

УДК 004.94

Разработка симулятора и алгоритмов обработки облаков точек для систем компьютерного зрения мобильных роботов

Рогалева Юлия Александровна

Козов Алексей Владимирович (научный руководитель)

МГТУ им. Н.Э. Баумана, Москва, Россия

Рассмотрена разработка легковесного симулятора лазерного дальномера (лидара) и набора алгоритмов обработки облаков точек, предназначенных для использования в системах компьютерного зрения мобильных роботов. Предложены математические модели сканирования, обработки облаков точек и геометрической классификации препятствий, реализована модульная архитектура на языке программирования C++ с использованием ROS2 и библиотеки PCL. Созданный симулятор и реализованные алгоритмы могут использоваться для тестирования и отладки систем восприятия в робототехнике в условиях, где доступ к реальному оборудованию ограничен, затруднен или небезопасен.

Ключевые слова: лидар, облако точек, симуляция, ROS2, классификация, фильтрация, системы компьютерного зрения

Введение. В мобильной робототехнике компьютерное зрение необходимо для навигации роботов. Основным источником данных для них являются трехмерные облака точек с лидаров, которые помогают роботам распознавать объекты, избегать препятствия и анализировать окружение. Однако обработка этих данных требует мощных вычислений и сложных алгоритмов, а тестировать их на реальных роботах дорого и рискованно. Это обуславливает актуальность разработки симуляторов, позволяющих безопасно генерировать и проверять облака точек в виртуальной среде [1]. Существующие решения, такие как Gazebo, Isaac Sim, AirSim, либо обладают высокой ресурсоемкостью, либо избыточны для небольших исследовательских и учебных проектов. В связи с этим поставлена задача создать легковесный симулятор, реализующий базовую геометрию лазерного сканирования, а также алгоритмы фильтрации и классификации, пригодные для использования в рамках программного каркаса и среды выполнения ROS и ROS2 [2, 3].

Математическое описание лидара и облака точек. Лидар моделируется как источник упорядоченного множества лучей, направленных из фиксированной точки [4]. Направляющий вектор луча в локальной системе координат лидара вычисляется по формуле

$$\vec{r}_{ij} = \begin{bmatrix} \cos\alpha_i \cdot \cos\varphi_j \\ \cos\alpha_i \cdot \sin\varphi_j \\ \sin\alpha_i \end{bmatrix},$$

где $\alpha_i = \alpha_{\text{begin}} + i(\alpha_{\text{end}} - \alpha_{\text{begin}})/(N - 1)$ — вертикальный угол сканирования, находящийся в пределах от α_{begin} до α_{end} , $i = 0, 1, \dots, N - 1$; $\varphi_j = j\Delta\varphi$ — горизонтальный угол с шагом дискретизации $\Delta\varphi$, $j = 0, 1, \dots, 2\pi/\Delta\varphi$.

В случае, если сенсор повернут относительно глобальной системы координат, направляющий вектор луча должен быть дополнительно преобразован с использованием матрицы поворота $\mathbf{M}$, соответствующей текущей ориентации сенсора. Это позволяет учесть наклон или поворот лидара при моделировании направления лучей

$$\vec{d}_{ij} = \mathbf{M} \cdot \vec{r}_{ij}.$$

Задача сканирования заключается в нахождении пересечений каждого луча с объектами сцены. Сцена моделируется в виде конечного множества объектов $\mathbf{O} = \{O_1, O_2, \dots, O_M\}$, каждый из которых относится к одному из поддерживаемых типов: сфера, плоскость или цилиндр. В таблице представлены стандартные уравнения и условия пересечения луча лидара с ними.

Определение пересечений луча лидара с различными объектами сцены

Объект	Формула пересечения	Условия валидности
Сфера	$at^2 + bt + c = 0$, где $a = \vec{d}_{ij} \cdot \vec{d}_{ij}$, $b = 2(\vec{o} - \vec{c}) \cdot \vec{d}_{ij}$, $c = \vec{o} - \vec{c} ^2 - R^2$ $\vec{c}$ — центр сферы, R — радиус	$t > 0, t < R_{\max}$ $\Delta = b^2 - 4ac \geq 0$
Плоскость	$t = \frac{-\vec{n} \cdot \vec{o} + d}{\vec{n} \cdot \vec{d}_{ij}}$, $d = -\vec{n} \cdot \vec{p}_0$ $\vec{n}$ — нормаль, $\vec{p}_0$ — точка на плоскости	$t > 0, t < R_{\max}$
Цилиндр (по Z)	$at^2 + bt + c = 0$, где $a = d_x^2 + d_y^2$, $b = 2(o_x d_x + o_y d_y)$, $c = o_x^2 + o_y^2 - R^2$ R — радиус, Z — ось — высота	$t > 0, t < R_{\max}$ $z = o_z + t \cdot d_z$ $\in [z_{\min}, z_{\max}]$

Из всех допустимых пересечений выбирается ближайшее значение параметра $t_{\min} \in (0, R_{\max})$. Если пересечение найдено, точка в пространстве определяется как $\vec{p}_{ij} = \vec{o} + t_{\min} \cdot \vec{d}_{ij}$. Результирующее облако точек формируется как множество таких точек

$$\mathbf{P} = \{\vec{p}_k \in \mathbf{R}^3 \mid \exists t_{\min}: t_{\min} < R_{\max}\}.$$

Обработка облака точек и классификация препятствий. В случае использования нескольких лидаров, каждый из которых формирует собственное облако $\mathbf{P}_i$, необходимо выполнить этап предварительной фильтрации и объединения. Это позволяет уменьшить плотность облака, удалить незначимую

информацию, а также привести все данные к общей системе координат. Каждое облако P_i от i -го лидара преобразуется в разреженное $\widetilde{P}_i$ с использованием воксельной фильтрации [5] путем разбиения пространства на кубическую сетку с фиксированным шагом (длиной ребра вокселя) $l > 0$ и выборе из каждой ячейки лишь одной репрезентативной точки. Это позволяет существенно сократить количество точек без значительной потери информации о геометрии сцены. Математически это записывается как

$$\widetilde{P}_i = \{\vec{p} \in P_i, \vec{p} \in \text{центры вокселей с длиной ребра } l\}.$$

После воксельной фильтрации выполняется удаление плоскости пола [6], чтобы исключить точки, не представляющие интереса при анализе окружающей сцены. Пол аппроксимируется плоскостью с нормалью $\vec{n}$ и смещением d , а точки, принадлежащие этой плоскости, отбрасываются из облака по правилу

$$\widehat{P}_i = \widetilde{P}_i \setminus \{\vec{p} \in \widetilde{P}_i, |\vec{n} \cdot \vec{p} + d| < \varepsilon\},$$

где ε — допустимая погрешность, определяющая, насколько близко точка должна находиться к плоскости, чтобы считаться принадлежащей ей.

Если используется несколько лидаров, каждое облако $\widehat{P}_i$, формируемое i -м устройством приводится к глобальной системе с помощью преобразования T_i :

$$P_{\text{merged}} = T_1 \cdot \widehat{P}_1 \cup T_2 \cdot \widehat{P}_2 \cup \dots \cup T_N \cdot \widehat{P}_N.$$

Каждая точка облака анализируется индивидуально. Отбрасываются точки, не соответствующие условиям

$$\sqrt{x_k^2 + y_k^2} \leq D_{\text{max}}, \quad d_k \geq d_{\text{min}},$$

где D_{max} — максимальное расстояние от сенсора; d_{min} — минимально допустимая интенсивность. Это позволяет удалить как слишком дальние шумовые точки, так и случайные слабые сигналы [7].

Для задачи классификации [8] на плоскость Oxy накладывается регулярная квадратная сетка с шагом l . Каждая ячейка сетки обозначается индексами (i, j) , где $i = x_k/l$, $j = y_k/l$; x_k , y_k — координаты точки $\vec{p}_k = (x_k, y_k, z_k, d_k) \in P_{\text{merged}}$. В каждой ячейке $C_{(i,j)}$ вычисляются ее параметры:

- высота объекта в ячейке $h^{(i,j)} = z_{\text{max}}^{(i,j)} - z_{\text{min}}^{(i,j)}$;
- координаты центра $(x_c^{(i,j)}; y_c^{(i,j)})$, где $x_c = \frac{1}{N} \sum x_k$, $y_c = \frac{1}{N} \sum y_k$;
- радиус по диагонали ячейки $r = \frac{l\sqrt{2}}{2}$ для оценки габаритов препятствия.

Ячейка классифицируется как занятая, если $h^{(i,j)} \geq h_{\text{thresh}}$, где h_{thresh} — пороговое значение, определяемое индивидуально.

Реализация и тестирование. Симулятор реализован на C++ с использованием ROS2 и библиотеки PCL [9]. Этот симулятор позволяет задать трехмер-

ную сцену в RViz2, содержащую объекты, и имитировать работу как одного, так и нескольких лидаров. В рамках данной работы архитектура взаимодействия построена на механизмах ROS2, где каждый алгоритм реализуется как отдельный узел, обменивающийся сообщениями через каналы (англ. topics) [2]. Схема взаимодействия компонентов проекта представлена на рис. 1.

Система состоит из следующих основных узлов: `multi_lidar_simulator` запускает несколько виртуальных лидаров, каждый из которых публикует облако точек в свой отдельный топик; `cloud_merger_node` подписывается на облака от всех сенсоров, выполняет фильтрацию, удаление плоскости пола и преобразование координат, после чего публикует объединенное облако в топик `/merged_cloud`; `trunk_detector_node` принимает объединенное облако, проводит пространственную агрегацию и классификацию возвышенных объектов и публикует результат в виде массива структур в топик `/detector_output`; `trunk_marker_visualizer_node` подписывается на результаты классификации и визуализирует препятствия в виде цилиндров; RViz предоставляет графический интерфейс для визуализации сцены, положения сенсоров, облаков точек и классифицированных объектов.

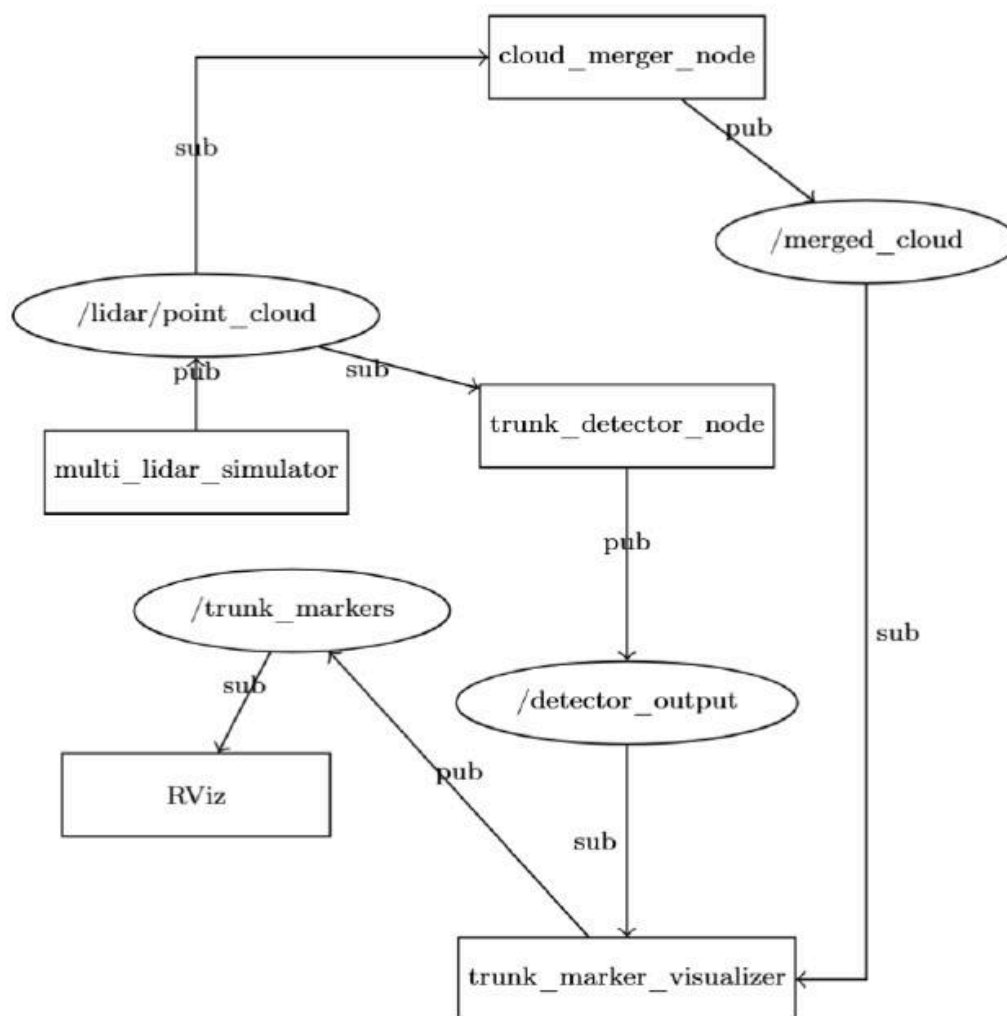


Рис. 1. Схема взаимодействия компонентов разработанного симулятора

Пример визуализации представлен на рис. 2.

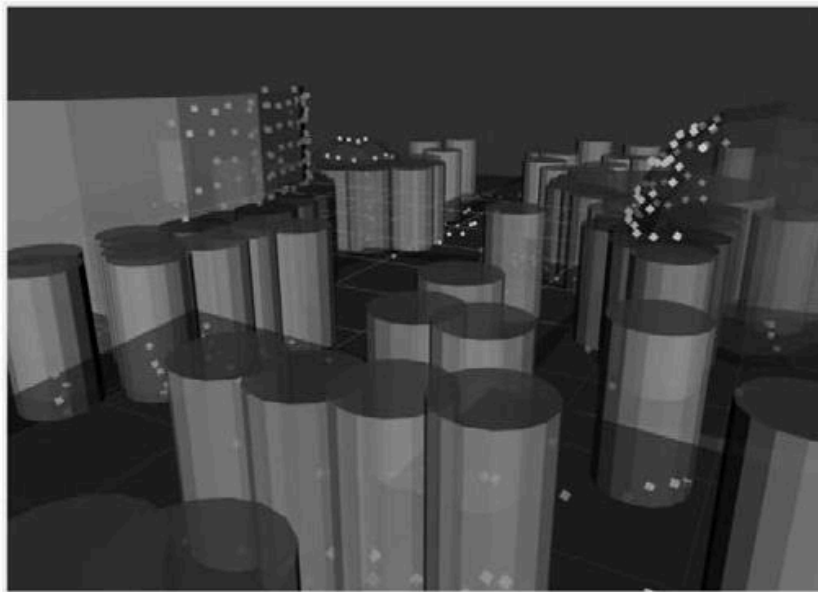


Рис. 2. Визуализация сцены с облаками точек, объектами и препятствиями

Заключение. В ходе работы был разработан открытый легковесный симулятор на C++ с использованием ROS2 и PCL для симуляции лидара и обработки облаков точек в системах компьютерного зрения мобильных роботов. Система может служить основой как для тестирования алгоритмов компьютерного зрения и восприятия, так и для проведения исследований в области навигации, планирования и управления мобильными роботами. Перспективы развития включают расширение поддерживаемых примитивов, добавление моделей шумов и атмосферных эффектов для реалистичности, а также внедрение методов классификации. Возможна интеграция с внешними источниками данных и экспорт результатов в совместимые форматы для построения карт и планирования движения.

Литература

- [1] Потапов А.С. *Системы компьютерного зрения*. Санкт-Петербург, Университет ИТМО, 2016, 160 с.
- [2] Quigley M., Gerkey B., Smart W.D. *Programming Robots with ROS*. O'Reilly Media, 2015, pp. 11–12.
- [3] Тачков А.А., Козов А.В., Вуколов А.Ю. Особенности портирования Robot Operating System на программно-аппаратную платформу «Эльбрус». *Программные продукты и системы*, 2019, № 4, с. 655–664.
- [4] Головки И.Д. Автоматическая система 3D-картографирования местности с помощью лидара. *Актуальные проблемы авиации и космонавтики*, 2023, № 1, с. 625–627.
- [5] Железнов И.И., Будаков В.П. Метод вокселизации при моделировании переноса излучения в неравновесной среде. *Атмосферная радиация и динамика (МСАРД-2023): сб. тр.* Санкт-Петербург, Изд-во ВВМ, 2023, с. 124–127.

- [6] Сулейманов А. Анализ экспериментальных данных методом главных компонент. *Известия Казанского государственного архитектурно-строительного университета*, 2007, № 1 (3), с. 81–83.
- [7] Wang P. Research on 3D Point cloud data preprocessing and clustering algorithm of obstacles for intelligent vehicle. *World electric vehicle journal*, 2022, no. 13.
<https://doi.org/10.3390/wevj13070130>
- [8] Kozov A.V., Volosatova T.M., Vukolov A.Y. Structural Obstacle Recognition Method and Its Application in Elevated Terrain Objects Search. *International Russian Automation Conference, RusAutoCon-2018*, 2018, art. no. 8501765.
<https://doi.org/10.1109/RUSAUTOCON.2018.8501765>
- [9] Можяев А.Н. Сегментация облаков точек с помощью средств библиотеки point cloud library. *Экстремальная робототехника*, 2018, № 1, с. 124–127.

Development of a simulator and point cloud processing algorithms for computer vision systems of mobile robots

Rogaleva Iuliia Alexandrovna

Kozov Alexey Vladimirovich (*scientific advisor*)

BMSTU, Moscow, Russia

The paper considers the development of a lightweight laser rangefinder simulator (lidar) and a set of point cloud processing algorithms designed for use in computer vision systems of mobile robots. Mathematical models of scanning, point cloud processing, and geometric classification of obstacles are proposed, and a modular architecture in the C++ programming language using ROS2 and the PCL library is implemented. The created simulator and implemented algorithms can be used to test and debug perception systems in robotics in conditions where access to real equipment is limited, difficult or unsafe.

Keywords: lidar, point cloud, simulation, ROS 2, classification, filtering, computer vision systems