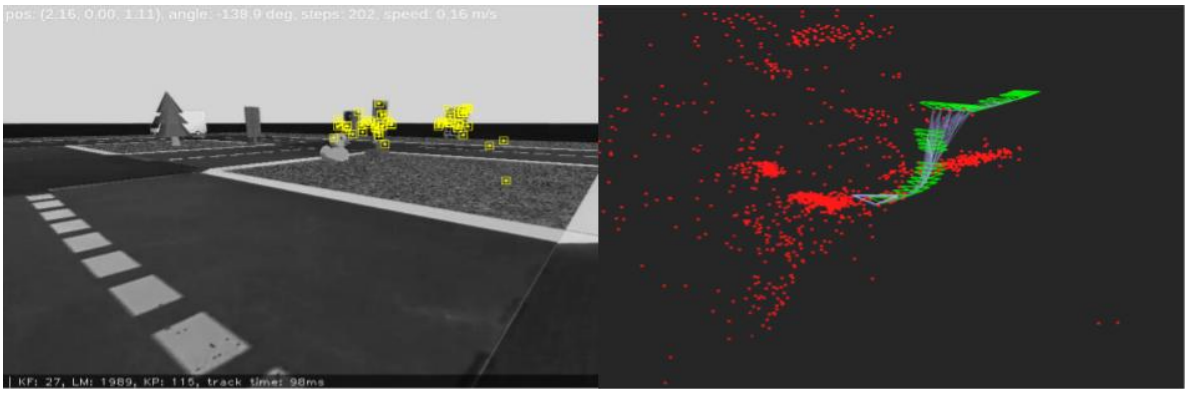

Рис. 3. Результат выставления меток на карте имитатора Duckietown и отрисовки карты фреймворком
Fig. 3. The result of marking on the Duckietown imitator map and of the framework's rendering of the map

В результате выполнения алгоритма OpenVSLAM расставил метки на каждом из объектов имитационного города Duckietown (рис. 3). Также была построена карта виртуальной местности уткогорода. Фреймворк с высокой точностью указывал положение препятствий на маршруте, а также свое местоположение.

Заключение

В статье проведено сравнительное исследование алгоритмов, использующихся для локализации мобильных роботов, оснащенных одной только монокулярной камерой и имеющих малые вычислительные мощности.

С учетом ограничений, которые накладывает используемое оборудование, установлено, что поставленным требованиям удовлетворяет фреймворк OpenVSLAM. Разработан алгоритм настройки фреймворка OpenVSLAM для работы с датасетами с Duckietown. Фреймворк был апробирован на стандартном видео, снятом в помещении, на видеодатасетах, полученных с официального сайта, и имитатора Duckietown. Лучший результат фреймворк OpenVSLAM показал на наборе фотографий, полученных с имитатора Duckietown.

В дальнейшей работе планируется адаптировать OpenVSLAM для работы на роботах проекта в режиме реального времени.

Список литературы

1. **Лях Т. В., Зюбин В. Е., Гаранина Н. О.** Автоматизированная верификация алгоритмов управления сложными технологическими объектами на программных имитаторах // Вестник НГУ. Серия: Информационные технологии, 2018. Т. 16, № 4. С. 85–94.
2. **Малышев А. Г., Полыгалов А. С., Алямкин С. А.** Автоматическое тегирование изображений одежды // Вестник НГУ. Серия: Информационные технологии. 2020. Т. 18, № 2. С. 54–61. DOI 10.25205/1818-7900-2020-18-2-54-61
3. **Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard.** Non-linear Constraint Network Optimization for Efficient Map Learning. *IEEE Transactions on Intelligent Transportation Systems*, 2009, vol. 10, no. 3, pp. 428–439.
4. **Raúl Mur-Artal, and Juan D. Tardós.** ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. In: ArXiv preprint arXiv:1610.06475, 2016.
5. **Jakob Engel, Prof. Vladlen Koltun, Prof. Daniel Cremers.** DSO: Direct Sparse Odometry. In: ArXiv preprint arXiv:1610.06475, 2017.

6. **Kin Leong Ho, Paul Newman.** Loop closure detection in SLAM by combining visual and spatial appearance. *Robotics Auton. Syst*., 2006, vol. 54, pp. 740–749.
7. **Baichuan Huang, Jun Zhao, Jingbin Liu**. A Survey of Simultaneous Localization and Mapping with an Envision in 6G Wireless Networks. In: arXiv:1909.05214v2, 2019.
8. **Michael Kaess, Ananth Ranganathan and Frank Dellaert**. iSAM: Incremental Smoothing and Mapping. *IEEE Transactions on Robotics (TRO)*, 2008, vol. 24, no. 6, pp. 1365–1378.
9. **Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard.** Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 2007, vol. 23, pp. 34–46.
10. **Hess W., Kohler D., Rapp H., and Andor D.** Real-Time Loop Closure in 2D LIDAR SLAM. In: Robotics and Automation (ICRA), IEEE International Conference, 2016, pp. 1271–1278.
11. **Tiar R., Lakrouf M., and Azouaoui O.** Fast ICP-SLAM for a Bi-Steerable Mobile Robot in Large Environments. In: IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), 2015, pp. 1–6.
12. **Tonghui Wang, Guoyun Lv, Shikai Wan, gHaili Li, Baicen Lu** SIFT Based Monocular SLAM with GPU Accelerated. CHINACOM Springer, 2018. DOI 10.1007/978-3-319-78139-6_2
13. **Kurt Konolige, Giorgio Grisetti, Rainer Kummerle, Wolfram Burgard, Benson Limketkai, and Regis Vincent.** Efficient sparse pose adjustment for 2d mapping. In: IEEE / RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 22–29.
14. **Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and UweKlingauf**. A flexible and scalable slam system with full 3d motion estimation. In: IEEE International Symposium on Safety, Security, and Rescue Robotics, 2011, pp. 155–160.
15. **Luca Carlone, Rosario Aragues, Jos ´e A Castellanos, and Basilio Bona.** A linear approximation for graph-based simultaneous localization and mapping. *Robotics: Science and Systems*, 2012, no. 7, pp. 41–48.
16. **Steux B. and Hamzaoui O.** A slam algorithm in less than 200 lines c-language program. In: Proc. of the Control AutomationRobotics & Vision (ICARCV). Singapore, 2010, pp. 7–10.
17. **Baichuan Huang, Jun Zhao, Jingbin Liu** A Survey of Simultaneous Localization and Mapping. In: ArXiv:abs/1909.05214, 2019.

References

1. **Lyakh T. V., Zyubin V. E., Garanina N. O.** Automated verification of control algorithms for complex technological objects on software simulators. *Vestnik NSU. Series: Information Technology*, 2018, vol. 16, no. 4, pp. 85–94. (in Russ.)
2. **Malyshev A. G., Polygalov A. S., Alyamkin S. A.** Automatic tagging of clothing images. *Vestnik NSU. Series: Information Technology*, 2020, vol. 18, no. 2, pp. 54–61. DOI 10.25205/1818-7900-2020-18-2-54-61
3. **Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard.** Non-linear Constraint Network Optimization for Efficient Map Learning. *IEEE Transactions on Intelligent Transportation Systems*, 2009, vol. 10, no. 3, pp. 428–439.
4. **Raúl Mur-Artal, and Juan D. Tardós.** ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. In: ArXiv preprint arXiv:1610.06475, 2016.
5. **Jakob Engel, Prof. Vladlen Koltun, Prof. Daniel Cremers.** DSO: Direct Sparse Odometry. In: ArXiv preprint arXiv:1610.06475, 2017.
6. **Kin Leong Ho, Paul Newman.** Loop closure detection in SLAM by combining visual and spatial appearance. *Robotics Auton. Syst*., 2006, vol. 54, pp. 740–749.
7. **Baichuan Huang, Jun Zhao, Jingbin Liu**. A Survey of Simultaneous Localization and Mapping with an Envision in 6G Wireless Networks. In: arXiv:1909.05214v2, 2019.

8. **Michael Kaess, Ananth Ranganathan and Frank Dellaert**. iSAM: Incremental Smoothing and Mapping. *IEEE Transactions on Robotics (TRO)*, 2008, vol. 24, no. 6, pp. 1365–1378.
9. **Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard.** Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 2007, vol. 23, pp. 34–46.
10. **Hess W., Kohler D., Rapp H., and Andor D.** Real-Time Loop Closure in 2D LIDAR SLAM. In: Robotics and Automation (ICRA), IEEE International Conference, 2016, pp. 1271–1278.
11. **Tiar R., Lakrouf M., and Azouaoui O.** Fast ICP-SLAM for a Bi-Steerable Mobile Robot in Large Environments. In: IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), 2015, pp. 1–6.
12. **Tonghui Wang, Guoyun Lv, Shikai Wan, gHaili Li, Baicen Lu** SIFT Based Monocular SLAM with GPU Accelerated. CHINACOM Springer, 2018. DOI 10.1007/978-3-319-78139-6_2
13. **Kurt Konolige, Giorgio Grisetti, Rainer Kummerle, Wolfram Burgard, Benson Limketkai, and Regis Vincent.** Efficient sparse pose adjustment for 2d mapping. In: IEEE / RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 22–29.
14. **Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and UweKlingauf**. A flexible and scalable slam system with full 3d motion estimation. In: IEEE International Symposium on Safety, Security, and Rescue Robotics, 2011, pp. 155–160.
15. **Luca Carlone, Rosario Aragues, Jos ´e A Castellanos, and Basilio Bona.** A linear approximation for graph-based simultaneous localization and mapping. *Robotics: Science and Systems*, 2012, no. 7, pp. 41–48.
16. **Steux B. and Hamzaoui O.** A slam algorithm in less than 200 lines c-language program. In: Proc. of the Control AutomationRobotics & Vision (ICARCV). Singapore, 2010, pp. 7–10.
17. **Baichuan Huang, Jun Zhao, Jingbin Liu** A Survey of Simultaneous Localization and Mapping. In: ArXiv:abs/1909.05214, 2019.

Информация об авторах

Александра Денисовна Девятовская, студентка
Никита Евгеньевич Бирючков, студент
Татьяна Викторовна Лях, кандидат технических наук
Константин Владимирович Чайка, аспирант

Information about the Authors

Alexandra D. Devyatovskaya, Bachelor Student
Nikita E. Biryuchkov, Bachelor Student
Tatyana V. Liakh, Candidate of Sciences (Engineering)
Konstantin V. Chaika, Post-Graduate Student

Статья поступила в редакцию 11.09.2021;
одобрена после рецензирования 01.12.2021; *принята к публикации 01.12.2021*
The article was submitted 11.09.2021;
approved after reviewing 01.12.2021; *accepted for publication 01.12.2021*