Санкт-Петербургский государственный университет
Кафедра компьютерных технологий и систем

Ковалев Дмитрий Александрович

Магистерская диссертация

Построение плана помещения при помощи автономного робота

Направление 010402
Прикладная математика и информатика
Магистерская программа прикладная математика и информатика в задачах цифрового управления

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Коровкин М.В.

Санкт-Петербург
2017

Содержание

Введение

В настоящее время индустрия автономных и полуавтономных платформ получила широкое развитие. Существует множество коммерческих продуктов, выполняющих самые разнообразные функции: от портативных роботов-пылесосов до беспилотных автомобилей. В данном исследовании рассматривается задача автономной навигации роботизированной платформы с использованием стереозрения. Для этого необходимо решить несколько нетривиальных задач: определение препятствий (obstacles detection), определение положения робота (pose estimation), построение карты местности (mapping) и построение маршрута (path planning). Данные подзадачи могут решаться независимо друг от друга или же синхронизировать свои данные — всё зависит от общей архитектуры алгоритма.

В настоящем исследовании предлагается для определения препятствий использовать стереозрение. Алгоритмы стереозрения позволяют определять расстояния до объектов, используя откалиброванную стереопару. В процессе калибровки происходит нахождение внутренних параметров камеры, их взаимного расположения и матрицы триангуляции (размера 4×4). После этого находятся преобразования, которые преобразуют изображения с камер так, чтобы плоскости камер совпали, а эпиполярная плоскость была ортогональна плоскостям камер (рис. 1). Для нахождения расстояния от стереопары до некоторой точки, необходимо определить координаты её проекций $A'_1(x_{left}, y_{left}), A'_2(x_{right}, y_{right})$. Так как после калибровки $y_{left} = y_{right}$, то поиск точки с одного изображения необходимо производить только вдоль оси x другого. Найденная глубина (диспаритет, disparity) $d = x_{left} - x_{right}$ позволяет определить реальные координаты объекта $(X/W, Y/W, Z/W)$:

$$\begin{pmatrix} X & Y & Z & W \end{pmatrix}^T = Q \begin{pmatrix} x_{left} & y_{left} & d & 1 \end{pmatrix}^T$$

После того, как в каждом конкретном случае известно расположение препятствий и данных о местоположении робота, можно провести построение карты местности. Метод одновременной локализации и построения карты (Simultaneous Localization and Mapping — SLAM) без внешних дан-

ных о положении робота позволяет построить карту местности. Но, к сожалению, этот подход сложен в настройке параметров, требует множества вычислительных ресурсов и его реализация выходит за предмет данного исследования. Вместо этого предлагается использовать встроенные в роботизированную платформу датчики (энкодеры).

Последним этапом в автономном построении карты местности является задача планирования маршрута до точки. Это задача решается при помощи теории графов. Существуют как точные решения данной задачи (алгоритм Дейкстры во взвешенном графе или поиск в ширину в невзвешенном),так и эвристические, которые различным образом оптимизируют нахождение пути (большинство из них основаны на алгоритме A*(рис. 2), являющимся эвристической модификацией алгоритма Дейкстры). Также существуют алгоритмы обзора местности без поиска пути до конкретных точек.

Таким образом, в данный момент существует большое количество разнообразных методов и подходов, позволяющих построить карту местности в автономном роботе. В данном исследовании предлагается подход, реализуемый на встраиваемом оборудовании и работающий в режиме реального времени.

Настоящая диссертация является дальнейшим развитием работы [1], где было реализовано построение карты местности при помощи платфор-

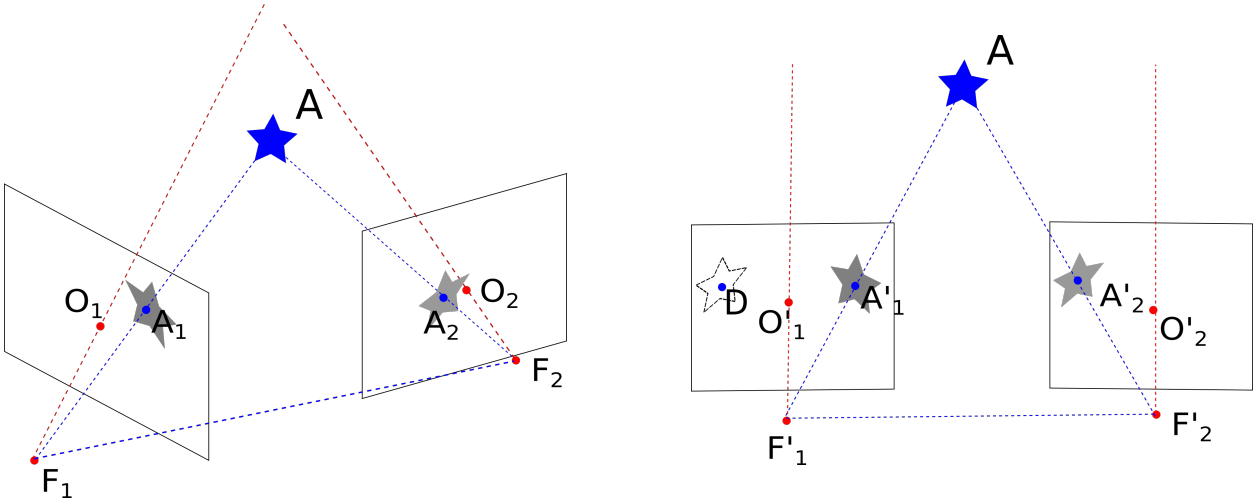

Рис. 1: Справа: изображения на неоткалиброванных камерах. OF_1, OF_2 - фокусные расстояния камер, F_1, F_2, A - эпиполярная плоскость. Слева: ректифицированные изображения, эпиполярная плоскость F'_1, F'_2, A ортогональна плоскостям камер, точки O'_1, O'_2, A'_1, A'_2 теперь лежат в одной плоскости, а координаты $O'_{1y} = O'_{2y}$. Глубина – $A'_{1x} - D_x$

мы, управляемой вручную. Так как вычисления велись на полноценном компьютере, а навигация шла вручную, то для реализации целей настоящей работы потребовалось значительная оптимизация алгоритмов и программного кода, также некоторые модули были модифицированы.

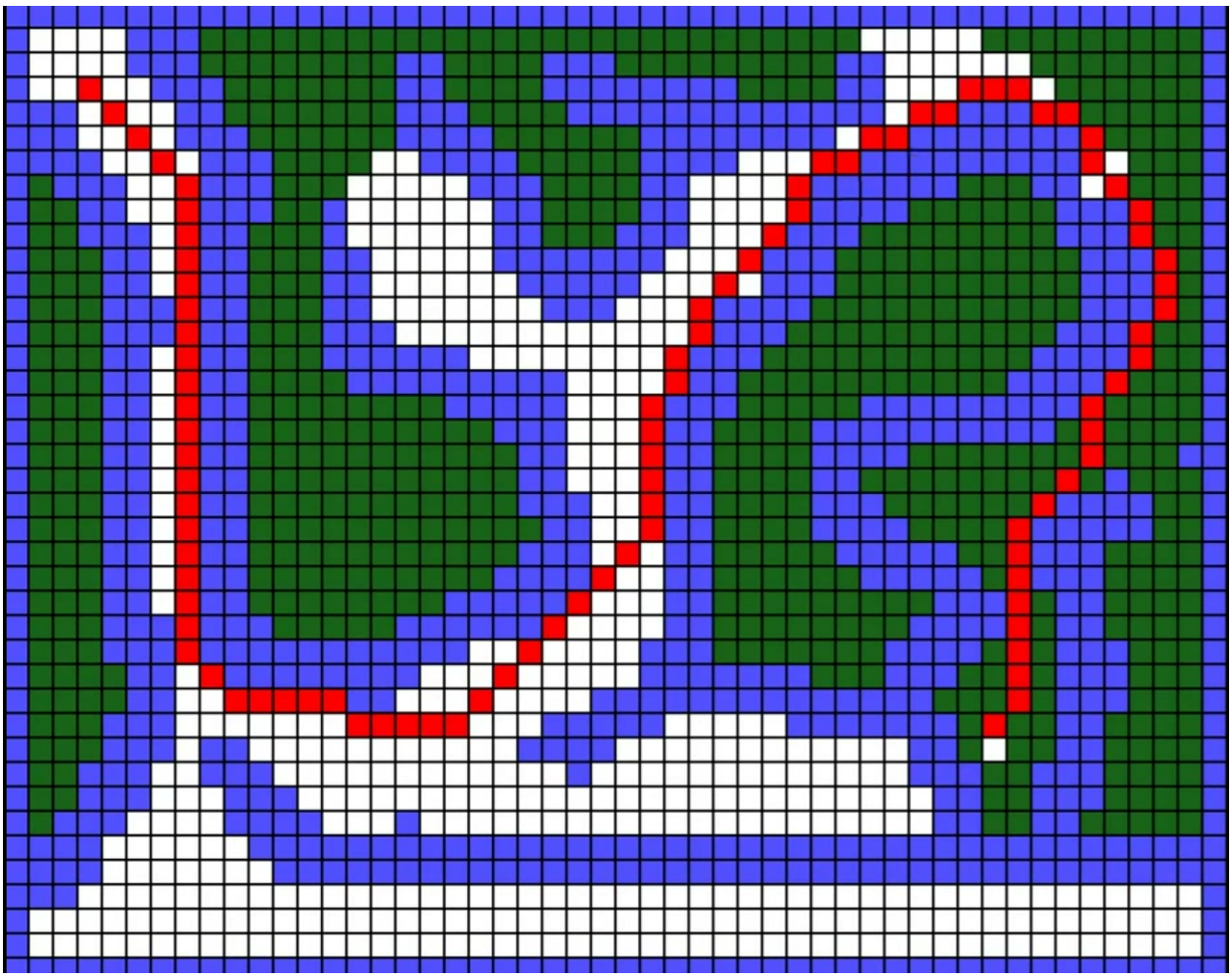

Рис. 2: Демонстрация работы алгоритма A*. Синий цвет - препятствия, зеленый цвет – свободные точки, черный цвет – начальная точка, красный цвет – конечная точка, белый цвет – проанализированные пиксели

Постановка задачи

Основной задачей данной работы является создание аппаратного и программного комплекса, выполняющего автономное построение карты местности в режиме реального времени. Выдвигаются общие требования ко всему комплексу:

- Все вычисления должны производиться непосредственно на платформе (на микроконтроллере и встраиваемом компьютере).
- Движение и построение карты местности должно происходить в режиме реального времени.
- Если местность является достаточно большой, с ровной подстилающей поверхностью и без слишком узких пространств (например, квартира или офисное помещение), то платформа должна полностью её исследовать и построить карту местности.

Весь комплекс можно разделить на четыре основных модуля: определение препятствий при помощи стереозрения, определение положения платформы при помощи встроенных энкодеров, построение карты местности и автономная навигация. Исходя из общих требований к комплексу, к каждому модулю определяется определенный набор требований.

Для определения препятствий требуется следующее:

- Калибровка стереопары: определение преобразований, выравнивающих эпиполярные плоскости и позволяющих определять реальные координаты препятствий по глубине.
- Построение карты глубины в режиме реального времени со стереоизображений разрешением 640×480.
- Замеченные препятствия должны быть отфильтрованы по высоте и размерам.

Определение положения платформы было реализовано в ходе работы [1], к этому предъявлялись следующие требования:

- Реализация считывания информации с энкодеров (при помощи прерываний).
- Интегрирование данных энкодеров.
- Реализация протокола отправки данных на микрокомпьютер.

Построение карты местности должно удовлетворять:

- Препятствия должны быть нанесены на карту, исходя из положения платформы.
- Карта должна быть восприимчива к изменениям (из-за изменения окружения или накопления ошибки энкодеров) и зашумленной информации.

Для реализации автономной навигации необходимо:

- Реализация построения маршрута до точки (например, отдаленной).
- Минимизация движения (в том числе и поворотов) в ходе построения маршрута.
- Управление исходя из маршрута и открытых в ходе движения препятствий.
- Реализация протокола между микрокомпьютером и микроконтроллером для управления платформой.

Обзор литературы

Проблема автономного управления, задач стереозрения и определения препятствий на сегодняшний день широко освещены в литературе. Подходы отличаются использованием различных датчиков, механизмами локализации и стратегиями исследования местности.

Одним из самых популярных подходов к построению карты местности является сочетание построения карты и локализации (SLAM). Подробное описание данного алгоритм содержится в работах [2], [3]. Он основан на расширенном фильтре Калмана, который позволяет по изменению расстояния до контрольных точек на карте определить смещение робота. Этот подход применимым ко множеству датчиков и платформ. Многие решения построения карт местности базируются на использовании сонаров или лазерных датчиков расстояния. Например, платформа Rossum [4] использует сонары для построения карты местности и локализации. В работах [5] представлено построение карты местности и локализации при помощи сканирующего лазера (рис. 3). В монографии [6] проанализированы различные датчики для задач построения карты (сонары, лазерные датчики расстояния, стереокамеры, обычные камеры, камеры глубины Kinect и TOF-камеры). В конечном счете было решено использовать TOF (Time-of-Flight) камеры на основе вычисления скорости отражения света от препятствий. Такие камеры являются на сегодняшний день крайне редкими и дорогостоящими, но позволяют достичь внушительных результатов (рис. 6). Также существуют SLAM-решения на основе камеры Microsoft Kinect

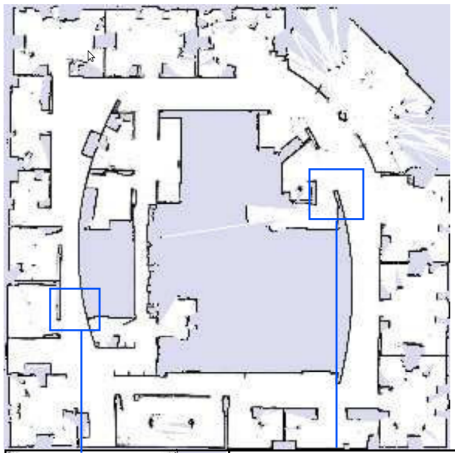

Рис. 3

Рис. 4

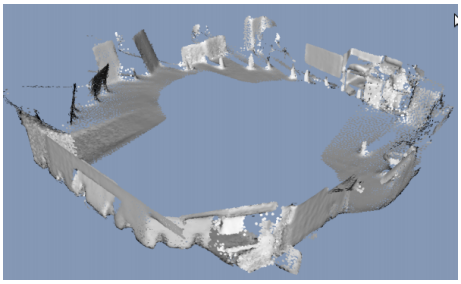

Рис. 5

Рис. 6

[7], сочетающей в себе видео камеру и инфракрасный проектор с КМОП-матрицей (позволяет при любом освещении получать трехмерное изображение). Такая камера позволяет получать точные 3D изображения местности (рис. 4). SLAM также применяется и совместно со стереозрением [8] (рис. 5), [9].

Несмотря на то, что SLAM позволяет достичь внушительных результатов в закрытой местности, на больших и открытых площадках он накапливает в себе большую интегральную ошибку. В открытой местности самым распространенным датчиком положения является спутниковая навигация (GPS). Обычно она применяется совместно с гироскопами, магнитометрами и акселерометрами (IMU-sensor). Такое сочетание датчиков было предложено в роботе Geomobil [10], где использовался сканирующий лазер для построения 3D-сцены. В работе [11] представлено комбинирование SLAM и IMU-датчиков.

Многие работы, связанные со SLAM и трехмерной реконструкцией обстановки ориентируются на подход восстановления структуры из движения (Structure from Motion, [12]. В работе [13] представлено построение карты побережья с вертолета при помощи этого подхода.

Для определения положения робота также используются и энкодеры, установленные на колесах и считающие их обороты. Энкодеры имеют высокую точность на малом промежутке времени, но подвержены накоплению интегральной ошибки и чувствительны к подстилающей поверхности. Применение энкодеров в задачах построения карт местности представлено в работах [14], [15].

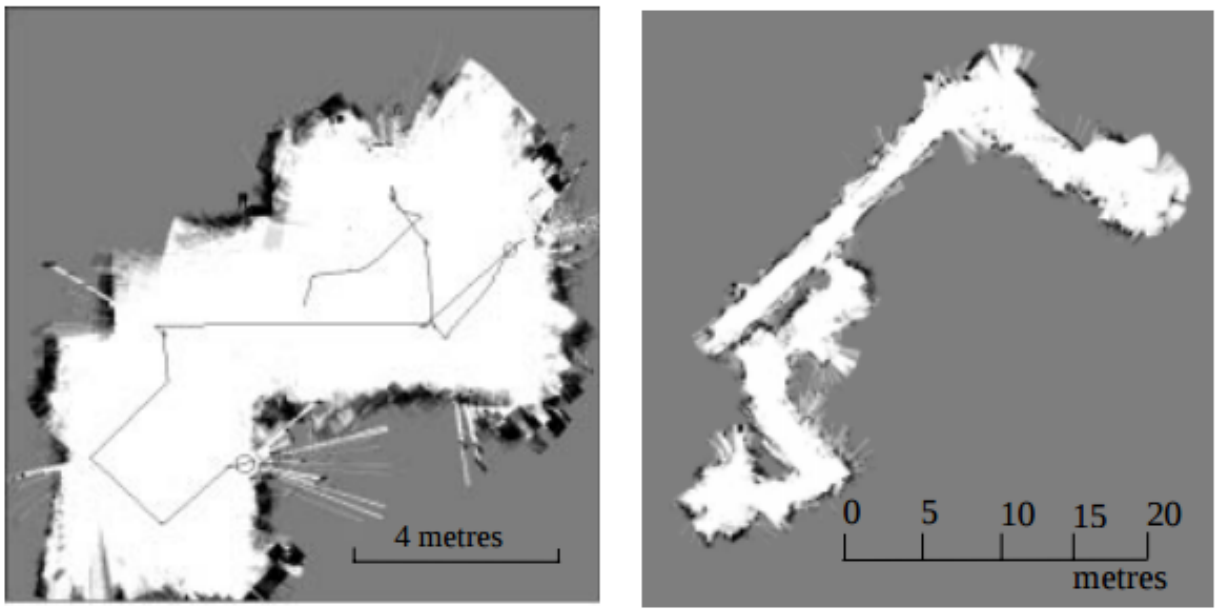

Рис. 7

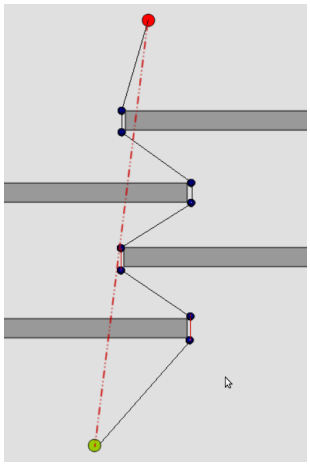

Рис. 8

Большинство представленных выше проектов производили лишь построение карты местности, в то время как управление платформами осуществлялось вручную. Задача автономной навигации и построения пути по готовой карте была сформулирована еще в середине XX века, тогда же были предложены первые решения, например, описанный во введении алгоритм A* [16]. В настоящее время существует множество алгоритмов планирования маршрута, в том числе для деформируемого окружения [17] специализированно для автономных платформ [18] (рис. 8), проводных роботов [19], для неизвестного окружения [20].

Автономная навигация была представлена во множестве проектов, использующих различные датчики. Как, например, сонары (Dolphin [21]), так и стереозрение: José [22], PIXHAWK [23], LAGR [24]. Препятствия, распознанные "триклопом"José, и построенная им карта местности изображены на рис. 7.

Нетривиальной и важнейшей задачей стереозрения является задача сопоставления объектов на двух изображениях (stereo correspondence problem). Одной из первых работ, анонсировавших данную проблему, является [25]. В данной статье описываются основные принципы стереозре-

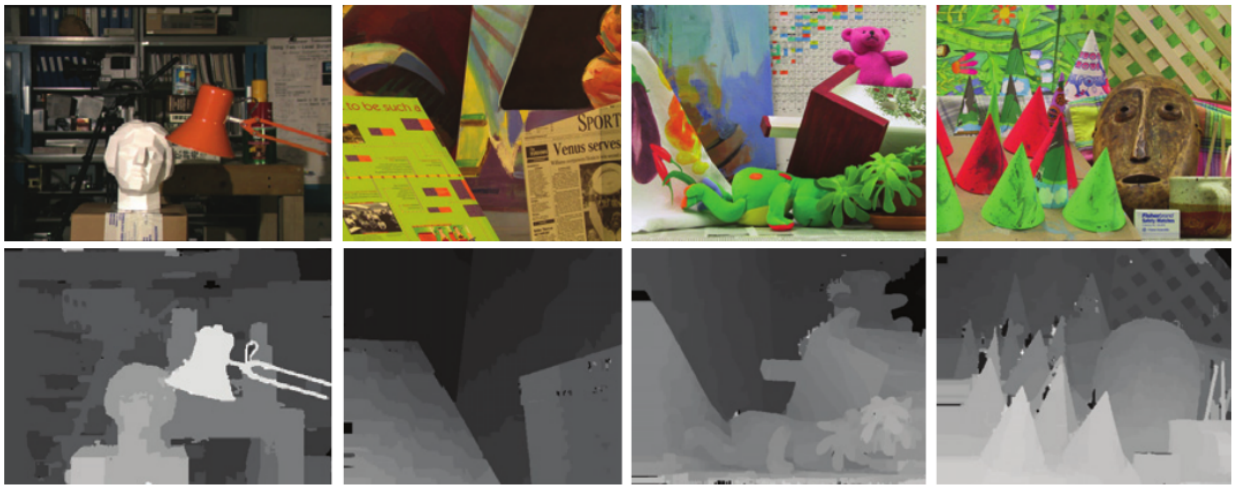

Рис. 9

ния на примере строения глаза человека. Также описываются основы нейрофизики и психологии зрения человека. Разработки первых алгоритмов, пригодных для вычислительных машин, велись с 1984 года (работы [26], [27], [28]). В работе [29] проводится сравнение различных подходов, разработан концепт портативных стереопар. Эти работы основывались на сопоставлении блоков изображений исходя из какой-либо либо меры (меры Хэмминга, среднеквадратичного отклонения и т.д.). Такой подход является классическим и до сих пор показывает свою эффективность, в частности, он реализован в нескольких свободных библиотеках компьютерного зрения (например, OpenCV).

С развитием технологий и математического аппарата компьютерного зрения и обучения, появилось множество новых подходов к решению данной проблемы. Созданная база алгоритмов [30] содержит более 170 алгоритмов, решающих проблему сопоставления стерео изображений. Среди них есть методы, основанные на теории графов [31], [32], адаптивного окна [33], [34], динамического программирования [35], [36]. Карта глубины, построенная в ходе работы [33], показана на рис. 9.

Глава 1. Аппаратное обеспечение

В настоящем исследовании была использована модификация роботизированной платформы, примененная в работе [1], с. 14]:

В качестве шасси используется Rover 5 robot platform. Шасси состоит из гусеничной платформы с четырьмя колесами, каждое из которых оборудовано мотором-редуктором и датчиком угла поворота (rotary encoder, далее просто "энкодер"). Для управления подачей питания на двигатели используется драйвер моторов Rover 5 4-channel controller. Управление драйвером осуществляется при помощи контроллера Arduino Uno, который производит обмен информации с компьютером с помощью USB кабеля по последовательному (serial) порту. Питание Arduino получает по USB-кабелю. Питание логических элементов драйвера осуществляется с Arduino, а питание моторов – с литий-полимерного аккумулятора (7.4 В, 2250 мА·ч). Также на платформе закреплены две камеры Logitech HD270, которые подключены к компьютеру с помощью USB-кабелей.

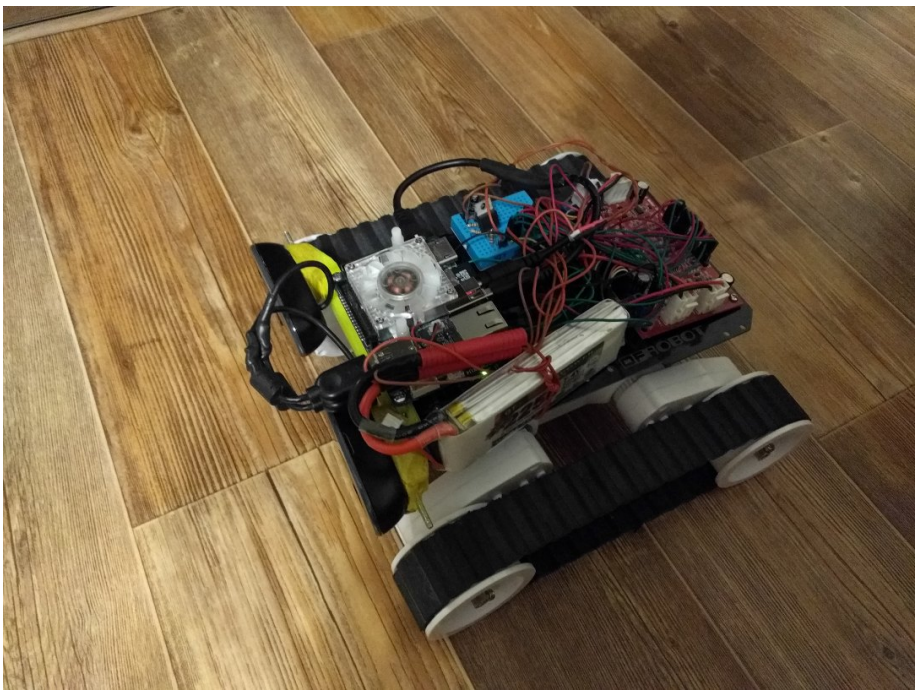

Рис. 10

В данной работе камеры были закреплены на меньшем расстоянии, что позволило добиться большей точности определения расстояния до близких препятствий и использовался аккумулятор на 7.4 В и 4600 мА·ч.

Так как все вычисления в данной работе должны производиться автономно, на борту, то основное вычислительное устройство находится непосредственно на платформе. Здесь был использован микрокомпьютер Odroid XU4, который работает на двухъядерном процессоре с четырьмя виртуальными ядрами (Samsung Exynos5422 Cortex^{TM}-A15 2Ghz). Для беспроводной связи с микрокомпьютером был установлен Wi-Fi модуль. Для питания компьютера использовался аккумулятор, подсоединенный через DC-DC преобразователь на 5В. Фотография платформы изображена на рис. 10.

Используемая платформа позволяет при помощи энкодеров определять положение и поворот робота, подробный алгоритм работы с энкодерами приведен в [1], с. 15-17].

Глава 2. Построение карты глубины

В данной работе для построения карты глубины использовался подход, предложенный в предшествующей работе [1]. Предварительно проводилась калибровка стереопары: вычислялись преобразования для ректификации и компенсации искажений, затем производилось сопоставление блоков изображений [1], с.19]:

> *Итак, после ректификации изображений имеется два исправленных нормализованных по яркости черно-белых изображения, на которых все проекции объектов находятся на одной горизонтальной линии. В качестве основного подхода к построению карты глубины был выбран алгоритм сопоставления блоков изображений: для каждой области на правом изображении проводится поиск области на левом изображении, исходя из какой-либо меры. В данной работе использована сумма абсолютных разностей, потому что другие меры (мера Хэмминга, разность квадратов, среднеквадратичное отклонение, сумма квадратов разностей) давали значительно худшие результаты. Это также подтверждается тем, что почти во всех последних работах в этом направлении использована именно сумма абсолютных разностей. Глубина для пары областей наносится на карту глубины соответственно координатам области с левого изображения. Таким образом, для некоторой правой области может быть найдено несколько оптимальных левых областей. Из них выбирается та, у которой значение меры меньше.*

В целях оптимизации алгоритма для слабых вычислительных устройств и реализации функционирования в режиме реального времени в данной работе сопоставление производилось по 9 точкам (рис. 11), и использовались блоки размером $\bar{b} = 7 \times 7$. На рис. 12 представлен результат работы алгоритма с данными параметрами. Построение кар-

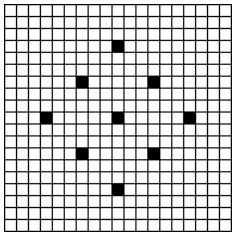

Рис. 11

ты глубины с низкой детализацией является достаточным для реализации поставленной в данной работе задачи.

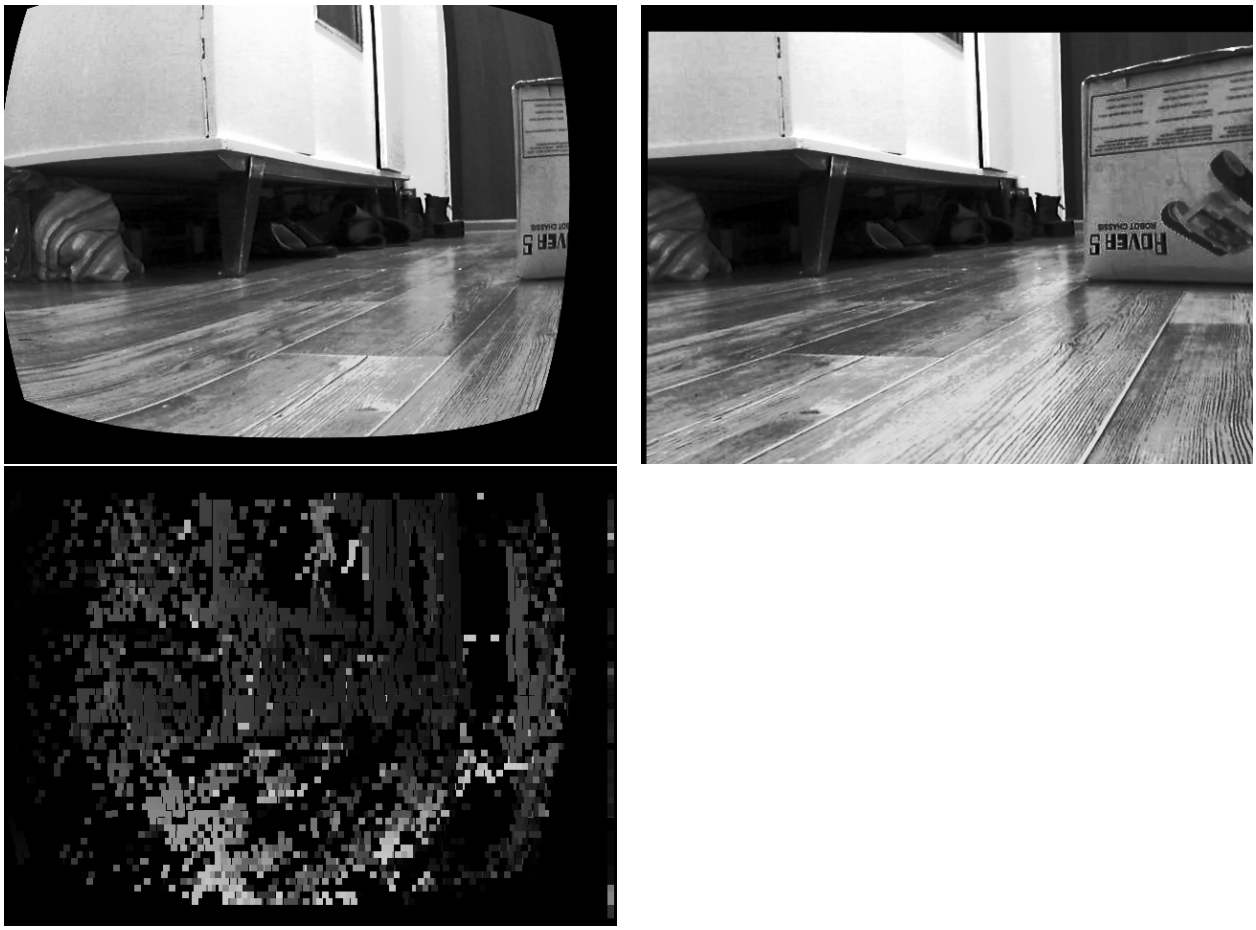

Рис. 12: Изображение с левой и правой камеры и карта глубины.

Глава 3. Определение препятствий

Основной принцип определения препятствий по карте глубины был взят из работы [1]. В ней предлагалось проводить триангуляцию для всех точек, для которых известна глубина, а затем проводить кластеризацию трехмерных объектов. После чего препятствия наносились на локальную карту препятствий.

В настоящем исследование предлагаются следующие модификации этого подхода:

- Проводить кластеризацию перед триангуляцией, основываясь на данных о глубине.
- При нанесении на карту препятствий использовать информацию об угле обзора двух камер, для того, чтоб отметить обозреваемые зоны, что будет в дальнейшем использоваться для построения карты местности.
- Карту препятствий строить в некотором масштабе, и считать, что в некоторой точке есть препятствие, только если на неё проецируется не менее $\bar{m}$ точек.

Таким образом, кластеризация объектов происходит по следующему алгоритму:

1. Выбираем точку, которая еще не кластеризована, добавляем в кластер.
2. Из кластера выбираем необработанную точку.
3. Добавляем все из 12 соседних точек, разница глубины с которыми не меньше некоторого порогового значения $\bar{d}$.
4. Производим шаги 2-3 до тех пор, пока все точки в кластере не будут обработаны.
5. Если количество элементов в кластере меньше заданного $\bar{k}$, то кластер отбрасывается.

6. Повторяем шаги 1-5, пока не будут кластеризованы все точки.

Таким образом, данный вид кластеризации позволяет найти большие группы точек с близкой глубиной, что обеспечивает фильтрацию объекта. Так как нас интересуют большие объекты, то кластеризация может быть достаточно жесткой (например, при значении параметров $\bar{d} = 2, \bar{k} = 2000$). Результат работы алгоритма с данными параметрами показан на рис. 13.

Затем для всех кластеризованных точек производится триангуляция и нанесение на локальную карту, фильтруя по вертикальной оси слишком низкие объекты (пол) и слишком высокие. Локальная карта, построенная в масштабе 1 пикс. $= 2$ см^2 и $\bar{m} = 10$, показана на рис. 14. Серый цвет показывает неизвестную зону, белый - препятствие, черный - зону, в которых препятствие отсутствует. В алгоритме считается, что за препятствием ничего не видно. Область зрения стереопары была вычислена вручную. Необходимо также учесть случай, когда ни одного кластера не было построено: в большинстве случаев это означает, что препятствие находится очень близко и видно только одной камерой. В этом случае карта глубины будет крайне зашумлена, и не будет выделено ни одного кластера. В этом случае на карту препятствий наносится "стена"на расстоянии чуть

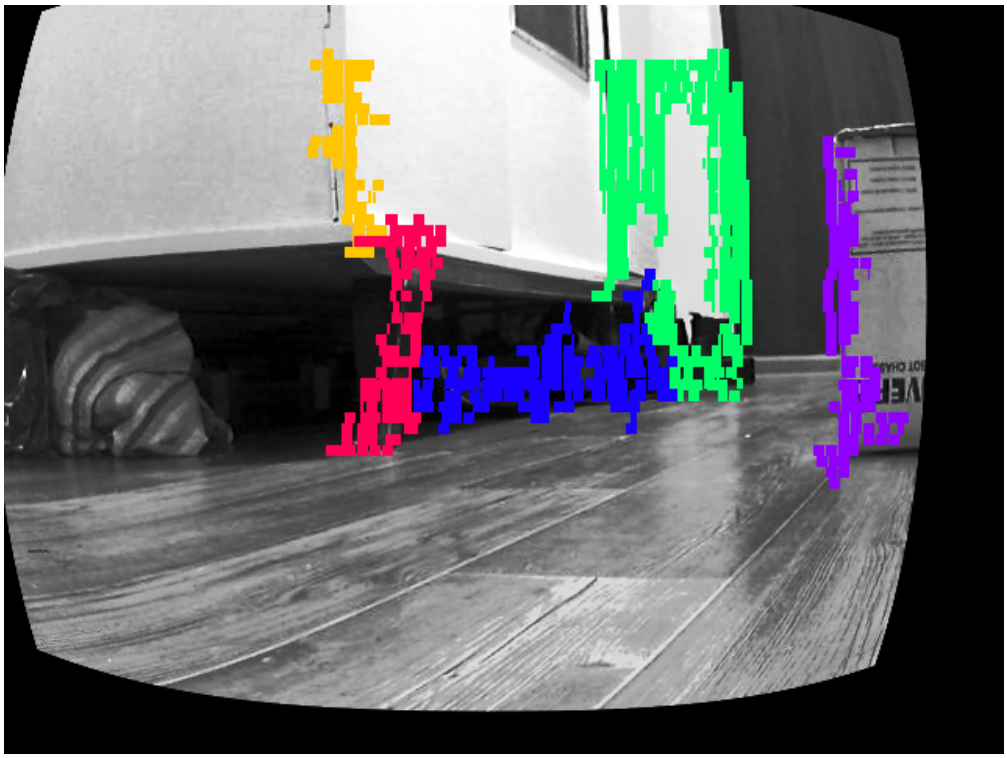

Рис. 13

Рис. 14

Рис. 15

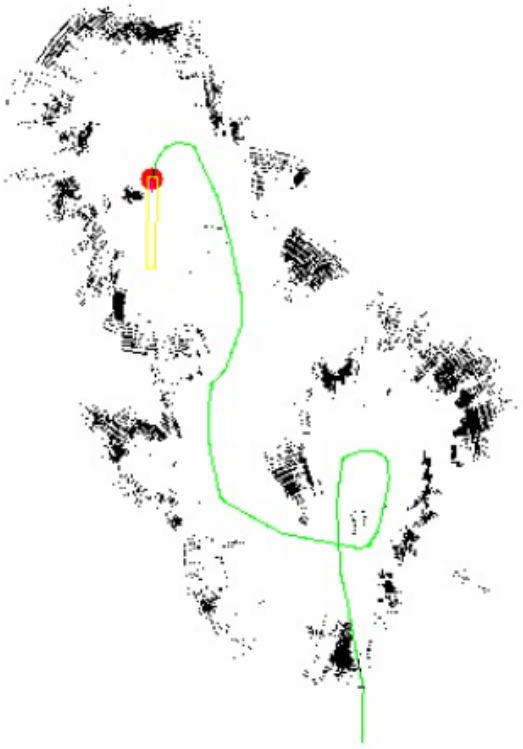

Рис. 16

большем мертвой зоны.

Зная расположение препятствий относительно робота и зная расположение робота (в данной работе - с энкодеров), можно построить карту местности. Для этого локальная карта переносится на глобальную, с учетом расположения робота по следующим правилам:

1. В локальной карте наличие препятствие считается за 1, отсутствие — за -1, слепые области — за 0.

2. Локальная карта прибавляется к глобальной карте местности с учетом угла поворота робота и его положения. Значения глобальной карты ограничиваются неким порогом $\bar{l}$.

3. Там, где значение карты больше 0, считается, что существует препятствие, где значение меньше 0 — препятствие отсутствует.

Карта местности (рис. 15), построенная при ручном управлении роботом, изображена на рис. 16). На карте видно, что есть существенное накопление ошибки угла поворота при использовании данных с энкодеров.

Глава 4. Автономное построение карты местности

Конечной целью данного исследования является реализация автоматического движения платформы при помощи построения карты местности. Существует множество стратегий исследования, которые не все подходят для задач реального времени. В данной работе реализовано движение к отмеченной точке на карте местности. Для исследования территории достаточно указать далекую точку, и платформа таким образом, исследуют любую замкнутую территорию. Серьезной проблемой является отсутствие неинтегральных данных о положении платформы, что вызывает проблемы с накоплением ошибки положения и поворота — что было продемонстрировано в ручном управлении платформой. Алгоритмы семейства SLAM или использование магнитометров, GPS или готовых решений ИНС помогают решить данную проблему, но это выходит за рамки настоящей работы.

Проблемы, связанные с тем, что карта местности постоянно обновляется, и порой может быть недостаточно достоверна, вызывают затруднение с простым следованием по маршруту с периодическим обновлением. Посему в рассматриваемом алгоритме предлагается сделать приоритетным моментальную навигацию для объезда препятствий, а найденный маршрут будет лишь задавать общий курс движения.

Автономное движение происходит по следующим принципам:

1. Строится маршрут до обозначенной точки (используя 4 направления на клетке карты).

2. Производится коррекция курса до некоторой точки маршрута (на опережение) на угол α_i.

3. Производится движение по прямой до тех пор, пока на определенном расстоянии $\bar{o}$ не будет найдено препятствие.

4. Производится поворот на небольшой угол ϕ в направлении обратном углу α_i.

5. Шаги 3-4 проводятся до тех пор, пока не истекло время $\bar{t}$ до следующей коррекции курса и не пройдено вперед хотя бы $\bar{M}$ раз.

6. Шаги 1-5 проводятся до тех пор, пока не будет достигнута точка или маршрут до неё не будет найден.

Такой подход позволяет производить навигацию в реальном времени, параллельно с определением препятствий и не отнимая серьезного количества ресурсов. Так же здесь решается вопрос о невозможности проезда платформы через найденный маршрут в силу ряда причин.

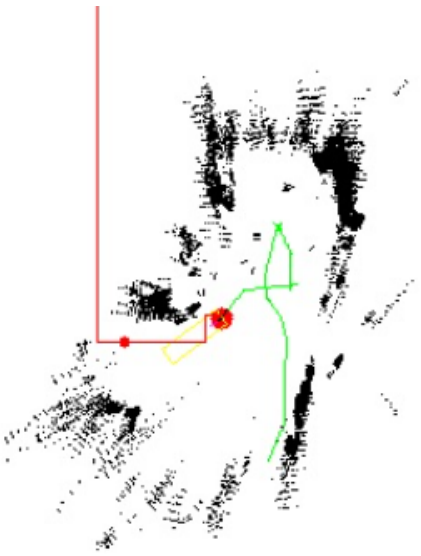

Рис. 17

Алгоритм построения маршрута основан на модификации A* для минимизации количества поворотов и быстрой реализации. Он основан на обходе графа в ширину (breadth-first search, BFS) по так называемой "макро-карте": карте местности, уменьшенной в $\bar{s}$ раз. Там, где на большой карте есть некоторое количество препятствий в сетке $\bar{s} \times \bar{s}$, там на соответствующей сетке "макро-карты"помечается препятствие. В алгоритм обхода в ширину была добавлена модификации для минимизации поворотов платформы. Общая его структура такова:

1. Создается пустая очередь элементов (x, y, dir), где dir - это номер

Рис. 18

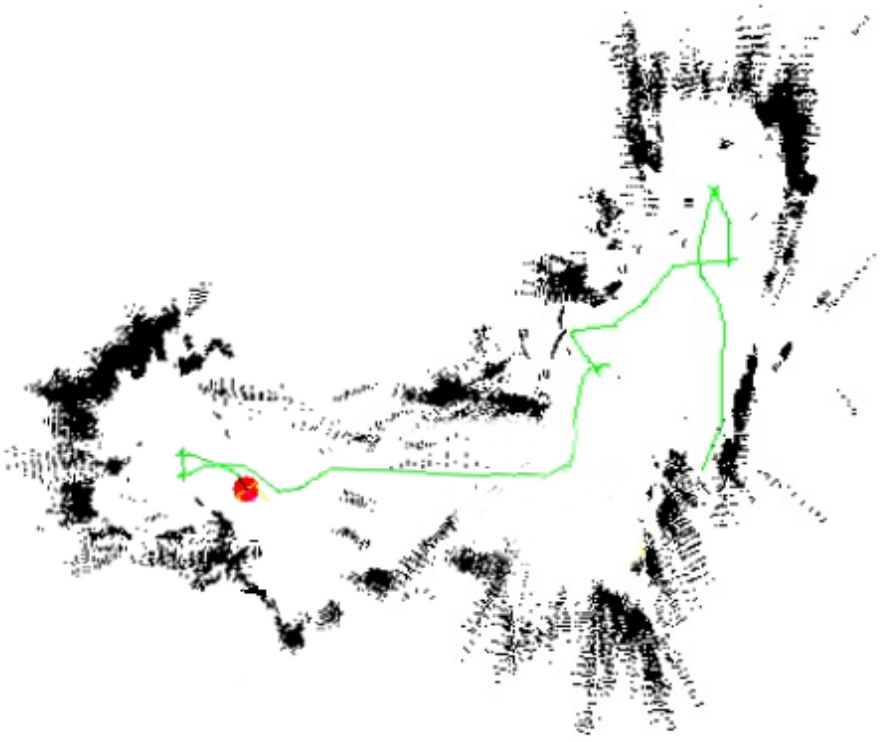

Рис. 19

одного из четырех направлений (вверх - 0, вправо - 1, вниз - 2, влево - 3).

2. В очередь добавляется элемент (x_0, y_0, dir_0), соответствующий текущей позиции и углу поворота робота в абсолютных координатах (так направлению 0 соответствуют углы поворота от $-45°$ до $45°$.

3. Для каждого элемента (x_i, y_i, dir_i) производится "обход"соседей (x_j, y_j, dir_j) в порядке увеличения (по модулю) dir_i.

4. (x_j, y_j, dir_j) добавляется в очередь, если точек с (x_j, y_j) нет в очереди и если в точке (x_j, y_j) на макрокарте нет препятствий.

5. Шаги 2-4 производятся до тех пор, пока все элементы в очереди не будут обработаны или не будет достигнута искомая точка.

Процесс построения карты местности (рис. 18) изображен на рис. 17: черные точки – препятствия, красная окружность – робот, зеленая – траектория движения, красная линия – построенный маршрут, красная точка –

точка траектории, исходя из которой корректируется курс, желтый прямоугольник – область с длиной $\bar{o}$, где анализируется возможность движения вперед. Готовая карта местности показана на рис. 19

Глава 5. Программная реализация

Программа на микроконтроллере выполняется параллельно в четыре основных потока:

1. Обработка и сохранение изображений с камер.
2. Построение карты глубины, определение препятствий и нанесение их на карту.
3. Считывание данных с микроконтроллера.
4. Автоматическое управление платформой.

Первый модуль получает изображения с камер, проводит нормализацию гистограммы и ректификацию изображений (с помощью ранее найденных параметров). Эти действия вынесены в отдельный поток для уменьшения задержки с камер.

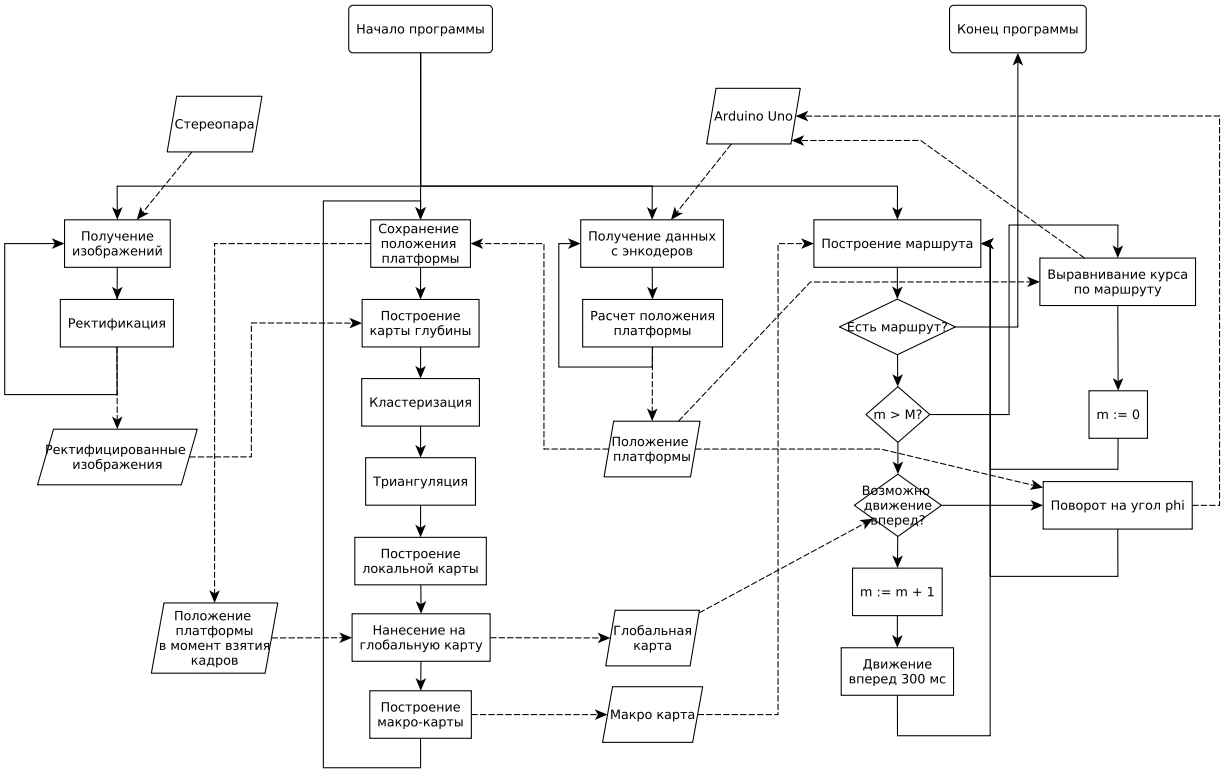

Рис. 20: Блок-схема программы на микроконтроллере. Условные обозначения: прямоугольник - действие, параллелограмм - данные в памяти или на устройстве, ромб - условный оператор, где правая стрелка - *нет*, нижняя стрелка - *да*.

Второй модуль получает ректифицированные изображения и положение робота в момент получения снимков. После этого проводится построение карты глубины (с использованием параллельного программирования для оптимизации), производит кластеризацию, триангуляцию и построение локальной карты препятствий. Затем локальная карта переносится на глобальную с учетом положения робота именно в момент, когда стереоизображения были сохранены. После этого обновляется макрокарта только в тех участках, где глобальная карта была изменена.

Третий модуль получает данные с микроконтроллера Arduino Uno и производит расчет положения платформы.

Четвертый модуль отвечает за автоматическое управление. Подробная его работа была рассмотрена в предыдущей главе.

Помимо этого существует возможность сохранения или передачи данных пользователю в отдельном потоке следующими способами:

- Сохранение карты на внутренний накопитель в формате PNG.
- Трансляция карты по каналу UDP в формате MJPEG.
- Демонстрация на виртуальный дисплей (например, по протоколу SSH).

В ходе данной работы были реализованы и вспомогательные утилиты, выполняющие данные задачи:

- Калибровка камер и сохранение параметров в формате YML.
- Автоматический подбор параметров для вычисления положения платформы по данным с энкодеров.
- Подбор дополнительных параметров для определения положений объектов: угла зрения стереопары, коэффициентов триангуляции.
- Дополнительные модули ручного управления платформой.

Программа исполняется в режиме реального времени. Ниже представлена таблица со значением параметров алгоритмов и времени их работы в конечной реализации ПО.

Алгоритм	Параметры	Время работы
Построение карты глубины	$\bar{b} = 7 \times 7$	200 мс
Кластеризация	$d = 2$, $\bar{k} = 2000$	80 мс
Триангуляция		25 мс
Построение локальной карты	$\bar{m} = 10$, масштаб: 1 пикс. = 2 см^2	2 мс
Нанесение на глобальную карту и построение макро-карты	$\bar{l} = 4$, $\bar{s} = 10$, размер карты: 900×900	2 мс
Поиск маршрута		100 мс
Автоматическое управление	$\bar{o} = 75$, $\bar{t} = 5$, $\bar{M} = 7$, $\phi = 15^\circ$	20 мс

Так как весь цикл обработки изображений занимается 3 кадра в секунду, перед каждым движением (вперед или поворотом) целесообразно выставить некоторую задержку для того, чтоб карта успела построиться. Экспериментально было найдено оптимальное значение задержки в 250 мс. Таким образом, автоматическое управление роботом и построение карты местности происходит в режиме реального времени.

Выводы

Данное исследование является продолжением работы [1]. Программно-аппаратное обеспечение было существенно доработано и оптимизировано для автономного построения карты местности на низкопроизводительном оборудовании:

- Оптимизирован алгоритм построения карты глубины.
- Усовершенствован алгоритм фильтрации препятствий, повышены его производительность и качество. Добавлен механизм определения объектов в слепой зоне.
- Построение карты местности доработано для реализации автономного управления.
- Реализован механизм автономного управления платформой, позволяющий строить карту местности без участия человека.
- Реализован механизм дистанционного просмотра хода построения карты местности.
- Аппаратный комплекс был существенно доработан для автономной работы.

Программное обеспечение на микрокомпьютере написано на языке C++ (стандарт C++14). Использовались библиотеки LibSerial [37] (для общения микрокомпьютера и микроконтроллера), libconfig++ [38] (для сохранения параметров), OpenCV [39] (для реализации работы с изображениями). Исходный код [40] содержит 3000 строк, и был переписан на 90% после публикации работы [1].

Несмотря на то, что поставленная задача в ходе данной работы была решена, существует несколько значительных проблем:

- Недостаточное качество определения препятствий — препятствия размываются, что оставляет на карте "шлейф препятствующий навигации.

- Определение положения при помощи энкодеров не является достаточно робастным, накопление ошибок делает карту нереалистичной.
- Полученное качество карты местности не позволяет применять оптимальный алгоритм построения маршрута.
- Недостаточно тщательный синтез поведения робота, если следование по маршруту невозможно.

Данные проблемы не являются неразрешимыми, и появились в силу того, что в проекте не были реализованы современные технологии, описанные в литературном обзоре. Следующие технологии помогут повысить качество и скорость построения карты местности:

- Внедрение SLAM.
- Вычисление движения через отслеживание движения объектов.
- Фильтрация "шлейфов"препятствий.
- Использование дополнительных систем позиционирования (например, IMU).
- Создание более оптимальной стратегии движения.

Существует большое количество подобных проектов, лишь малая часть которых была представлена в обзоре литературы. Среди преимуществ и новшеств данного исследования можно отметить:

- Крайне низкая стоимость оборудования (350 долларов).
- Демонстрация возможности определения препятствий по карте глубины низкого разрешения (с блоками 7×7).
- Реализация локализации платформы при помощи энкодеров — редко встречающийся способ позиционирования. Совместно с технологиями SLAM и IMU (с использованием фильтра Калмана) ожидается крайне высокая точность позиционирования.
- Оригинальный алгоритм фильтрации препятствий и определения объектов в "слепой зоне".

Заключение

В результате данного исследования был создан программно-аппаратный комплекс, производящий автономное построение карты местности при помощи стереозрения. За основу был взят комплекс, созданный в ходе предыдущей работы автора, в котором производилось построение карты местности роботом, соединенным с вычисляющим устройством – компьютером. Для создания автономного комплекса, выполняющего все вычисления на встраиваемом микрокомпьютере, существующие алгоритмы построения карты глубины, определения препятствий и нанесения их на карту были существенно дополнены и оптимизированы. Помимо этого, был реализован алгоритм построения пути для автономного движения, который учитывал найденные препятствия и построенную карту местности. Реализованный комплекс может автономно исследовать офисные и жилые помещения и строить двумерную карту местности в режиме реального времени. Пользователь может отслеживать движения платформы и ход построения карты по беспроводному каналу связи. Данная работа опиралась на новые исследования в предметно области, и в ней использовались современные информационные и инженерные технологии.

Список литературы

[1] Д. Ковалев. Построение карты местности с использованием стереозрения: бакалаврская работа // СПбГУ, СПб, 2015.

[2] Hugh Durrant-Whyte, Tim Bailey. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms // IEEE Robotics and automation magazine, 2006. Vol. 2.

[3] Hugh Durrant-Whyte, Tim Bailey. Simultaneous Localisation and Mapping (SLAM): Part II State of the Art // IEEE Robotics and automation magazine, 2006. Vol. 2.

[4] Varveropoulos V. Robot Localization and Map Construction Using Sonar Data // The Rossum Project. http://rossum.sourceforge.net

[5] Ruhnke M., Kümmerle R., Grisetti G., Burgard W. Highly accurate maximum likelihood laser mapping by jointly optimizing laser points and robot poses // Proceedings of 2011 IEEE International Conference on Robotics and Automation. 2011.

[6] Wang X. 2D Mapping Solutions for Low Cost Mobile Robot // KTH Royal Institute of Technology, 2013.

[7] Labbe M. Real-time appearance-based mapping in action at the Microsoft Kinect Challenge // IROS 2014.

[8] Diebel J., Reuterswärd K., Thrun S., Davis J., Gupta R. Simultaneous Localization and Mapping with Active Stereo Vision // Proceedings of 2006 9th International Conference on Control, Automation, Robotics and Vision. 2006.

[9] J. J. Little. Stereo vision SLAM: Near real-time learning of 3D point-landmark and 2D occupancy-grid maps using particle filters // Proceedings of IEEE/RSJ International Conference on Intellegent Robots and Systems, IROS. 2007.

[10] Alamus R., Baron A., Casacuberta J., Pla M., Sanchez S., Serra A., Talaya J. Geomobil: ICC land based mobile mapping system for cartographic data capture // Proceedings of the XXII International Cartographics Conference, 2005.

[11] Aghili F. 3D simultaneous localization and mapping using IMU and its observability analysis // Robotica, Cambridge University Press. 2010

[12] Jebara T., Azarbayejani A., Pentland A. 3D Structure from 2D Motion // IEEE Signal Processing Magazine, "3D And Stereoscopic Visual Communication"May 1999, Vol. 16. No. 3.

[13] James T. Dietrich. Riverscape mapping with helicopter-based Structure-from-Motion photogrammetry // Geomorphology, 2016, Vol 252, P. 144-157.

[14] Chikurtev D. Optimizing the Navigation for Mobile Robot for Inspection by Using Robot Operating System // Problems of Engineering Cybernetics and Robotics, Vol. 66, Sofia, 2015.

[15] Zhang Ji, Singh S. LOAM: Lidar Odometry and Mapping in Real-time // Robotics: Science and Systems Conference (RSS). Berkeley, CA, July 2014.

[16] Hart P. E., Nilsson, N. J., Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths // IEEE Transactions on Systems Science and Cybernetics SSC4, 1968, № 2, С. 100 — 107.

[17] Frank B., Becker M., Stachniss C., Burgard W., Teschner M. Efficient Path Planning for Mobile Robots in Environments with Deformable Objects // Proceedings of 2008 IEEE International Conference on Robotics and Automation. 2008.

[18] Hachour O. Path planning of Autonomous Mobile robot // International journal of systems applications, engineering & development, Issue 4., Vol. 2, 2008.

[19] Kim S., Bhattacharya S., Kumar V. Path Planning for a Tethered Mobile Robot // Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA). 2014.

[20] Ersson T., Xiaoming Hu. Path Planning and Navigation of Mobile Robots in Unknown Environments // Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2001.

[21] Elfes A. A sonar-based mapping and navigation system // Proceedings. 1986 IEEE International Conference on Robotics and Automation, 1986. Vol. 8, P. 1151 - 1156.

[22] Murray D., Little J. Using real-time stereo vision for mobile robot navigation // Autonomous Robots, Vol. 8, 2000.

[23] Fraundorfer F., Heng L., Honeggar D., Gim Hee Lee, Meier L., Tanskanen P., Pollefyes M. Vision-Based Autonomous Mapping and Exploration Using a Quadrotor MAV // Proceedings 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2012

[24] Learning Long-Range Vision for Autonomous Off-Road Driving // Journal of Field Robotics - Special Issue on LAGR Program, Part II. Vol. 9, 2009.

[25] D. Marr, T. Poggio. A computational theory of human stereo vision // Proceeding of Royal Society, 1979, B-204, P. 301-328

[26] Nishihara, H. K. Practical real-time imaging stereo matcher // Optical Engineering, 1984, Vol.23, N. 5, P. 536-545.

[27] Moravec, H. P. Visual mapping by a robot rover. // Proceedings of IJCAI, 1979, P. 598-600

[28] Kanada, T. et al. A stereo machine for video rate dense depth mapping and its new applications. // Proceedings of CVPR, 1996.

[29] Kurt Konolige. Small vision system: Hardware and implementation // Proceedings of the Intl. Symp. of Robotics Research (ISRR), 1997. P. 111-116.

[30] Middlebury Stereo Evaluation. http://vision.middlebury.edu/stereo/eval/

[31] Sebastien Roy, Ingemar J. Cox. A Maximum-Flow Formulation of the N-camera Stereo Correspondence Problem // Proceedings of Int. Conference on Computer Vision, 1998.

[32] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms // Proceeding of IJCV, 2002.

[33] K. Zhang, J. Lu, and G. Lafruit. Cross-based local stereo matching using orthogonal integral images // IEEE Transactions on Circuits and Systems for Video Technology, 2009.

[34] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister. High-quality real-time stereo using adaptive cost aggregation and dynamic programming // Proceedings of International Symposium of 3D Data Processing, Visualization, and Transmission, 2006.

[35] X. Chang, Z. Zhou, L. Wang, Y. Shi, and Q. Zhao. Real-time accurate stereo matching using modified two-pass aggregation and winner-take-all guided dynamic programming // Proceedings of International Symposium of 3D Data Processing, Visualization, and Transmission, 2011.

[36] L. Wang, R. Yang, M. Gong, and M. Liao. Real-time stereo using approximated joint bilateral filtering and dynamic programming // Journal of Real-Time Image Processing, 2014. Vol 9, N. 3, P. 447-461.

[37] LibSerial library. http://libserial.sourceforge.net/

[38] LibConfig library. http://www.hyperrealm.com/libconfig/

[39] OpenCV library. http://opencv.org

[40] GitLab repository. https://gitlab.com/dmkovv/robot-stereo-vision