

DOI: 10.32517/2221-1993-2025-24-3-19-23

**Вадим Лебеденко**

победитель Всероссийского конкурса «Робототехника в информатике»
в номинации «Программируем роботов — изучаем информатику»,
Одинцовский «Десятый лицей», г. Одинцово, Московская область, Россия

СОЗДАНИЕ АВТОНОМНОЙ СИСТЕМЫ НАВИГАЦИИ ДЛЯ СКЛАДСКИХ РОБОТОВ*

1. Введение

В XXI веке — веке цифровых технологий — для оптимизации различных процессов активно используются роботы. Их применяют везде: на заводах, в сервисных центрах, в космической промышленности и т. д. Одним из привычных применений роботов является их использование на складах.

Для решения актуальных проблем, связанных с применением таких роботов, был осуществлен **проект «Разработка автономной системы навигации для складских роботов»**, представленный в данной статье.

1.1. Проблема проекта

С развитием технологий, применяемых на складских работах, возрастают стоимость оборудования и требуемая мощность компьютеров, установленных на складских роботах. При увеличении мощности компьютера растет и размер всего робота. Это приводит к тому, что подобных роботов нельзя использовать в маленьких или узких складских помещениях, а также в организациях с недостаточно большой прибылью для покупки оборудования.

Соответственно, можно обозначить следующую **проблему**: дороговизна и требовательность к пространству уже созданных систем навигации для складских роботов в маленьких помещениях.

1.2. Актуальность проекта

По данным Aritzon, рынок автоматизированных управляемых складских тележек будет расти в среднем на 10 % в год [4]. Такие тележки помогают компаниям не только сократить расходы на сотрудников, но и повысить скорость и эффективность работы склада, исключить человеческий фактор из рисков. Это позволяет увеличить прибыль организации и качество, например, качество продуктов, хранящихся на складе.

1.3. Цель и задачи проекта

Цель проекта заключается в разработке системы автономной навигации для складских роботов, не тре-

бующей дорогих и больших по размеру комплектующих для своей работы.

В задачи проекта входит:

1. Изучить, какие технологии обычно используются в автономных системах навигации.
2. Создать схему работы автономной системы навигации.
3. Разработать такую систему и оптимизировать ее.
4. Провести тестирование алгоритма на устройствах разной мощности.
5. Сделать вывод из полученной информации, устранить неполадки и улучшить итоговый продукт.

1.4. Ожидаемые результаты

По итогам работы алгоритм, созданный в ходе реализации проекта, должен представлять собой полностью автономную (за вычетом управляющего воздействия) систему навигации. Для этого он должен:

- обнаруживать препятствия;
- строить карту местности;
- принимать управляющий сигнал от алгоритма или человека;
- создавать путь из текущей точки в указанную.

1.5. Практическая значимость проекта

Алгоритм может быть использован на роботах для небольших компаний, которые хотят автоматизировать работу своего склада. Это, например:

- магазины;
- пункты выдачи заказов;
- сортировочные центры;
- другие предприятия.

1.6. Этапы осуществления проекта

В соответствии с заявленными целью и задачами проекта были определены этапы его осуществления:

1. *Подготовительный этап* (сентябрь — октябрь): определение темы проекта, сбор необходимой информации о системах навигации.
2. *Аналитический этап* (ноябрь — декабрь): конкретизация полученных знаний, анализ рынка

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_03-2025/

существующих систем навигации, их преимуществ и недостатков.

3. *Практический этап* (январь — март): создание алгоритма, его доработка и тестирование.

2. Теоретическая часть проекта

2.1. Анализ существующих решений

В большинстве современных роботов для картографирования и навигации используется система **SLAM** (Simultaneous Localization and Mapping — одновременная локализация и построение карты). Она позволяет одновременно на основании показаний с разных видов датчиков составлять карту окружающей местности и определять свою относительную позицию на ней [3].

Обычно используют следующие датчики:

- *LiDAR* (Light Detection and Ranging — обнаружение и определение дальности с помощью света) — технология, позволяющая определять дистанцию до окружающих объектов с помощью лазерного дальномера;
- *одометры* — для дополнительной поддержки информации о перемещении робота;
- *гироскопы и акселерометры* — для динамического реагирования на изменения в среде и отклонения в пути;
- *UWB-навигация* (Ultra Wideband — сверхширокая полоса, беспроводная технология связи, использующая сверхширокую полосу для несущего сигнала и позволяющая передавать данные на небольшие расстояния при крайне низком энергопотреблении) — для определения позиции в маленьких помещениях с большой точностью.

Такие технологии используют роботы крупных компаний, таких как Amazon, Alibaba.

2.2. Научная база проектного решения

Существует несколько видов систем навигации [2]:

- *глобальные системы навигации* — те, которые работают в больших масштабах: GPS, ГЛОНАСС и подобные;
- *локальные системы навигации* — системы навигации, обычно работающие в помещениях или на небольших территориях. Они больше всего подходят для замкнутых пространств;
- *автономные системы навигации* не зависят от внешних сигналов (например, GPS), что играет важную роль при их использовании на складах;
- существуют также и *гибридные подходы*, которые комбинируют преимущества разных систем.

В локальных системах навигации активно используется **технология SLAM**. Это основа для построения карты склада в реальном времени. Существует несколько ее вариантов: визуальный (с помощью камер) и LiDAR-SLAM (на основе лазерных сканеров).

Для перемещения из одной точки в другую нужны **планировщики пути**. Они используют разные **методы планирования пути**:

- *Тангенциальное избегание препятствий*. Это алгоритм планирования пути, который позволяет

роботам динамически объезжать препятствия, сохраняя плавность траектории и минимизируя резкие изменения направления. Этот метод особенно эффективен в средах с движущимися объектами, например, на складах, где на пути робота могут внезапно появляться люди или другие роботы.

- *Потенциальные поля*. Этот метод построения пути заключается в «притягивании» робота к целевой точке и «отталкивании» от препятствий, по аналогии с магнитными полями.
- *Гистограммы векторных полей*. Это алгоритм планирования пути для роботов, который помогает выбирать безопасное направление движения в условиях препятствий. Он анализирует данные с датчиков, строит «карту» препятствий вокруг робота и вычисляет оптимальный маршрут, избегая столкновений. Алгоритм очень эффективен, из-за чего стал весьма популярным.

В большинстве роботов также используются **компьютерное зрение** и **искусственный интеллект** [1]. Они помогают как эффективнее строить пути, так и обнаруживать препятствия на изображениях в реальном времени.

3. Практическая часть проекта

3.1. Основы написания алгоритмов

Для написания кода был выбран язык Python из-за наличия в нем необходимых для проекта библиотек. При этом ROS (Robot Operating System* — операционная система для роботов) не использовалась для большего контроля над происходящим «под капотом».

Для более эффективной работы продукта было принято решение о структуризации алгоритма и его разбиении на потоки. Каждый модуль робота находится в отдельном файле и представляет собой класс с функцией *run()*, которая возвращает объект потока для запуска. Использование объектно-ориентированного программирования гарантирует расширение и дополнение системы новыми модулями без особых изменений в основном коде проекта.

Схему работы всего алгоритма можно представить следующим образом (рис. 1):

1. *Входные данные*. Информация берется с одометров, LiDAR и камеры.
2. *Обработка*. Данные с одометров и LiDAR обрабатываются алгоритмом SLAM, изображение с камеры передается в алгоритм компьютерного зрения. После обработки мы получаем относительную позицию робота и информацию о наличии препятствия перед роботом.
3. *Внешнее воздействие*. Чтобы роботу было понятно, куда ему ехать, он должен управляться внешним алгоритмом. Такой алгоритм должен задавать, куда роботу ехать в данный момент.
4. *Построение и планирование пути, движение*. На основе информации, полученной на предыдущих этапах, робот строит оптимальный маршрут и движется, объезжая препятствия.

* <https://www.ros.org/>

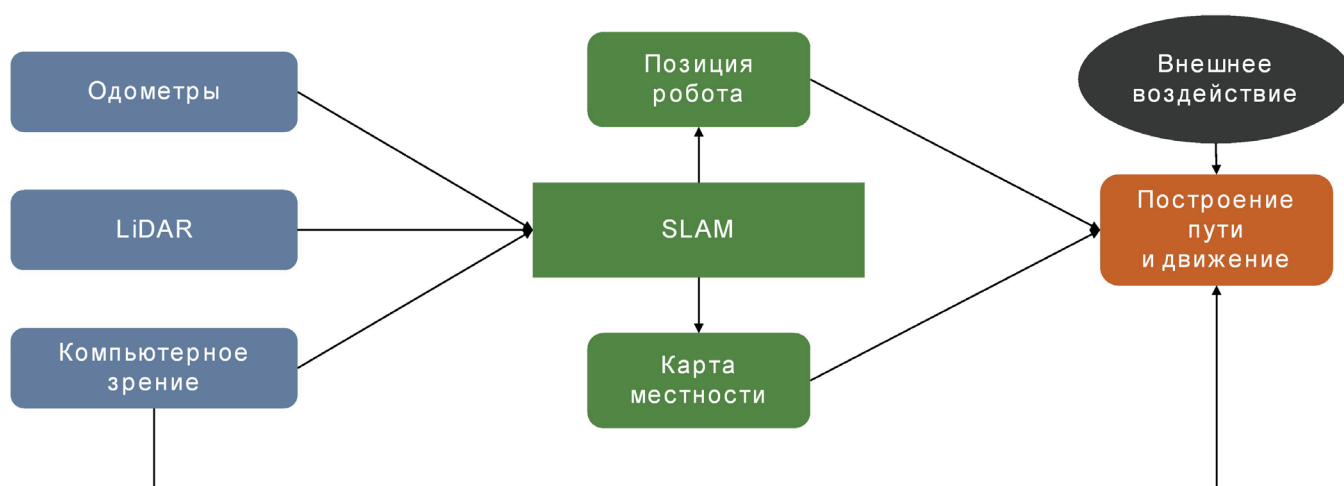


Рис. 1. Схема работы алгоритма

3.2. Алгоритм SLAM

Для задач картографирования была выбрана библиотека BreezySLAM. Среди своих аналогов она имеет следующие преимущества:

- не требует ROS для своей работы и полностью написана на Python;
- быстрая и оптимизированная;
- ее можно легко использовать с любым набором датчиков.

Данные для алгоритма берутся с LiDAR и одометров колес. Для примера я использовал RPLidar A1. Он очень мал в размерах и по цене, что соответствует требованиям моего проекта. Для взаимодействия с ним была использована библиотека RPLidar. Любой другой подобный модуль может быть подключен к алгоритму, достаточно написать функцию для сбора данных с него.

Для взаимодействия с библиотекой для SLAM я создал отдельный класс. В нем содержатся методы для запуска основного потока, где производится сканирование местности и построение карты, а также методы для рендеринга карты — ее переноса в формат изображения для отображения пользователю.

3.3. Алгоритм компьютерного зрения

Так как продукт моего проекта должен быть максимально оптимизированным для работы на маломощных компьютерах, я решил не использовать нейросети для задач компьютерного зрения. Поэтому я разработал следующий алгоритм детекции препятствий на картинке (рис. 2):

1. Изначальное изображение обрезается, при этом остается только нижняя часть (будем считать, что в нижнюю часть изображения попадают все препятствия, которые могут мешать роботу). Размер нижней части можно настроить в зависимости от размеров робота.
2. На изображении выделяется доминантный цвет (доминантный — самый часто встречающийся). Такой алгоритм был выбран путем отбора из нескольких (средний цвет, самый яркий/темный цвет, доминантный цвет) по показателям точности и эффективности.

3. Делается бинарная маска изображения по доминантному цвету (белым выделяются те области изображения, которые отличаются от цвета маски не более чем на значение порога, которое задается в функции). Оптимальным значением оказалось 50. Все остальное изображение окрашивается в черный цвет.
4. Применяется размытие по Гауссу для скрытия артефактов бинарной маски. Линии и контуры с разрывами становятся целыми.
5. Алгоритм Кэнни выделяет границы объектов.
6. Преобразование Хафа находит на изображении прямые линии и возвращает координаты их начал и концов.
7. Выделяются только линии, угол наклона которых меньше заданного (оптимальный — примерно 40 градусов). У каждой линии находятся центр (среднее между координатами начала и конца) и далее среднее между всеми серединами линий. Считается, что полученные координаты и есть препятствие.

Такой алгоритм позволяет находить объекты, которые не видны LiDAR, но заметны для камеры. У него есть существенный недостаток: иногда из-за шумов он может быть неточен. Но это исправляется с помощью простейшего медианного фильтра.

3.4. Алгоритм построения пути

За основу алгоритма построения пути был взят алгоритм A* из-за своей скорости и эффективности за счет эвристической функции. Так как карта, созданная SLAM, имеет вид двумерного массива, эвристическую функцию можно записать в следующем виде:

$$E = |x - x_1| + |y - y_1|,$$

где x, y — координаты текущей точки, x_1, y_1 — координаты точки, до которой определяется расстояние, E — оценка расстояния.

3.5. Веб-интерфейс для взаимодействия с системой

Чтобы управлять роботом, был создан веб-интерфейс. Подключение происходит через любой браузер с поддержкой WebSockets по протоколу HTTP.

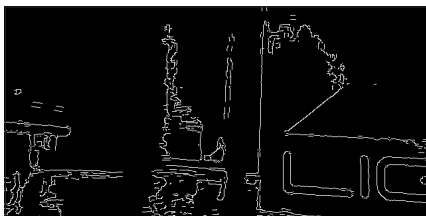
Получение кадра и обрезка его верхней части	
	
Изначальное изображение	Обрезанное изображение
Определение доминантного цвета, использование бинарной маски и размытие	
	
Бинарная маска по доминантному цвету	Размытие по Гауссу
Выделение границ объектов, поиск линий по преобразованию Хафа и обработка результатов	
	
Выделение границ объектов по Кэнни	Результат работы алгоритма

Рис. 2. Иллюстрации к работе алгоритма компьютерного зрения

С помощью данного интерфейса можно:

- просматривать карту, построенную алгоритмом SLAM;
- просматривать позицию робота и целевой точки, а также построенный путь;
- выбирать точку, в которую робот должен проехать.

Веб-интерфейс получает и отправляет данные по протоколу WebSockets. Такой протокол позволяет непрерывно получать и отправлять данные. Был реализован простой API, где все данные для удобства передаются в формате JSON. Изображение карты рендерится на роботе и отправляется в виде base64-строки, а на клиенте декодируется и отображается пользователю.

3.6. Тестирование алгоритма

Алгоритм был протестирован целно (работа всего алгоритма) и модульно (проверка каждого модуля на

работоспособность и скорость работы). Тесты проводились на одноплатных компьютерах Jetson NANO и Raspberry PI 4.

1. *Тестирование всего алгоритма.* Для данного теста был взят колесный робот с LiDAR и камерой; был запущен алгоритм. В веб-интерфейсе был виден результат сканирования окружающей обстановки — комната. После выбора целевой точки был построен путь. Данных о нахождении препятствия алгоритмом компьютерного зрения не было. После установки препятствия (картонной коробки) алгоритм показал наличие препятствия и перестроил путь для его обхода.

2. *Проверка алгоритма SLAM.* Был запущен модуль LiDAR и написан небольшой скрипт для проверки скорости сканирования и обработки. По итогам тестирования получены результаты, представленные в таблице 1.

Таблица 1

Результаты тестирования алгоритма SLAM

Компьютер	Скорость сканирования, Гц
Jetson NANO	3 (максимальная скорость для данной модели LiDAR)
Raspberry PI 4	2

3. *Проверка алгоритма компьютерного зрения.* Была запущена камера и поставлено препятствие напротив нее, а также был написан скрипт для замера скорости обработки одного кадра. Итоги тестирования представлены в таблице 2.

Таблица 2

Результаты тестирования алгоритма компьютерного зрения

Компьютер	Скорость обработки кадров, FPS
Jetson NANO	25 (лимит)
Raspberry PI 4	25 (лимит)

4. *Проверка алгоритма построения пути.* Был создан двумерный массив размером 8000 на 8000 элементов. Случайным образом в нем были нарисованы линии — препятствия. Тестирование алгоритма построения пути было проведено на нескольких расстояниях (табл. 3).

Таблица 3

Результаты тестирования алгоритма построения пути

Расстояние от начала до конечной точки	Результат на Jetson NANO, Гц	Результат на Raspberry PI 4, Гц
1000	4	1.8
4000	2	0.7
11000 (по диагонали)	0.8	0.2

Таким образом, по результатам тестирования разработанный мной алгоритм может с приемлемой скоростью работать на компьютерах малой мощности.

4. Перспективы развития проекта

В будущем я хочу развивать свой проект по нескольким направлениям:

1. Создать образец робота, использующего данный алгоритм навигации. Робот сможет перевозить небольшие коробки с помощью манипулятора, установленного на нем.
2. Улучшить алгоритм навигации: заменить алгоритм А* более современными методами построения пути для роботов (например, методом векторных полей).
3. Создать интерфейс и протокол для роевого интеллекта — объединения роботов в единую сеть.

5. Экономическая и экологическая составляющие проекта

Разработанный мною алгоритм снижает расходы компаний на сотрудников и издержки, связанные с человеческим фактором. Он также оптимизирует процесс перевозки на складах для уменьшения потребления электроэнергии.

6. Заключение

Разработанный алгоритм для автономной навигации робота в складских помещениях маленького размера полностью выполняет свои функции: определяет позицию робота, создает пути, динамически определяет препятствия и обновляет путь.

Мною также был разработан веб-интерфейс для управления алгоритмом извне.

Ссылка на исходный код проекта (GitHub):

<https://github.com/Vadimych1/CloudBots.Rover/tree/final>

Список источников

1. *Иванько М. А., Клепикова А. В.* Системы искусственного зрения // Вестник МГУП имени Ивана Федорова. 2015. № 5. С. 50–52. EDN: VNUFBT.
2. Курс радионавигации. https://srns.ru/wiki/Курс_радионавигации
3. *Луговской В. В.* SLAM как передовой метод навигации и его виды // Инновационная наука. 2023. № 4-2. С. 46–49. EDN: DAMJKW.
4. FlexQube приступил к коммерциализации AMR // RoboTrends. 18.12.2023. <https://robotrends.ru/pub/2351/flexqube-pristupil-k-kommercializacii-amr>