Содержание

Аннотация

Робототехника быстро развивается в последние годы, приводя к появлению все большего числа беспилотных технологий, роботов-доставщиков и других устройств, помогающих людям повседневно. Роботы стали неотъемлемой частью различных отраслей, таких как логистика, строительство, охрана и обслуживание. Использование роботов может существенно упростить жизнь людей, освободив их от рутины и части обязанностей. Не исключением стали и передвигающиеся платформы, которые могут перемещаться по вертикальным или изогнутым поверхностям – уже сейчас они используются в разных областях, где люди не могут получить доступ для обслуживания. Например, Газпром уже в 2022 году разработал роботов на гусеницах, которые используются для проверки труб изнутри на утечки. В рамках нашего проекта под названием Whiteboard bot, мы решили спроектировать аналогичного робота для перемещения по вертикальной поверхности маркерной доски. Робот призван помочь стирать записи с доски, что облегчит жизнь преподавателей, учащихся и персонала. Этот проект также помогает студентам изучать основы инженерии, компьютерного зрения и планирования траектории.

Ключевые слова

Робототехника, инженерное моделирование, перемещающиеся платформы, компьютерное зрение, обработка изображений, планирование траектории.

1 Основные термины и понятия

Аппаратная платформа– основная часть робота, которая передвигается по доске.

OpenCV – библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом.

ROS2 – это гибкая платформа для разработки программного обеспечения роботов.

Boost – собрание библиотек классов, использующих функциональность языка C++ для различных повседневных подзадач программирования.

Aruco маркеры – технология для позиционирования робототехнических систем с использованием компьютерного зрения.

Локализация бота – определение положения робота в пространстве относительно краев маркерной доски.

Дисторсия – аберрация оптических систем, при которой коэффициент линейного увеличения изменяется по мере удаления отображаемых предметов от оптической оси.

Гомография – преобразование, которое отображает точки одного изображения в точки соответствия другого изображения.

Сегментация – в обработке изображений: процесс разделения цифрового изображения на несколько сегментов.

Нода – в ROS2: фундаментальный элемент ROS2, который выполняет одну функцию в робототехнической цепи.

2 Введение

Робототехника – это наука, которая изучает проектирование, создание и использование роботов. Она имеет широкое применение в жизни людей и оказывает значительное влияние на различные отрасли промышленности, образования и медицины. В настоящее время наблюдается быстрое развитие технологии робототехники, что приводит к увеличению числа разнообразных устройств, задействованных как в повседневной жизни людей, так и в промышленности. Робототехника нашла применение в таких сферах, как логистика, строительство, охрана, а также обслуживание и контроль. Уже сейчас во многих домах можно встретить роботов-пылесосов, мойщиков окон или газонокоссльщиков, а на складах – роботов-погрузчиков. Такая универсальность позволяет роботам значительно облегчить жизнь человека, помогая ему избавиться от рутины и освободить личное время от бытовых занятий, а также помочь рабочим в исполнении их обязанностей.

В рамках нашей задачи мы решили расширить сферу использования роботов до применения в институтах при обучении студентов, а именно – для уменьшения времени стирания с маркерной доски. Во время проведения занятий преподаватель может достаточно часто тратить свое время на рутинную работу по очистке доски для ведения новых записей, а именно такую работу и может заменить робот. В рамках проекта мы спроектируем робота, который по команде сможет стирать написанное с маркерной доски, а также разработаем все необходимое программное обеспечие для его работы.

Для успешной реализации данного проекта мы разделили всю работу на две части. Первая – проектирование аппаратной платформы, с помощью которой и будет производиться очистка доски. Данная часть включает в себя проектирование прототипа робота, создание и печать 3D-модели, а также тестирование нагрузки и вариантов размещения на маркерной доске. Вторая часть – калибровка камеры, написание алгоритмов распознавания краев доски, позиции робота, записей на доске, а также построение кратчайшего маршрута для объезда нужных частей доски. Также в этой части предусмотрена реализация связи всех отдельных частей задачи для совместной работы.

3 Общая структура кода и использованные библиотеки

Основными технологиями, которые использовались для реализации данного проекта, были ROS2 (Robot Operating System) [5] – открытая платформа для разработки программного обеспечения, которая предоставляет инфраструктуру для построения и управления комплексными робототехническими системами; а также OpenCV (Open Source Computer Vision) [2] – библиотека с открытым исходным кодом, которая предоставляет инструменты для разработки приложений компьютерного зрения, а также набор алгоритмов для обработки изображений. Таким образом, все отдельные части программного обеспечения будут связаны с помощью ROS2, а в самих частях будут применяться некоторые алгоритмы и функции из OpenCV.

Альтернативами ROS2 являются ROS (прошлое поколение), LCM (набор библиотек от MIT), ZeroMQ и другие. ROS2 имеет понятную документацию и используется в других проектах лаборатории робототехники, поэтому выбор остался на этой платформе. OpenCV, в свою очередь, имеет достаточное количество обучающих уроков и фрагментов, а также весь необходимый функционал для реализации проекта.

Всю программную часть проекта можно разделить на 3 части: локализация доски и робота в просранстве, сегментация изображения, а также планирование траектории и построение маршрута. Каждая из частей будет выполняться в своей отдельной ноде – процессе, котороый будет обрабатывать входящие параметры и публиковать в топики (каналы для связи нод друг с другом) полученные результаты. Соответственно, всего будет реализовано 3 ноды, речь о которых пойдет далее.

Для визуализации работы робота и всех нод была использована программа Foxglove Studio [1], в которой есть возможность настраивать отображение тех или иных сообщений, а также отслеживать приходящие параметры.

4 Калибровка камеры

Выполнил Скоробогатов Гордей

В данной главе будет описан алгоритм калибровки камеры для того, чтобы убрать искажения линзы и получить четкое изображение.

Многие камеры так или иначе вносят искажения в захватываемые изображения. Наличие таких искажений приводит к тому, что прямые линии кажутся изогнутыми, причем искажение может увеличивается по мере удаления от центра изображения. К примеру, на

рис. 4.1 видно, что при наличии искажений стороны квадрата могут как выпирать наружу, так и быть вогнутыми внутрь.

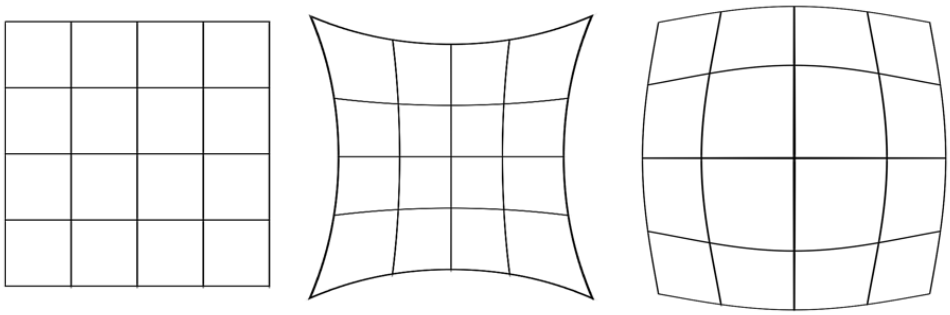

Рис. 4.1: Виды дисторсии: исходный объект; изображение при положительной и отрицательной дисторсии

Чтобы избавиться от подобных искажений, необходимо найти "внутренние"параметры камеры, которые включают в себя фокусное расстояние и оптические центры. Эти величины можно выразить в виде матрицы размера 3×3, которая будет уникальной для каждой камеры:

$$\text{camera matrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

А также нужно найти коэффициенты дисторсии, которые выражаются в виде:

$$\text{distortion coefficients} = \begin{pmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{pmatrix}$$

Чтобы найти эти параметры, мы должны предоставить несколько образцов изображений с четко определенным рисунком (например, шахматной доски). Мы находим некоторые конкретные точки, относительное положение которых нам уже известно (например, углы квадратов на шахматной доске). Мы знаем координаты этих точек в пространстве реального мира, и мы знаем координаты на изображении, поэтому мы можем рассчитать коэффициенты искажения.

Таким образом, необходимо было сделать несколько десятков снимков шахматной доски с заданным числом строк и столбцов, причем располагать ее нужно было по всем частям

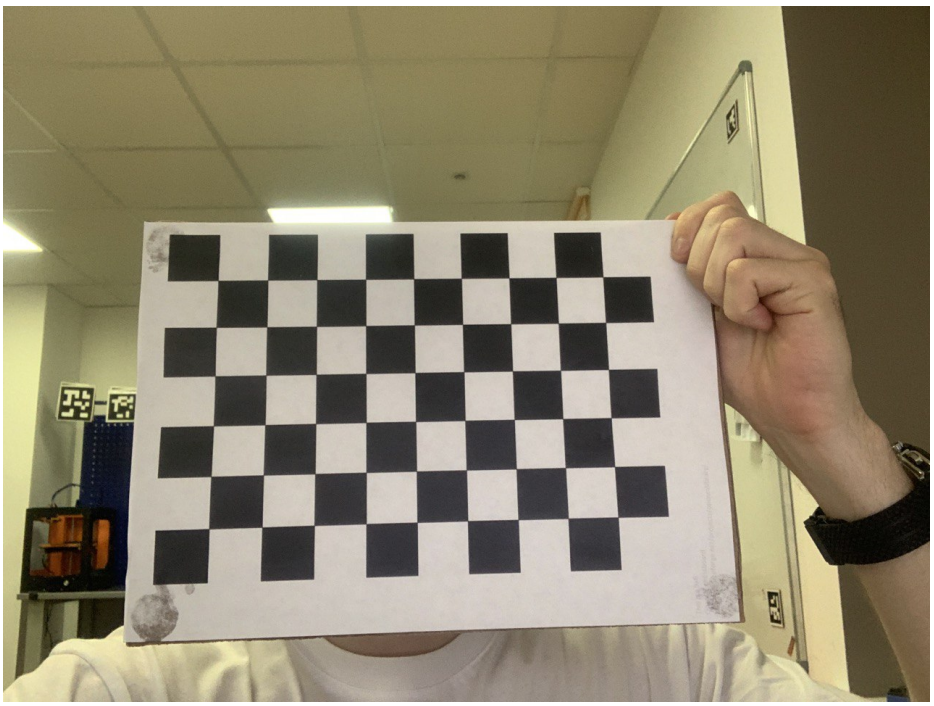

Рис. 4.2: Пример фото с шахматной доской

кадра. После получения снимков с помощью функций findChessboardCorners и calibrateCamera мы получили необходимые коэффициеты дисторсии и матрицу камеры, сохранив все эти параметры.

Далее при инициализации камеры оставалось только применить полученные матрицы и коэффициенты к камере, чтобы избавить ее от искажений.

5 Локализация

Выполнил Скоробогатов Гордей

В данной главе будет описан алгоритм преобразования исходного изображения и последующим распознаванием местоположения робота на доске.

5.1 Репроекция доски на прямоугольник

Самым первым этапом работы является локализация границ доски и ее репроекция на плоскость, без которой дальнейшие шаги не представляются возможными. Репроекция необходима для того, чтобы превратить искревленное изображение с камеры в плоский прямоугольник, на котором будут сохранены все отношения расстояний и прямые углы.

Для данного проекта был необходим четкий и быстрый способ определения доски, для чего была выбрана технология Aruco маркеров. Одной из основных функций Aruco марке-

ров является определение точного положения и ориентации объектов в пространстве. Для этого маркеры устанавливаются на объектах, чье положение и ориентация необходимо определить. Камера считывает позицию маркеров и использует полученную информацию для определения точного положения и ориентации объекта. Aruco маркеры являются одним из наиболее эффективных и простых в использовании видов маркеров в компьютерном зрении. Они состоят из уникального набора черных и белых квадратов, расположенных в определенной последовательности внутри квадрта (см. рис. 5.1). Также у каждого маркера есть свой идентификационный номер, что позволяет отдельно обрабатывать маркеры по номерам.

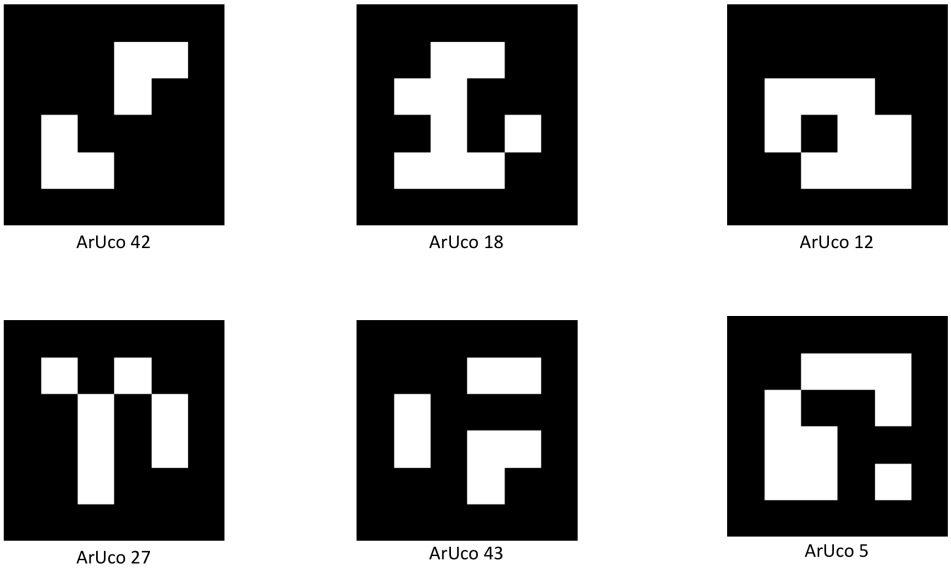

Рис. 5.1: Пример нескольких Aruco маркеров с указанием идентификационного номера

В нашем случае было расположено 4 маркера по краям доски, с помощью которых мы смогли практически моментально определять расположение доски в пространстве камеры.

Следующим шагом была реализация репроекции доски по углам на прямоугольный кадр. В данном случае нам необходимо поместить углы доски, найденные по Aruco маркерам, в углы прямугольного кадра заданного размера. Такое преобразование плоскости называется гомографией [4] – оно переводит прямые линии в прямые, сохраняя при этом параллельность, и записывается в равенстве:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = H \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

где x, y – исходные координаты точки, x', y' – полученные координаты после преобразования, а H – матрица преобразования размера 3×3. Для нахождении этой матрицы необходимы

всего 4 пары различных опорных точек. В нашем случае этими опорными точками послужат соответствующие углы маркеров, по которым и будет найдена необходимая матрица преобразования.

Алгоритм нахождения матрицы реализован в методе findHomography библиотеки OpenCV, который принимает на вход необходимые массивы точек и возвращает нужную нам матрицу. Данный алгоритм основан на решении системы линейных уравнений, и является достаточно эффективным и быстрым для наших задач. Таким образом, с помощью этого метода была найдена матрица преобразования, которая использовалась для гомографии доски. Результатом стал прямоугольный кадр, углами которого являются углы соответсвующих маркеров.

Рис. 5.2: Изображение доски после репроекции. Также определенные на доске Aruco маркеры выделены красной границей

5.2 Локализация робота

Локализация робота в пространстве тоже является важной частью данного проекта. Необходимо всегда поддерживать актуальную позицию робота относительно доски, а также его направление движения. Эту задачу также решали с помощью Aruco маркеров, прикрепив один из маркеров на аппаратную платформу. Таким образом, мы смогли несколько раз в секунду считывать изображение с камеры, распознавать на нем маркеры и понимать, где находится робот в данный момент времени. Также маркеры позволяют получить не только позицию, но и ориентацию объекта в пространстве, что позволило определять, в какую сторону смотрит бот, и куда он поедет в будущем.

5.3 Описание ноды камеры

Самой главной нодой проекта является нода камеры. Она публикует 3 параметра: репроецированное на плоскость изображение доски (с сжатием для визуализации и без для дальнейшей обработки), местоположение и ориентацию робота и угловых маркеров, а также несколько сообщений для визуализации – а именно границы маркеров и самого робота.

Нода несколько раз в секунду получает входящее изображение с камеры и выполняет следующий алгоритм:

1. Распознает местоположение Aruco маркеров на доске, а также местоположение робота
2. Находит матрицу гомографии, сохраняет ее значение во внутреннее состояние и применяет преобразование к изображению
3. Публикует полученное изображение, необходимые местоположение и линии для визуализации

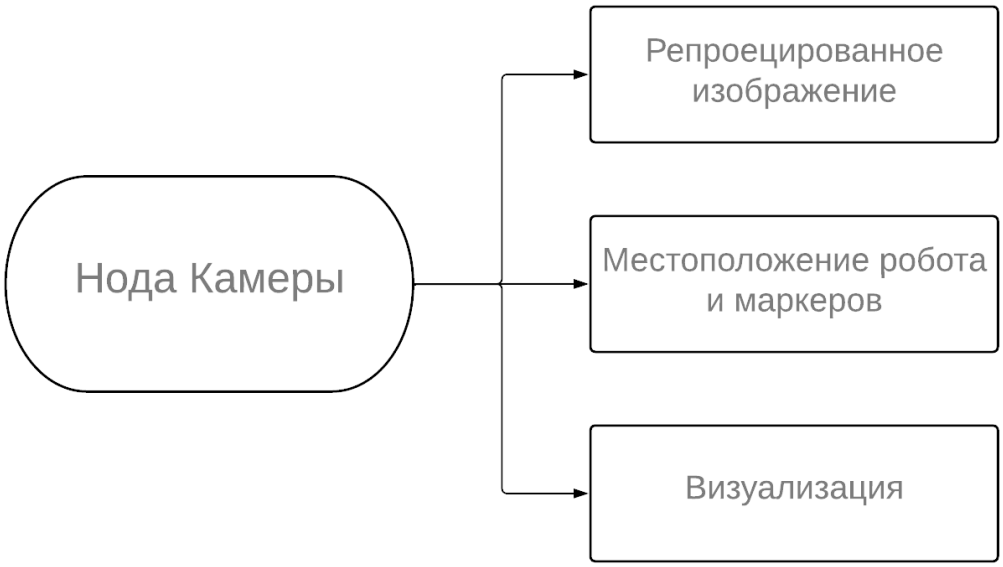

Рис. 5.3: Схема работы ноды камеры

Если же для нахождения матрицы гомографии не хватает распознанных маркеров (кто-то мог закрыть один из маркеров рукой), к изображению применяется матрица, полученная на предыдущем шаге. Таким образом, даже если камера не сможет распознать все 4 маркера, репроекция изображения все равно продолжит работать, а погрешность в таком случае будет минимальной.

Также для ноды можно задать несколько начальных параметров – а именно период обновления изображения и необходимое выходное разрешение.

6 Сегментация

Выполнил Скоробогатов Гордей

В данной главе будет описан алгоритм сегментации изображения и выделение частей доски с маркернами надписями.

6.1 Распознавание написанного на доске текста

На данном этапе мы умеем определять распроложение доски, а также позицию и ориентацию робота в пространстве. Теперь при поступлении сигнала нам необходимо определить, где на доске есть записи, чтобы понимать, куда роботу нужно будет двигаться.

В данном случае необходимо выделить цветной текст от белой доски. В нашем случае был применен следующий алгоритм:

1. Перевести изображение в градацию серого
2. Применить алгоритм threshold
3. Применить алгоритм adaptive threshold
4. Объединить полученные изображения из пунктов 2 и 3

Первый шаг алгоритма применяется для того, чтобы привести все в единую гамму. Алгоритм threshold заключается в том, что мы красим все пиксели в черный цвет, если их значение превысило определенный порог; и в белый иначе. Таким образом, мы отделяем четко выделяющиеся на фоне белого фрагменты, задавая ту или иную границу.

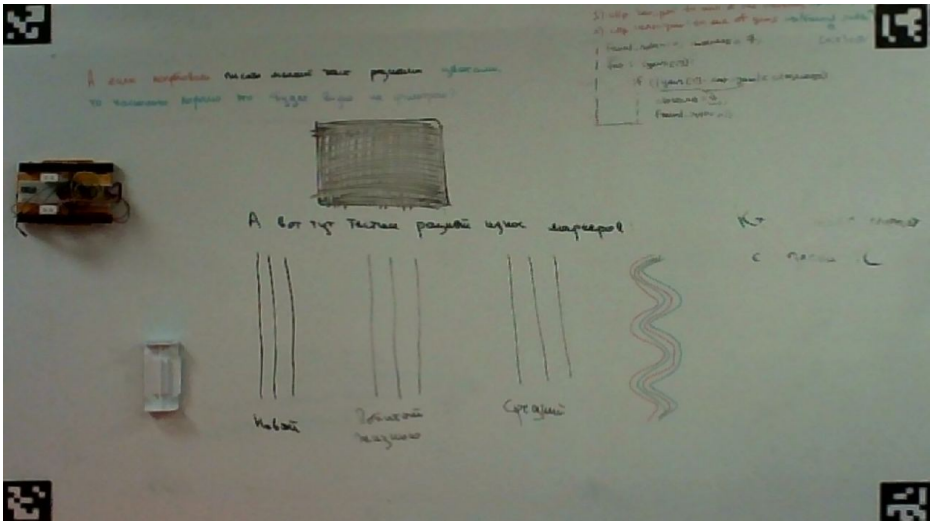

Рис. 6.1: Изображение до применения алгоритма adaptive threshold

Adaptive threshold работает по похожему принципу, только в адаптивном случае алгоритм определяет пороговое значение для пикселя на основе небольшой области вокруг него. Таким образом, мы получаем разные пороговые значения для разных областей одного и того же изображения, что дает лучшие результаты для изображений с различной освещенностью. Однако, у такого алгоритма есть свои проблемы – если дать ему на вход изображение с черным закрашенным квадратом, то он не сможет выделить весь квадрат целиком, а только его границы (см. рис. 6.1 и 6.2). Именно для этого на шаге 4 мы объединиямем изображения с адаптивным и обычным алгоритмами, тем самым улучшая точность распознавания.

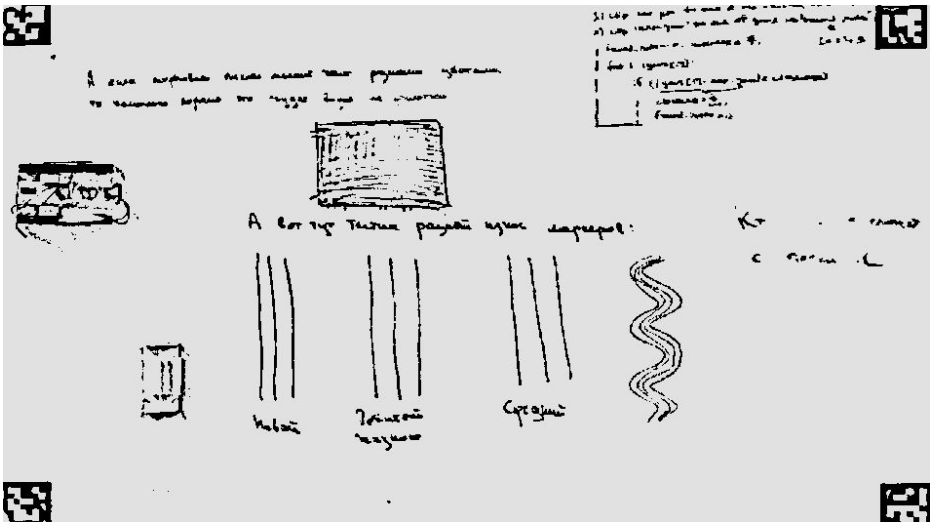

Рис. 6.2: Изображение после применения алгоритма adaptive threshold

В конечном итоге мы смогли успешно распознать написанные на доске символы, рисунки и текст. Результаты можно увидеть на рисунке 6.3.

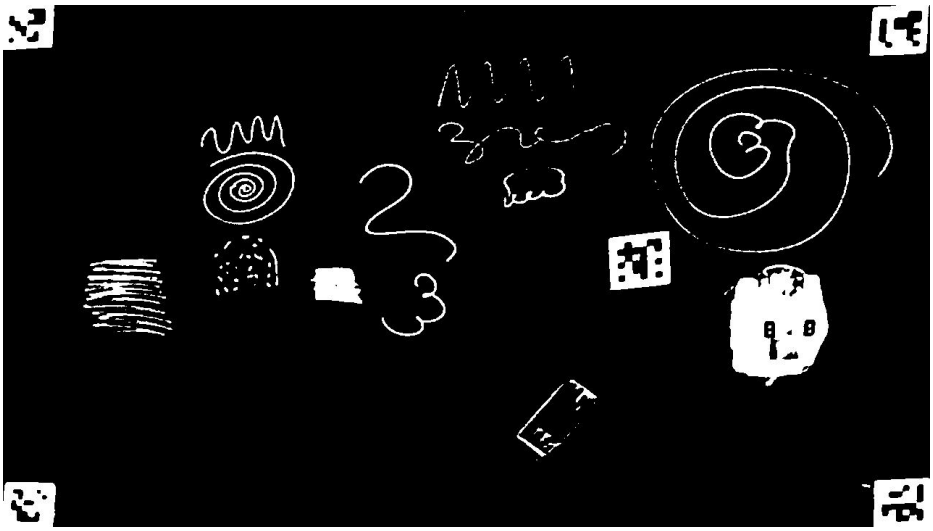

Рис. 6.3: Результат применения алгоритмов. Белым выделены части, отличающиеся от фона доски

6.2 Сегментация доски на фрагменты

После того, как был определен написанный текст, необходимо было разбить всю доску на условные фрагменты, по которым будет передвигаться робот. В качестве этих фрагментов были выбраны квадраты с заданной в пикселях стороной, причем размеры этих квадратов подбираются таким образом, чтобы проехавший по квадрату робот точно стер все внутри этого квадрата. Таким образом, необходимо поделить всю доску на сегменты и пометить те, в которых есть какие либо надписи.

Чтобы понять, нужно ли помечать очередной квадрат, был применен самый простой способ – если в этом квадрате есть хотя бы один пиксель белого цвета (то есть пиксель, который нужно стереть), то квадрат помечался условным флагом; в остальных случаях квадрат не помечался.

Таким образом, после всех примененных алгоритмов и шагов на выходе мы получаем прямоугольную матрицу, в которой некоторые клетки помечены флагом, сигнализирующем о необходимости стирания в этой области. После этого этапа можно переходить к построению маршрута робота.

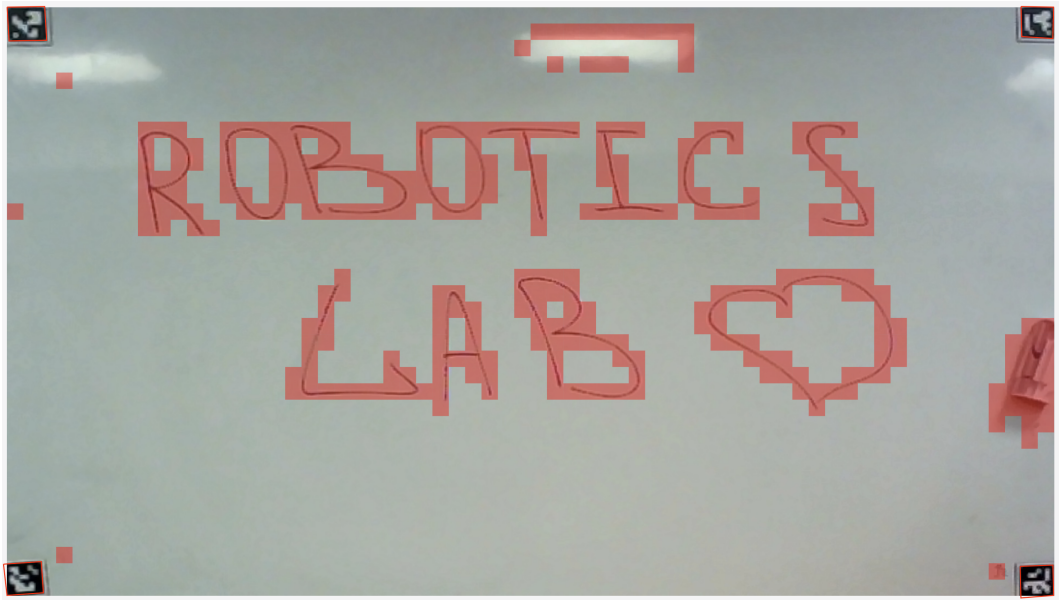

Рис. 6.4: Пример визуализации сегментации изображения с камеры. Красным помечены сегменты, внутри которых есть маркерные надписи

6.3 Описание ноды сегментации

Нода сегментации принимает репроецированное изображение с камеры и применяет к нему необходимые фильтры и алгоритмы распознавания надписей на доске. На выходе нода публикует сообщение, содержащее матрицу с пометкой каждого сегмента – нужно ли

стирать надписи в этом сегменте или нет. Также нода публикует сообщение с визуализацией, перекрашивая необходимые сегменты в полупрозрачный красный цвет.

Во время применения алгоритмов было необходимо сделать так, чтобы маркеры на углах доски, а также сам робот не влияли на сегментацию (иначе при планировании траектории мы бы захотели наехать на угловые маркеры). Для решения этой проблемы было принято решение убирать с сегментов вокруг маркеров и робота пометки, чтобы не считать их "закрашенными". Таким образом, на выходе мы получаем необходимую матрицу, по которой уже можно строить траекторию для объезда.

Для ноды сегментации можно задать следующие параметры: размер сегментов в пикселях (чем меньше размер, тем больше сегментов будут покрывать доску), а также период сегментации.

Рис. 6.5: Схема работы ноды сегментации

7 Планирование траектории

Выполнил Скоробогатов Гордей

В данной главе будет описан алгоритм построение траектории объезда клеток с некоторыми условиями на длину маршрута.

7.1 Постановка задачи

После сегментации изображение становится понятно, какие клетки должен объехать робот. Однако теперь возникает вопрос, в каком порядке объезжать эти клетки, чтобы проехать меньшее расстояние и затратить меньше времени. Эту проблему можно свести к задаче коммивояжера – одной из самых известных задач комбинаторной оптимизации, условием которой является проезд по всем городам хотя бы по разу с возвращением в стартовую точку по самому выгодному (в нашем случае самому короткому) маршруту. Данная задача является NP–сложной и в общем случае может потребовать очень много времени на поиск точного решения.

Однако в нашем случае можно воспользоваться некоторыми эвристиками и допущениями, которые минимально повлияют на итоговый результат, но зато очень сильно помогут сократить время вычисления маршрута. Давайте представим, что все сегментационные клетки – вершины некоторого графа G, причем соседние клетки соединены ребром. Робот в свою очередь умеет ездить от одной клетке к соседней по стороне или диагонали, тем самым передвигаясь по нашему графу. Также выделим все необходимые сегменты и для каждой пары вершин, соответствующих выделенным сегментам, проведем ребро заданного веса между ними.

Таким образом, мы получим полный взвешенный неориентированный граф, в котором нам нужно найти путь, который проходит по каждой вершине хотя бы раз, возвращается в исходную вершину и имеет минимальную (или достаточно близкую к минимальной) сумму весов.

Данную задачу можно решить перебором, рассматривая каждый из возможных обходом и подсчитывая его вес. Однако при таком методе решения будет затрачено очень много времени, причем даже для 2500 вершин время рассчета может привысить несколько десятков минут, что никак не подходит для решения нашей задачи.

7.2 Реализация

В нашем случае мы воспользовались готовым алгоритмом, предоставляемым библиотекой Boost для C++. Метод $metric\_tsp\_approx$ принимает на вход необходимый для обхода граф и возвращает последовательность вершин для обхода. Алгоритм поиска основан на алгоритме из книги "Introduction to Algorithms"Томаса Кормена [3], которая является одной из самых популярных книг по алгоритмам в мире и цитируется во многих университетах и учебных заведениях. Алгоритм основан на итеративном поиске минимального остовного дерева для имеющейся части графа с последующим применением алгоритма Прима.

При размере графа около 1000 вершин затраченное время на поиск пути составляло около 1 секунды, что вполне реально вписывалось в требования к нашему проекту.

7.3 Описание ноды планирования траектории

Нода планирования траектории принимает на вход матрицу сегментации, полученную от ноды сегментации, и преобразует ее в полный взвешенный неориентированный граф для обхода. Далее применяется описанный выше алгоритм библиотеки Boost, после чего нода публикует необходимую последовательность вершин, а также маршрут в виде отрезков,

соединяющих центры двух последовательных вершин обхода.

Также необходимо было помнить о том, что после посещения роботом запланированной клетки, ее нужно удалить из очереди. Для этого нода принимает на вход текущую позицию робота и во время каждого цикла сравнивает положение робота с клеткой, которую нужно посетить по маршруту. Если робот заехал в данную клетку, то она удаляется из очереди, и начинает рассматриваться уже следующая клетка. Таким образом, на каждом цикле осуществляется поддержка актуальной траектории, которая учитывает позицию робота.

Рис. 7.1: Схема работы ноды планирования траектории

8 Модель управления роботом

Выполнил Савчук Андрей

Для корректного позиционирования робота в пространстве и следования по траектории используется модель Аккермана (рис. 8.1)

Скорость каждой гусеницы вычисляется по формуле, где $c = 1/r$: кривизна поворота (curvature), r: радиус поворота в метрах, w_b: ширина робота в метрах (wheel base), v_b: линейная скорость робота от 0 до 1.

$$(1 \pm c \cdot w_b) \cdot v_b$$

8.1 Управление аппаратной платформой

Подключение к роботу осуществляется при помощи WiFi по протоколу Websockets: разработана ROS2 нода управления, принимающая команды перемещения из контроллера пути либо с пульта и передающая их по сети. Робот принимает следующие команды: `0;<curvature>;<velocity>;`, где curvature - кривизна поворота, velocity: линейная скорость; `1;` - поднять стирающий мезанизм, `2;` - опустить стирающий механизм.

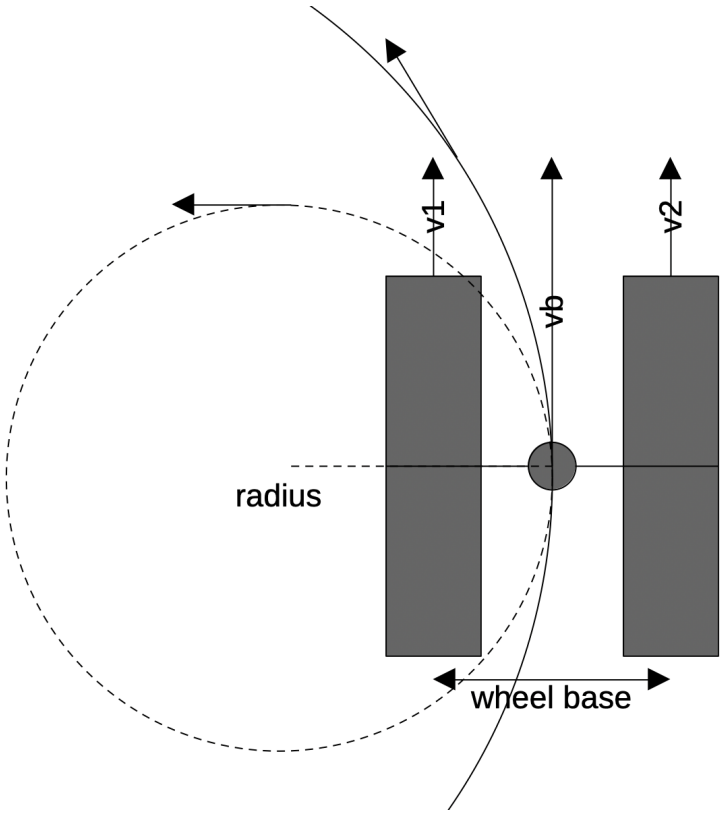

Рис. 8.1: Ackermann steering model

9 Аппаратная платформа

Выполнил Савчук Андрей

Финальная версия аппаратной платформы робота позволяет перемещаться в любое место доски, также присутствует механизм, который предотвращает непроизвольное стирание записей. Для перемещения по вертикальной доске используются гибкие магнитные гусеницы и магнитные стабилизаторы, удерживающие гусеницы в нужном положении и предотвращающие сваливание робота с доски. Гусеницы распечатаны из гибкого TPU филамента с жескостью 95A по Шору для того, чтобы предотвратить их растяжение под собственным весом робота. Для преодоления неизбежного соскакивания гусеницы со шкивов используются натяжители с произвольным уровнем натяжения. Управляющая часть робота состоит из микроконтроллера ESP32, драйвера мотора и блока аккумуляторов. Вследствие высокого натяжения гусениц и проскальзывания по доске при повороте в качестве двигателей используются мощные сервоприводы без обратной связи.

9.1 Разработка прототипов

Выполнил Савчук Андрей

Первый прототип робота состоял из тестовой платформы и магнитой гусеницы без

натяжителей для проведения тестов различных конфигураций механики робота. Экспериментальным путём было обнаружено, что обыкновенный сплав необимовых магнитов NdFeB N35-N38 не способен удерживать робота на доске. Было принято решение использовать сплав N42H, который обеспечивает большую магнитную силу. Во втором прототипе было установлено 2 версии моторов: маломощные DC моторы и миниатюрные шаговые двигатели формата Nema 11. Первый вид моторов не предоставлял необходимую мощность для прокручивания гусеницы, а шаговые двигатели превышали максимальный вес полезной нагрузки. В третьей ревизии модель робота была перенесена на модульную систему при помощи универсальных креплений (латунные вставки M3), что позволяло снимать и устанавливать различные модули на робота, а также значительно ускоряло процесс разработки и изготовления компонентов. Также были добавлены натяжители и стабилизаторы гусениц, которые значительно улучшали стабильность робота на доске. В финальном прототипе были установлены высокомощные сервоприводы для уменьшения веса при сохранении прежней мощности шаговых двигателей, а также был установлен страховочный трос для предотвращения удара робота о землю при заезде за край доски во время ручного управления с пульта. Также было обнаружено, что гусеницы, распечатанные TPU филаментом с жесткостью 75A по Шору, создают слишком высокий контакт с доской и способны растягиваться слишком сильно, в результате чего робот не способен выполнить некоторые повороты. Замена филамента на менее эластичный (95A) решила данные проблемы.

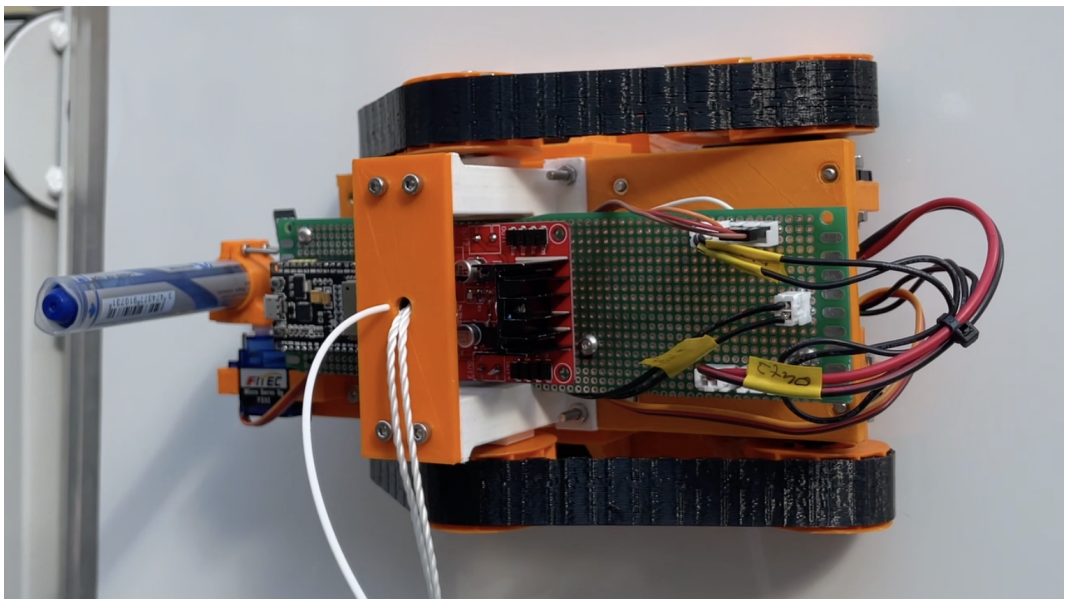

Рис. 9.1: Whiteboard bot на выставке робототехники, фото: EXTRA.HSE

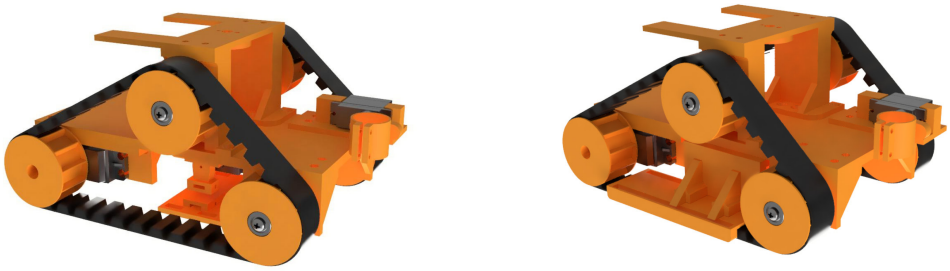

Рис. 9.2: Визуализация CAD модели робота в Fusion 360

10 Заключение

В [**enwiki:1146576754**] ходе проекта была разработана аппаратная платформа, имеющая возможность передвигаться по маркерной доске в любых направлениях с использованиям гусениц. Также было разработано программное обеспечение для работы и связи всех составных частей проекта.

Среди возможных вариантов развития проекта можно выделить следующие:

- Улучшение качества сегментации полученного изображения. Это позволит лучше распознавать маркерные надписи на доске и качественнее производить очистку.
- Учучшение качества построения траектории. Тогда робот сможет проезжать меньшее расстояние и тратить меньше времени на стирание с доски.
- Обучение модели для распознавания краев доски и последующий отказ от Aruco маркеров.
- Уменьшение размеров аппаратной платформы.
- Увеличение максимальной скорости аппаратной платформы и ее подвижности.

Список литературы

[1] Jacob Bandes-Storch. *Foxglove Studio — Visualizing and debugging your robotics data*. 2023. URL: https://foxglove.dev/studio (дата обр. 05.11.2022).

[2] G. Bradski. “The OpenCV Library”. В: *Dr. Dobb’s Journal of Software Tools* (2000).

[3] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest и Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[4] Richard Hartley и Andrew Zisserman. “Multiple view geometry in computer vision”. В: Cambridge university press, 2003. Гл. 2.4.

[5] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette и William Woodall. “Robot Operating System 2: Design, architecture, and uses in the wild”. В: *Science Robotics* 7.66 (2022), eabm6074. DOI: 10.1126/scirobotics.abm6074. URL: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074.