МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА» (НГТУ)

Институт промышленных технологий машиностроения

Направление подготовки (специальность) 221000
(код и наименование)
мехатроника и робототехника

Направленность (профиль) образовательной программы

(наименование)

Кафедра автоматизации машиностроения

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

Студента Гришагина Михаила Владимировича группы М15-РТ
(Ф.И.О.)

на тему «Выделение и классификация объектов по трехмерным облакам точек,
(наименование темы работы)
получаемых с использованием двумерного сканирующего лазерного дальномера»

СТУДЕНТ:

_____ Гришагин М. В.
(подпись) (фамилия, и., о.)

_____
(дата)

РУКОВОДИТЕЛЬ:

_____ Егорушкин Е. О.
(подпись) (фамилия, и., о.)

_____
(дата)

РЕЦЕНЗЕНТ:

_____ _____
(подпись) (фамилия, и., о.)

_____
(дата)

ЗАВЕДУЮЩИЙ КАФЕДРОЙ

_____ Манцеров С. А.
(подпись) (фамилия, и.о.)

_____
(дата)

ВКР защищена _____
(дата)

протокол № _____

с оценкой _____

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Задача выделения объектов реального физического в данных, получаемых различными средствами измерений, в современном мире возникает очень часто. Например, интеллектуальные системы беспилотных автомобилей должны уметь определять и правильно классифицировать объекты окружающего пространства: пешеходов, дорожную разметку, другие автомобили, дорожные знаки. На производстве может применяться автоматизированный контроль качества, основанный на определении дефектов[1], или осуществляться совмещение деталей по опорным точкам деталей. В картографии может применяться распознавание объектов, таких как здания, деревья, кустарник или дороги для более полного представления об обрабатываемой области[45].

Одним из средств измерения являются лазерные сканирующие дальномеры, которые позволяют измерять расстояния до объектов. Задача измерения расстояний до удаленных объектов имеет большое значение, как для развития робототехники, так и для промышленности, научных исследований, военного дела, накопления знаний об окружающем мире, создания информационных и мультимедийных услуг различного назначения. Для решения этой задачи используется множество различных средств, выбор каждого из которых зависит от конкретных условий и требований. Одним из принципов построения таких приборов является лидар. Сканирующие лидары в системах машинного зрения формируют двумерную или трёхмерную картину окружающего пространства в виде облака точек.

Основное применение двумерных сканирующих лидаров – ориентация робототехнических платформ или автомобилей в пространстве, предотвращение столкновений с препятствиями, в то время как трехмерные лидары служат в основном для построения моделей промышленных объектов, зданий, сооружений или местности и имеют существенно большую точность вкупе с дополнительными функциями.

Зачастую возникает ситуация, когда возможностей двумерного лидара недостаточно, например, когда необходимо оценить габариты потенциального маршрута для робототехнической платформы не только по ширине, но и по высоте при невысоких требованиях по точности, а применение трехмерных сканирующих лидаров нецелесообразно из-за их существенно большей стоимости. Другим примером может служить необходимость определения объекта специфической формы, нахождение его позиции и характеристик.

Данные сканирования лидаров представляют собой так называемые облака точек. На первый взгляд кажется, что анализ трехмерных данных является более простой задачей, чем анализ плоских изображений, поскольку при получении фотографии значительная часть информации о пространственной структуре сцены теряется. Однако на практике ситуация с данными лазерного сканирования не такая оптимистичная: облако точек является слабо связанным разреженным набором отдельных точек, информация об объекте является неполной, объект может быть заслонен другими объектами. Часто имеет место шумовой сигнал, отсутствует цветовая информация. К тому же, существующие средства обработки визуальной информации ориентированы больше на плоские изображения.

Таким образом, актуальность данной работы вызвана необходимостью выделять и классифицировать объекты реального мира имея в своем распоряжении лишь недорогой двумерный сканирующий лазерный дальномер. Кроме того, в некоторых случаях может возникнуть потребность замены обычных оптических сенсоров (фото- и видеокамер) на лидар с сохранением логики обработки данных.

Исходя из начальных условий, можно выявить проблему исследования: насколько достоверно можно выделять и классифицировать объекты реального мира, если такая возможность вообще принципиально существует, используя только двумерный лидар.

Целью настоящей работы является определение возможностей сбора достоверной информации об окружающем пространстве в виде облаков точек с

использованием сканирующего лазерного дальномера и изучение эффективности обработки таких данных классическими методами анализа изображений.

Задачи исследования:

- разработать способ получения трехмерных данных измерения расстояний до окружающих объектов и реализовать его на практике;
- определить параметры сканирования, при которых достигается оптимальное соотношение между разрешением, быстродействием, точностью и потребляемой памяти;
- разработать и реализовать алгоритм представления трехмерных точек как плоского двумерного изображения с целью дальнейшего анализа;
- изучить работу методов обнаружения и классификации объектов на полученных данных.

Научная новизна данной работы заключается в том, что предпринята попытка реализации комплексного подхода к выделению и классификации объектов методами обработки двумерных изображений, полученных как проекция трехмерного облака точек на плоскость, при этом построение трехмерной картины выполняется с использованием двумерного лидара. Спецификой полученных двумерных изображений является низкое разрешение, вызванное низким разрешением сканирования лидара.

Настоящая работа имеет следующую структуру. В первой главе приведены общие сведения о применяемых технологиях, проведен анализ существующих работ по данной тематике, а так же выполнена более конкретная постановка задачи и предложено решение. Вторая глава описывает разработку аппаратно-программного комплекса, который позволяет получать трехмерные облака точек, соответствующие реальному окружению, от концептуальной модели до практической реализации. В третьей главе

описывается механизм преобразования облака точек в карту расстояний через проекцию, а также исследовано применение алгоритмов компьютерного зрения, которые обычно используются для плоских цветных изображений (фотографий), к полученным картам расстояний. В заключении сделаны выводы по проделанной работе и рассмотрены возможные направления дальнейших исследований.

ГЛАВА 1. ОБЗОР

Проблема выделения и классификации объектов окружающего пространства может быть разделена на две основные подзадачи:

- построение трехмерного облака точек;
- построение карты глубины и выделение характерных объектов.

В современных работах [1, 2], посвященных обработке карт глубины, основной целью является сопоставление цветных изображений (фотографий объектов) с трехмерными моделями этих объектов.

Другие работы [3, 4] описывают подходы к выделению и классификации объектов по картам глубины, полученным с помощью систем стереозрения (цветным фотографиям).

Подходы к получению трехмерного облака точек с использованием двумерного лазерного сканирующего дальномера изучены и описаны в работах [5-7].

Первый раздел настоящей главы содержит сведения о принципах работы сканирующих лазерных дальномеров. Во втором разделе проводится анализ работ, посвященных построению трехмерных облаков точек, а также существующих методов обработки полученной информации. В третьем разделе ставиться формальная задача, и предлагается схема ее решения.

1.1 Общие сведения о принципах работы сканирующих лазерных дальномеров

Лидар (транслитерация LIDAR англ. Light Identification Detection and Ranging — световое обнаружение и определение дальности) – это технология получения и обработки информации о расстоянии до объектов с помощью оптических систем, использующих явления отражения света и его распространения в прозрачных и полупрозрачных средах[10].

Лидар как прибор представляет собой, как минимум, активный дальномер оптического диапазона. Сканирующие лидары в системах машинного зрения формируют двумерную картину окружающего пространства. Принцип действия лидара не имеет больших отличий от радара: направленный луч источника излучения отражается от целей, возвращается к источнику и улавливается высокочувствительным приёмником (в случае лидара — светочувствительным полупроводниковым прибором); время отклика прямо пропорционально расстоянию до цели [12]. Для сканирования горизонта в одной плоскости применяются простые сканирующие головки. В них неподвижные излучатель и приёмник направлены в зенит; под углом 45° к горизонту и линии излучения установлено зеркало, вращающееся вокруг оси излучения. Для синхронизации мотора, вращающего зеркало, и средств обработки принимаемого сигнала используются точные датчики положения ротора, а также неподвижные реперные риски, наносимые на прозрачный кожух сканирующей головки [11].

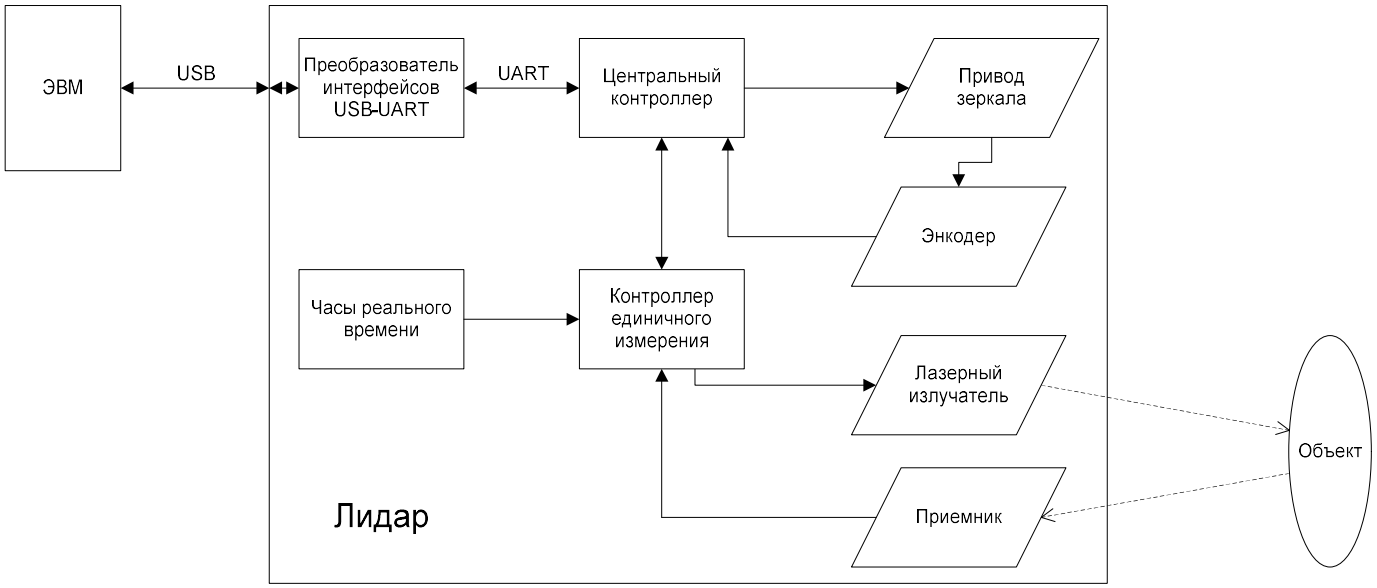


Рис. 1. Структурная схема лидара.

Общая структурная схема лидара представлена на рис.1. Контроллер единичного измерения передает в центральный контроллер время, которое

прошло между моментом излучения импульса и моментом его возврата его отражения. Центральный контроллер пересчитывает эти данные в расстояние между сканером и объектом, измеряемое в миллиметрах, и добавляет угол в градусах, на который было повернуто зеркало в момент излучения импульса. Эта информация кодируется в простом формате UART, который затем пакуется в USB-формат. Это сделано для совместимости с современными ЭВМ, которые имеют USB-порты, однако использование простых последовательных интерфейсов все реже поддерживается. Виртуализация последовательных портов позволяет работать с простыми устройствами ввода/вывода, применяя все стандартные приемы и методы работы с обычными последовательными интерфейсами.

Один пакет содержит измеренное расстояние в миллиметрах и угол в градусах, множество пакетов формируют представление об исследуемой области в диапазоне сканирования. На стороне ЭВМ при помощи специализированного ПО происходит извлечение получаемой информации, представление в требуемом виде, обработка и анализ.

1.2 Анализ существующих работ

1.2.1 Построение облака точек

Для получения трехмерных облаков точек при помощи двумерного лазерного сканирующего дальномера необходимо изменять угол сканирования. Существует два основных подхода к решению этой задачи: вращение лидара вокруг продольной оси (линия, проведенная от центра прибора до середины рабочей зоны) или вокруг поперечной оси. В работе [7] описывается сбор информации об окружающем пространстве при помощи непрерывного вращения лидара вокруг поперечной оси на 360^0. Примером другого подхода может служить работа [8], где лидар циклически наклоняется на угол к горизонту от -26^0 до 44^0 относительно горизонта. Оба подхода позволяют получать данные одного порядка точности, однако вращающийся лидар

снимает окружающее пространство полностью непосредственно вокруг оси только при вращении на 360^0, в то время как система с наклонением позволяет получать полные измерения перед собой при достаточно малых углах наклонения, что сказывается на скорости сканирования и объеме выходных данных. Так же описан другой способ [9], предполагающий использование вращающегося зеркала, которое отклоняет сканирующий луч.

Если лидар установлен на подвижную платформу, становится возможным получение трехмерной модели при неподвижном лидаре (относительно платформы)[13].

Другим аспектом получения расстояний до объектов в трех плоскостях является дискретность измерений в одной плоскости. В работах [7, 8] используется непрерывный механизм изменения положения лидара и отмечается, что этот подход предъявляет более жесткие требования к временным характеристикам оборудования и алгоритмов в плане сбора, передачи и обработки информации, однако упрощает управление.

Существуют промышленно изготавливаемые лазерные сканеры, позволяющие проводить сканирование в трех плоскостях с высокой скоростью и большим разрешением, однако такие системы чрезвычайно дороги и их применение в некоторых задачах неоправданно.

1.2.2 Выделение особенностей

Существует несколько подходов к выделению объектов по результатам сканирования, которые можно разделить на две большие группы: с использованием информации из трех измерений и алгоритмы, обрабатывающие проекцию трехмерной картины на плоскость.

В свою очередь, алгоритмы первой группы могут быть подразделены на алгоритмы, обрабатывающие облака точек и алгоритмы, обрабатывающие полигональные сетки. Отмечается, что задача выделения особенностей только по облакам точек является более сложной, поскольку они содержат недостаточно информации о нормалях к поверхности и связанности точек.

Обработка облаков точек описана в работах [14, 15, 16]. Методы, обрабатывающие полигональные сетки, требуют вычисления нормалей и построения самих полигонов, что является задачей, потребляющей значительное количество вычислительных ресурсов[15]. Обработка полигональных сеток представлена в работах [17, 18, 19].

Алгоритмы, обрабатывающие проекцию облака трехмерных точек распространены не так широко и используются в основном в задачах наложения цветных фотографий на трехмерные модели этих объектов[26].

Другие работы описывают построение карт глубины и выделение объектов по результатам стереосъемки[20, 22]. Много работ посвящено анализу RGB-D изображений, где каждый пиксель помимо значения интенсивности каждого цвета содержит расстояние до этой точки[21, 23, 24, 25]. Общим в этих работах является использование информации о цветах для более точного выделения объектов, а также высокого разрешения съемки.

1.3 Постановка задачи и предлагаемое решение

В настоящей работе предлагается применить методы поиска особенностей на плоских двумерных изображениях к проекции трехмерных точек на плоскость, где интенсивность каждого пикселя будет соответствовать расстоянию до исходной точки в пространстве.

Для сканирования пространства предлагается использовать схему качающегося лидара, аналогично конструкции, описанной в работе [8]. Предлагается применить модульную иерархическую архитектуру, поскольку основные функции могут быть реализованы независимо друг от друга под управлением основного алгоритма, достаточно соблюдать согласованность интерфейсов. Такой подход также позволяет упростить дальнейшее развитие и изменение комплекса: изменение реализации одного модуля никак не отражается на других модулях.

Предлагаемая схема построения аппаратно-программного комплекса, состоящего из следующих подсистем:

- ЭВМ с основной управляющей программой;
- лидар;
- микроконтроллер, выдающий команды управления приводом и отправляющий на ЭВМ угол фактического поворота вала;
- привод.

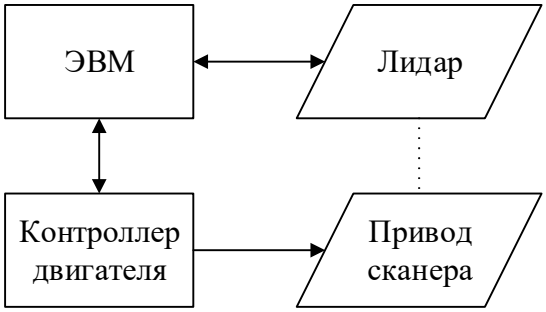


Рис. 2. Архитектура.

Выводы по главе 1

В настоящей главе был проведен анализ существующих подходов к получению трехмерных облаков точек и методов выделения особенностей. Большинство подходов предполагает использование полной трехмерной информации или RGB-D изображений.

Для получения облаков точек в данной работе предлагается схема качающегося лидара с модульно-иерархической архитектурой.

ГЛАВА 2. ПОЛУЧЕНИЕ ОБЛАКОВ ТОЧЕК

В данной главе описывается разработка аппаратно-программного комплекса, который позволяет получать трехмерные облака точек, соответствующие реальному окружению.

2.1 Аппаратная часть

В качестве привода был выбран шаговый двигатель с соответствующим драйвером, поскольку шаговый двигатель позволяет поворачивать вал на заданный угол с высокой точностью, а также неподвижно удерживать вал при отсутствии управляющих команд. Использование энкодера не предполагается, поскольку использование шагового двигателя позволяет знать, на какой угол был повернут вал. Вследствие того, что необходимый крутящий момент двигателя существенно превосходит прилагаемую нагрузку и скорость вращения невелика, вероятность срыва шага и возникновение ошибки маловероятна. Однако использование модельной архитектуры позволяет применить энкодер без необходимости изменять всю систему целиком, достаточно лишь скорректировать программу микроконтроллера.

Был выбран шаговый двигатель российской компании PureLogic R&D (ООО «Интех», г. Воронеж) серии PL42H48-2.4-4, поскольку этот двигатель имеется в наличии и его характеристики удовлетворяют заявленным требованиям. Основные характеристики приведены в таблице 1.

Таблица 1 – характеристики шагового двигателя PL42H48-2.4-4.

Угловой шаг, угловых минут	Число фаз	Ток фазы, А	Момент удержания, кг*см	Момент инерции, г*см^2
1,8 ± 5%	2	2.4	5,5	70

Для управления двигателем используется драйвер того же производителя серии PLD545-G3. Основные характеристики приведены в таблице 2.

Таблица 2 – характеристики драйвера шагового двигателя PLD545-G3.

Напряжение питания, В	18 … 48
Рабочий ток ШД, А	1,25 … 5
Деление шага ШД	1:4, 1:8, 1:16, 1:32
Частота встроенного генератора STEP, кГц	2
Частота сигнала STEP, кГц	макс. 350
Фронт	передний
Максимальная частота вращения вала ШД, об/мин	8200

Устройство имеет встроенные цепи защиты от КЗ обмоток ШД, от эффекта обратной ЭДС от ШД; встроенный автоматический компенсатор среднечастотного резонанса ШД; морфинг; защиту от переполюсовки напряжения питания и схему плавного пуска ШД; дампер; генератор частоты STEP. Драйвер работает со стандартным протоколом управления STEP/DIR/ENABLE или CW/CCW/ENABLE. Все управляющие входы драйвера оптоизолированы и совместимы с логическими уровнями 2,5 В, 3,3 В, 5 В. Так же модуль снабжен режимом AUTO-SLEEP, который включается при отсутствии сигнала STEP.

Для выдачи управляющих воздействий на двигатель используется плата Arduino UNO с микроконтроллером ATMEGA328P фирмы Atmel. Использование МК в составе Arduino, а не в самостоятельном виде выбрано из соображений удобства и универсальности при незначительно худшем быстродействии и более высокой цене. Ключевые для данной работы параметры приведены в таблице 3.

Таблица 3 – характеристики платформы Arduino UNO

Объем flash-памяти, кБ	2
Объем SRAM-памяти, кБ	2
Количество портов ввода/вывода	14
Интерфейсы	UART, USB(через виртуальный COM-порт)

Проектирование механической части осуществлялось в САПР Inventor. Отдельные элементы системы, такие как муфта, опорная площадка, ось качалки и втулки были выбраны из набора Tetrix. Модель аппаратного комплекса приведена на рисунке 4.

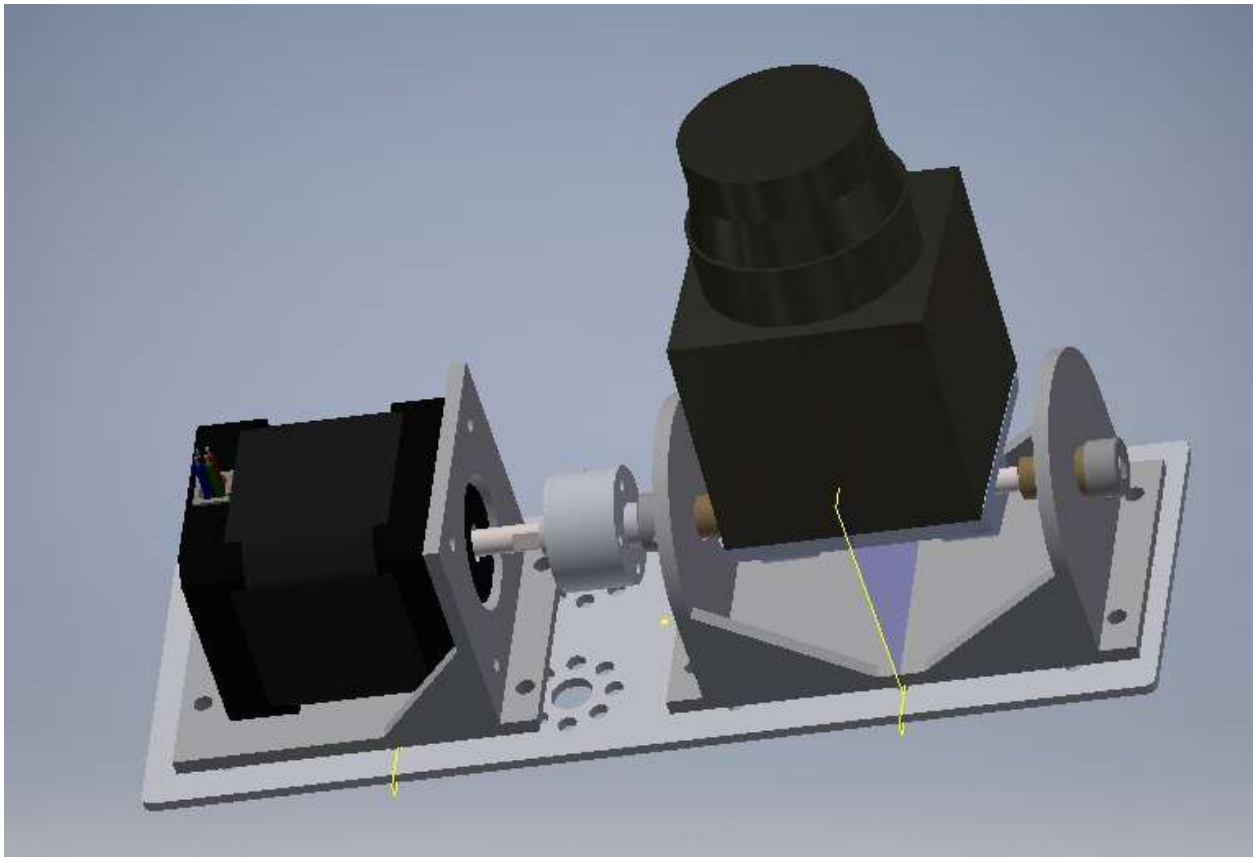


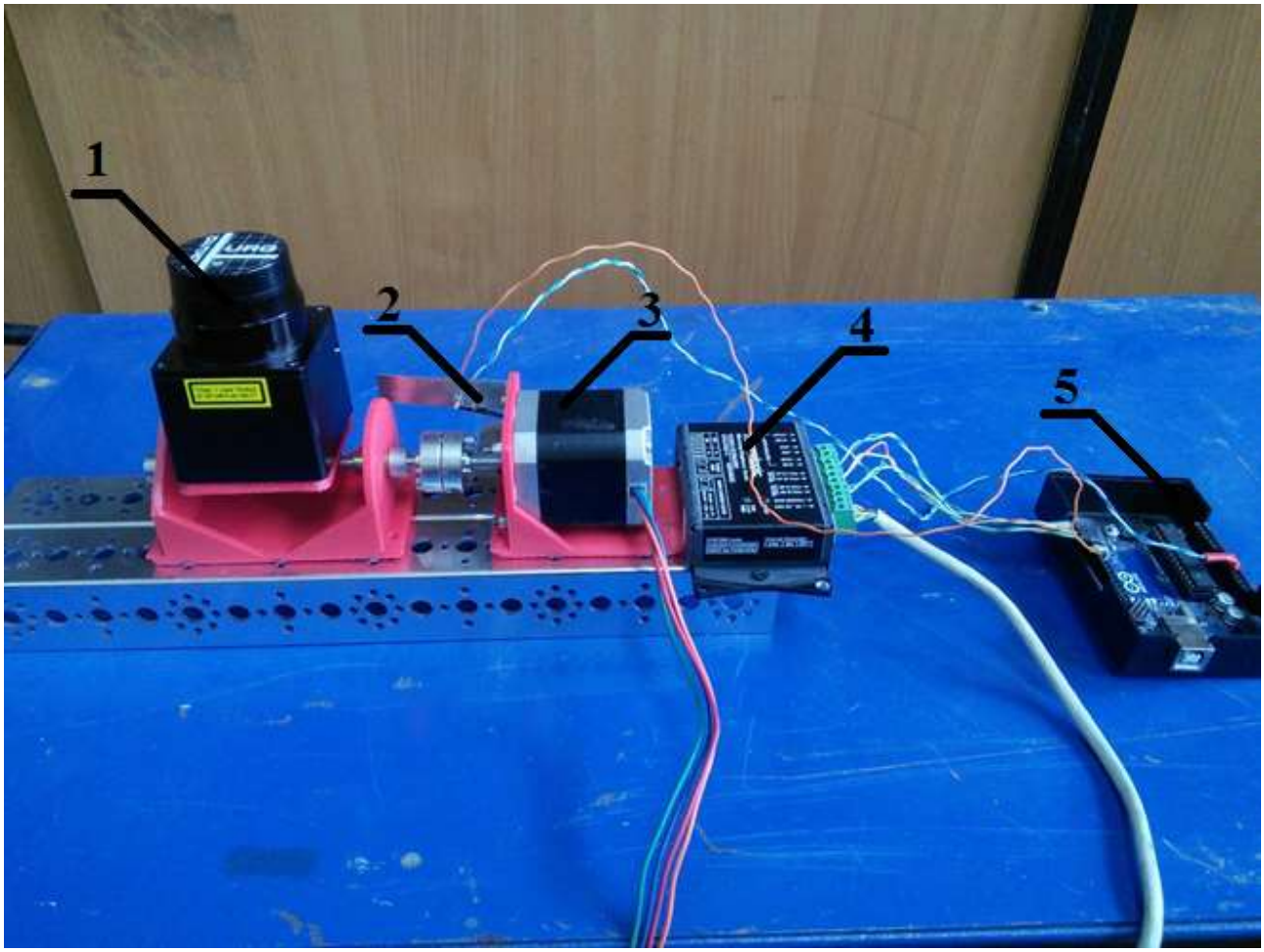
Рис. 3. Трехмерная модель

Опора двигателя, опора оси лидара и площадка крепления дальномера были изготовлены на трехмерном принтере методом наплавления нитей, при которой построение объекта идет за счет расплавления нити прочного

полимера, подаваемого на рабочую поверхность через экструдер. Этот метод изготовления был выбран из-за дешевизны сырья и непосредственно процесса печати, скорости изготовления и точности, удовлетворяющей требованиям данной работы.

Для калибровки двигателя и автоматического возврата в нулевое положение используется конечный выключатель и флажок, закрепленный на валу двигателя.

Внешний вид установки приведен на рисунке 4.



1 – лидар; 2 – конечный выключатель на кронштейне; 3 – шаговый двигатель; 4 – драйвер шагового двигателя; 5 – контроллер двигателя, платформа Arduino UNO

Рисунок 4. Установка в сборе

2.2 Программная часть

Сбор и обработка данных, а так же согласованность работы лидара и двигателя обеспечивается ЭВМ. Программа на ЭВМ написана в среде разработки LabVIEW компании National Instruments. Эта среда была выбрана из-за простоты и наглядности написания, редактирования и поддержки алгоритмов, интегрированности инструментов для проектирования, создания и разворачивания систем, широкого инструментария для всех требующихся задач, простоты интеграции с периферийным оборудованием и высокого быстродействия [27].

2.2.1 Алгоритм управления двигателем

Управление двигателем осуществляется при помощи двух подпрограмм, одна из которых функционирует на микроконтроллере, другая на ЭВМ в составе общего алгоритма. Обмен информацией осуществляется через протокол USB, посредством которого эмулируется работа простого последовательного интерфейса UART.

Программа для МК написана на языке Wiring, который является упрощенной версией языка C++. Управление двигателем осуществляется при помощи выдачи импульсов на один из портов ввода/вывода, каждый импульс соответствует одном шагу двигателя. Для установки вала двигателя в исходное положение используется микровыключатель. Алгоритм программы микроконтроллера приведен на рисунке 5. Исходный код приведен в приложении А. МК принимает через UART значение угла в минутах, на которое необходимо повернуть вал шагового двигателя. Это значение может лежать в пределах от 0 до 240 включительно. Значения, большие 240 используются как особые команды. Например, число 241 используется как команда возврата в исходное нулевое положение.

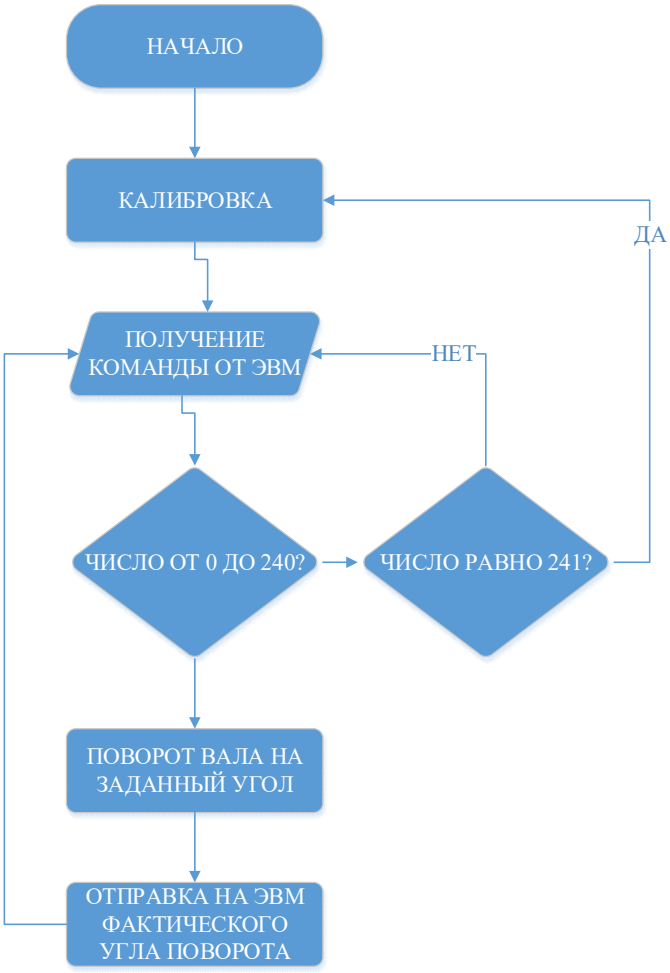


Рисунок 5. Алгоритм программы МК

Для выдачи углов поворота на МК и для получения фактического значения угла на ЭВМ в составе основного алгоритма написан отдельный модуль, отвечающий за обмен информацией между МК и ЭВМ. Инкапсуляция логики в отдельный модуль соответствует выбранному принципу модульно-иерархической структуры и позволяет вносить изменения в модуль без необходимости что-либо менять в основном алгоритме. Блочная диаграмма приведена на рисунке 6.

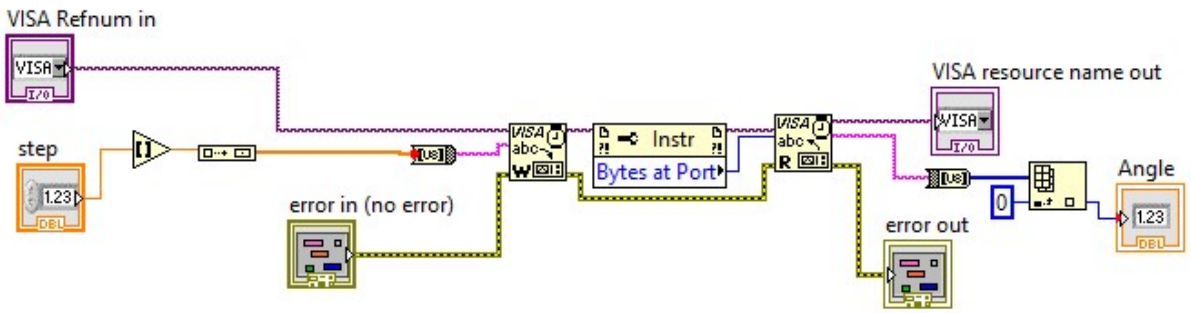


Рисунок 6. Блочная диаграмма алгоритма обмена информацией с МК

Модуль принимает в качестве входных параметров название виртуального COM-порта и количество угловых минут, на которое необходимо повернуть лидар на каждой итерации, а также код на общей шине ошибки. Угол кодируется в битовое слово и отсылается на МК. Затем происходит ожидание ответа, который представляет собой фактическое значение угла поворота лидара, закодированное в восьмибитовое слово. Фактический угол поворота может не совпадать с тем значением, которые было подано на вход, поскольку шаговый двигатель поворачивается кратно минимальному шагу. Кроме того, такая компоновка позволяет установить энкодер без необходимости редактирования алгоритма. Восьмибитовое число преобразуется в обычное целое и подается на выход.

2.2.2 Алгоритм управления лидаром

Алгоритм вынесен в отдельный модуль, как и в случае с двигателем. Блочная диаграмма алгоритма приведена на рисунке 7.

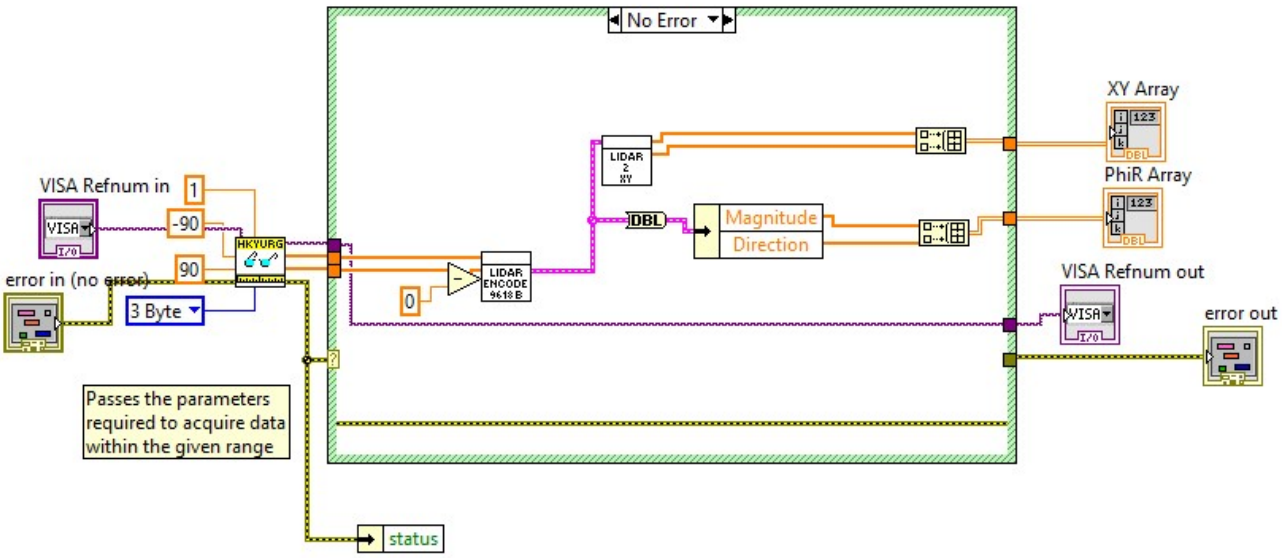


Рисунок 7. Алгоритм управления лидаром

Модуль принимает в качестве входных параметров название виртуального COM-порта, к которому подключен лидар, а также подключение к общей шине ошибки. Далее происходит установка параметров единичного сканирования в плоскости: пределов сканирования, количества измерений в одной точке и параметров кодирования результата. Затем происходит прием данных сканирования от лидара и формирование двух двумерных массивов, представляющих эти данные – один и них в полярной системе координат, другой в декартовой системе координат. Эти массивы, а так же код название COM-порта и код ошибки (в случае ее возникновения) подаются на выход модуля.

2.2.3 Обобщенный алгоритм работы программного-аппаратного комплекса

Сбор и обработка данных, а так же согласованность работы лидара и двигателя обеспечивается ЭВМ. Блочная диаграмма алгоритма приведена на рисунке 8. В начале работы алгоритма происходит инициализация параметров, устанавливается связь с лидаром и платой управления ШД.

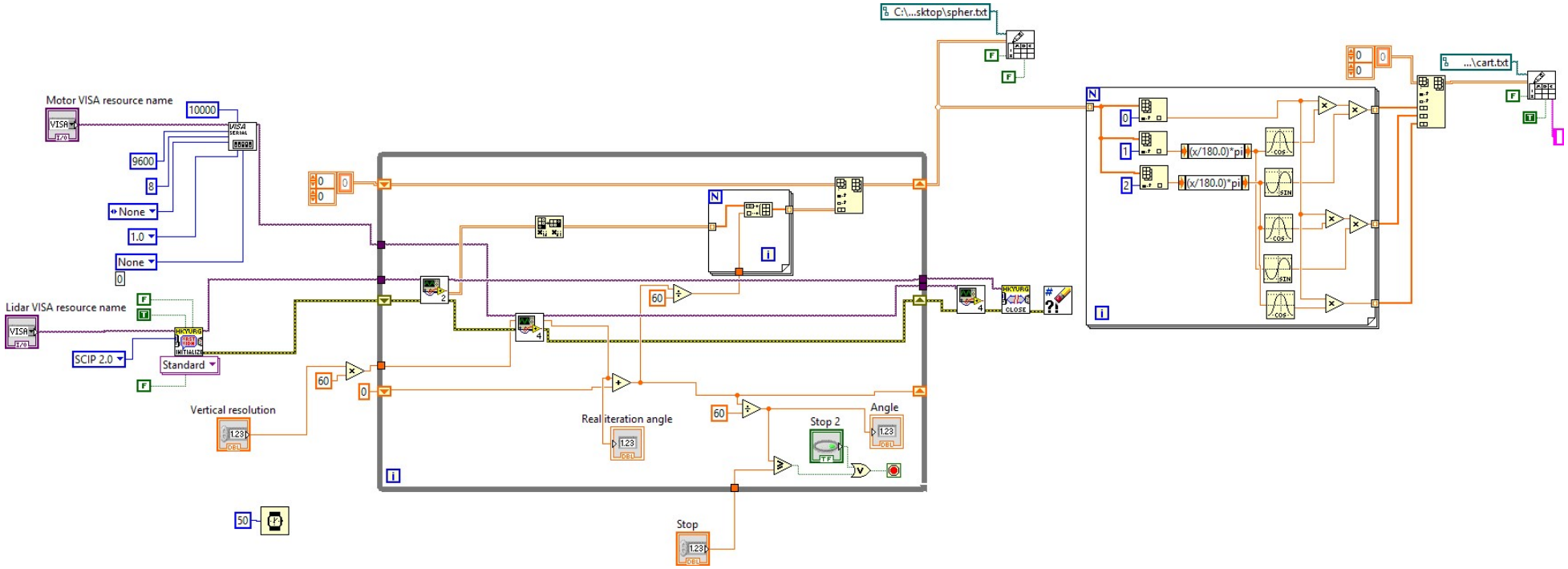


Рис. 8. Блочная диаграмма основного алгоритма.

Затем начинается основной цикл, состоящий из трех основных шагов:

- выдача команды на поворот лидара на заданный угол, в результате значение, представляющее собой угол поворота платформы к горизонту (угол места);
- сканирование в одной плоскости, в результате получается двумерный массив точек;
- добавление к массиву значения угла поворота платформы к горизонту;
- добавление полученного массива к общему массиву точек;
- проверка условия останова.

При достижении угла поворота, равного или большего заданного условия остановки, на МК отсылается команда 241, возвращающая установку в исходное нулевое положение.

Полученный массив данных представляет собой трехмерное облако точек в сферической системе координат. Для возможности построения трехмерных моделей происходит перевод из сферической системы координат в прямоугольную по соотношениям $x = r \cdot \sin\theta \cdot \cos\varphi$, $y = r \cdot sin\theta \cdot cos\varphi$, $z = r \cdot cos\theta$, где r – расстояние от оси зеркала лидара до препятствия, θ – азимут, φ – угол места. Каждая строка двумерного массива представляет собой одну точку с координатами x, y и z соответственно. Этот массив записывается на диск. На рисунке 9 изображена визуализация облака точек, полученного в одном из экспериментов. Визуализация проводилась при помощи программы MeshLab.

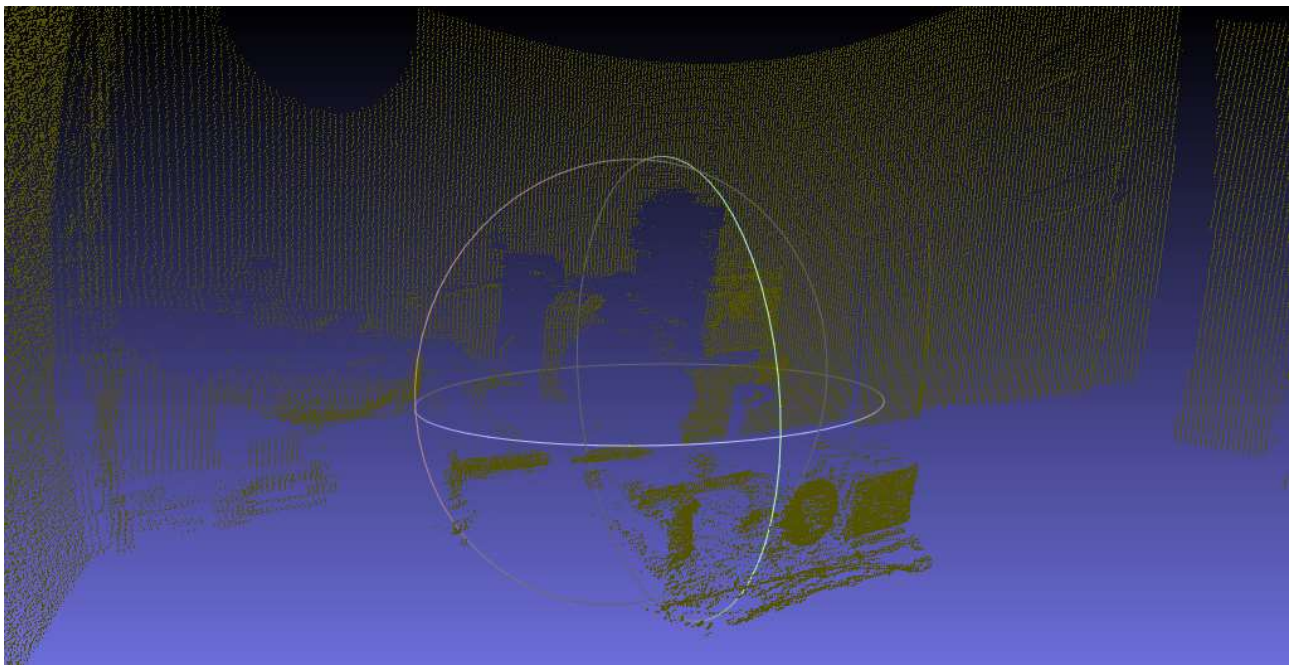


Рисунок 9. Визуализация облака точек

Выводы по главе 2

В главе 2 был описан программно-аппаратный комплекс, позволяющий производить сканирование пространства в пределах 270^0 по горизонтали с разрешением до $0{,}36^0$ и от -20^0 до 20^0 по вертикали с разрешением до $0{,}038^0$.

Результат сканирования представляет собой облако точек, записанное в виде двумерного массива, каждая строка которого представляет собой одну точку с координатами x, y и z соответственно. Дополнительно записывается на диск облако точек в сферической системе координат.

ГЛАВА 3. ВЫДЕЛЕНИЕ ОСОБЕННОСТЕЙ

В данной главе описан механизм преобразования облака точек в карту расстояний через проекцию, а также исследовано применение алгоритмов компьютерного зрения, которые обычно используются для плоских цветных изображений (фотографий), к полученным картам расстояний.

3.1 Построение карты глубины

3.1.1 Теоретическая модель

В трехмерной компьютерной графике карта глубины – это изображение, которое содержит информацию о расстояниях до поверхностей от точки наблюдения. В рамках данной работы под картой глубины понимается проекция облака точек на плоскость, где точка $P(x, y, z)$ проецируется в точку $P_c(u, v, i)$, где u, v – координаты точки на плоскости, i – расстояние от точки $P(x, y, z)$ до наблюдателя. Расстояние i вычисляется по формуле $i = \sqrt{x^2 + y^2 + z^2}$.

Для проекции точек на плоскость применяется модель камеры с точечной диафрагмой [28, 29]. Схема проекции показана на рисунке 10.

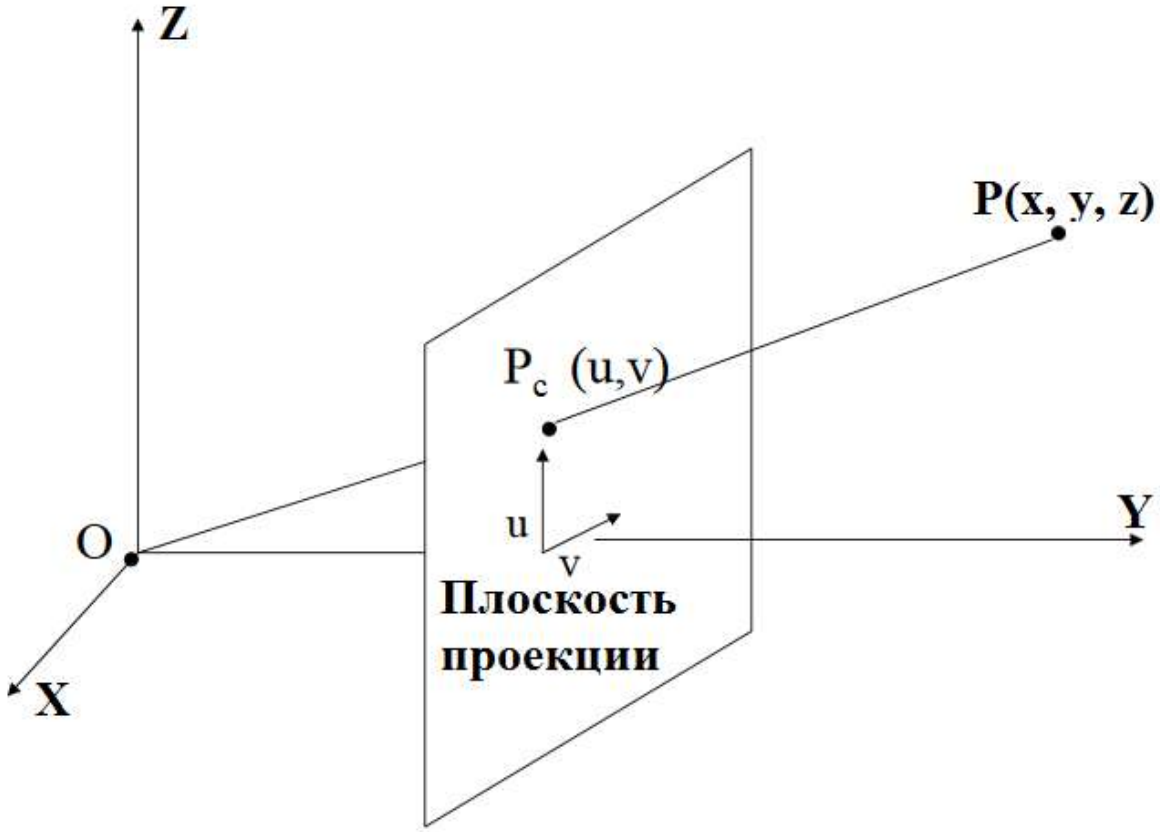


Рисунок 10. Проекция точки *P(x, y, z)* на плоскость

Модель камеры с точечной диафрагмой определяется центром камеры *O*, главной осью — лучом начинающимся в центре камеры и направленным туда, куда камера смотрит, плоскостью изображения — плоскостью на которую выполняется проецирование точек, и системой координат на этой плоскости. В такой модели, произвольная точка пространства *P(x, y, z)* проецируется на плоскость изображения в точку P_c, лежащую на отрезке *OP*, который соединяет центр камеры *O* с исходной точкой *P*. Важно отметить, что эта модель позволяет получить лишь координаты точки P_c, но не дополнительные характеристики, такие как цвет или расстояние.

Формула проецирования выглядит как $P_c=AP$, где *P* – однородные координаты точки трехмерного пространства, P_c - однородные координаты точки плоскости, *A* – матрица камеры размером 3x4. Матрица *A* выражается следующим образом $A = KR[I \mid -c] = K[R \mid t]$, где *K* — верхняя треугольная матрица внутренних параметров камеры размера 3 × 3 (конкретный вид

приведен ниже), R — ортогональная матрица размера 3 × 3, определяющая поворот камеры относительно глобальной системы координат, I — единичная матрица размера 3 × 3, вектор $\boldsymbol{c}$ — координаты центра камеры, а $t = -Rc$.

Поскольку все измерения проводятся относительно центра лидара, т.е. центр совпадает с началом координат O, и главная ось сонаправлена оси OY, оси координат на плоскости имеют одинаковый масштаб, а центр изображения имеет нулевые координаты, то матрица A будет равна $A=K[I|0]$, где

$$K = \begin{matrix} f & 0 & 0 \\ 0 & 1 & 0, \\ 0 & 0 & f \end{matrix}$$

где f – фокусное расстояние, которое является настраиваемым параметром и влияет на размер проекции. Результаты варьирования величины f показаны в следующем разделе.

3.1.2 Практическая реализация

Алгоритм проекции облака точек на плоскость выполнен так же в LabView. Исходными данными для алгоритма является трехмерное облако точек, представленное в виде двумерного массива данных, где каждая строка содержит координаты x, y и z в декартовой системе координат с центром O, который совпадает с осью вращения зеркала в лидаре. Кроме того, алгоритм вычисляет значение расстояния i для каждой точки. Блок-схема алгоритма приведена на рисунке 11. В приложении Б приведена более подробная блочная диаграмма алгоритма в среде LabView. Блок выделения геометрических фигур будет рассмотрен подробнее в следующем разделе.

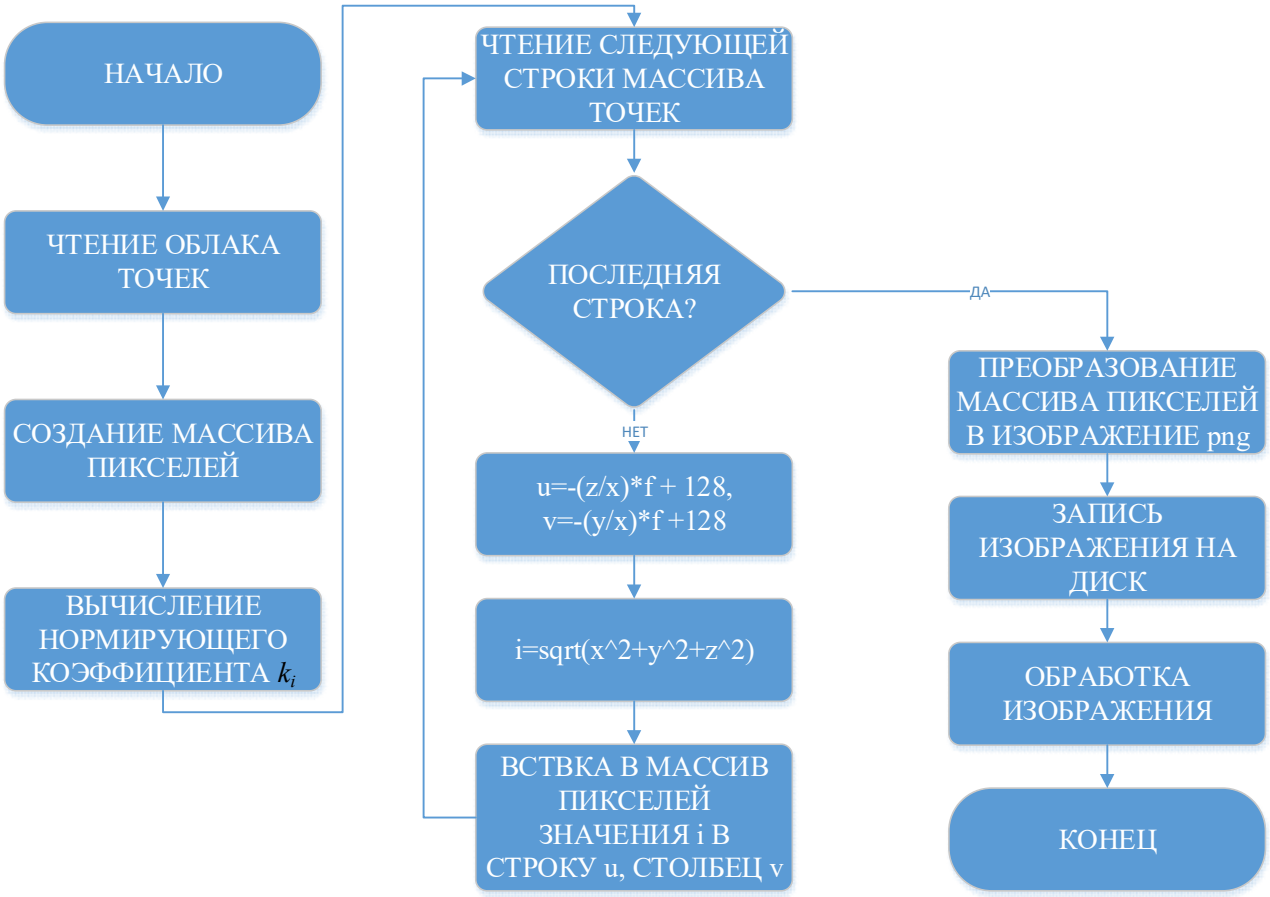


Рисунок 11. Блок-схема алгоритма проекции трехмерных точек на плоскость.

В начале работы алгоритма создается двумерный массив размеров 256x256, где каждая ячейка представляет собой один пиксель в результирующем монохромном изображении. Значение, которое хранится в ячейке, представляет собой интенсивность (яркость) пикселя. Значение интенсивности может быть от 0 до 255, где 0 соответствует черному цвету, а 255 – белому, а промежуточные значения являются градациями серого. Далее вычисляется нормирующий коэффициент яркости k_i, который используется в дальнейшем для того, чтобы абсолютные значения расстояний до точек преобразовать в относительные значения интенсивности. Нормирующий коэффициент вычисляется как $k_i = \frac{\max(i)}{256}$. Затем начинается основной цикл проекции, в котором происходит чтение строки, которая представляет собой точку в трехмерном пространстве, и вычисление ячейки массива, в которую это точка должна быть спроецирована. Строки массива пикселей соответствуют

оси *Ov* плоскости проекции, а столбцы – оси *Ou*. Вычисление точек u и v производится по формулам:

$$u = -\frac{z}{x} \cdot f + 128,$$

$$v = -\frac{y}{x} \cdot f + 128,$$

где *f* – фокусное расстояние. После вычисления всех данных происходит запись значения *i* в ячейку массива с индексами (*u, v*), а затем преобразование массива пикселей в изображение формата PNG и запись этого изображения на диск. На рисунке 12 приведен пример карты глубины, а на рисунке 13 - фотография на цветную фотокамеру, снятая с того же ракурса.

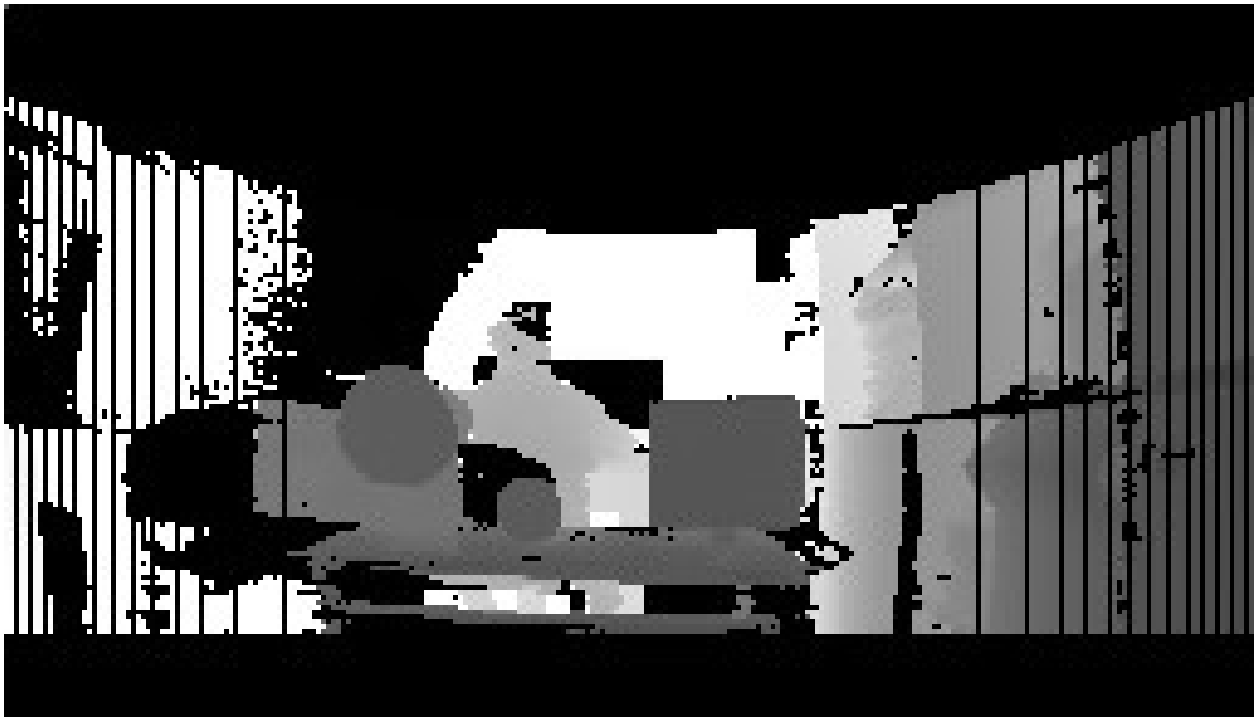


Рисунок 12. Карта глубины

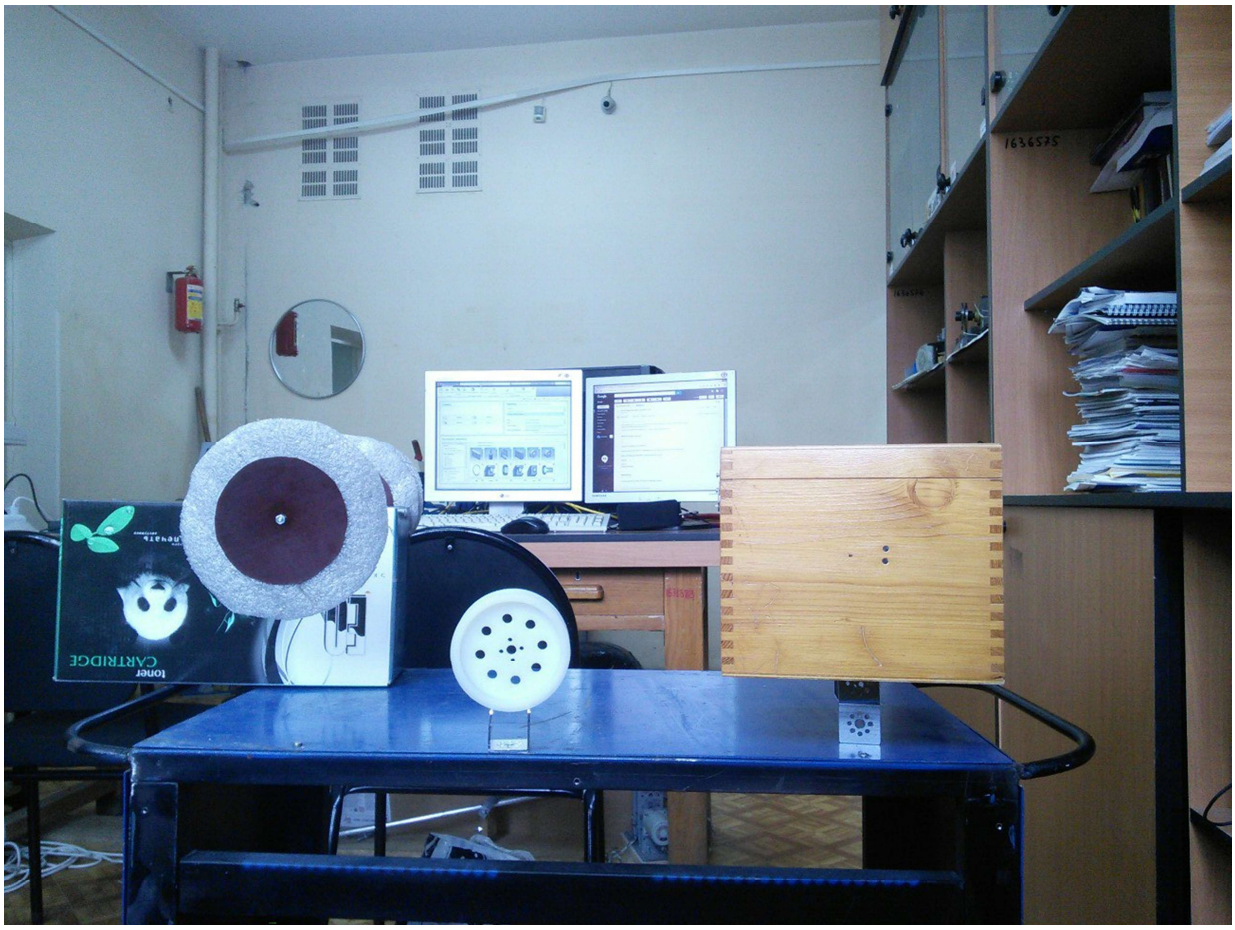


Рисунок 13. Фотография

Рисунки 14 и 15 демонстрируют другие объекты съемки.

Рисунок 14. Карта глубины

Рисунок 15. Карта глубины.

На рисунках 12, 14 и 15 хорошо заметны вертикальные черные полосы, возникающие у краев изображения. Причина их возникновения – неравномерность при проекции на плоскость.

Результаты, показанные на рисунках 12, 14 и 15, были получены при значении $f = 140$. Для других значений f результаты показаны на рисунках 16, 17 и 18. Все изображения размером 256x256 пикселей, в данной работе приведены их увеличенные копии для лучшей наглядности.

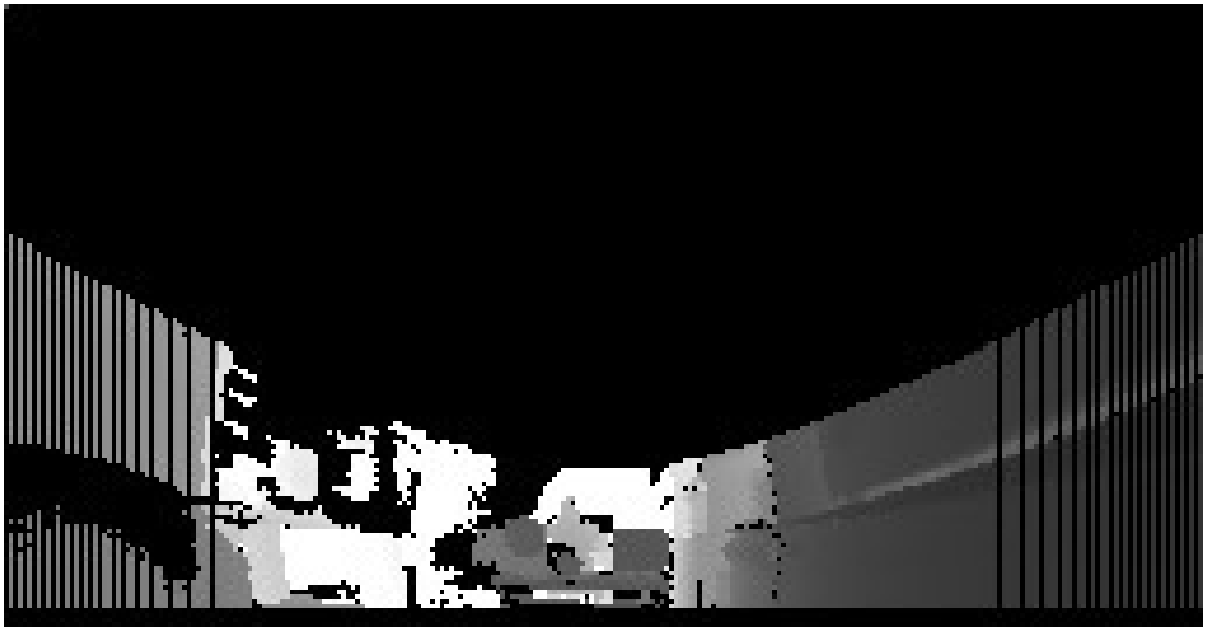


Рисунок 16. Карта глубины при f = 50.

Рисунок 17. Карта глубины при f = 100.

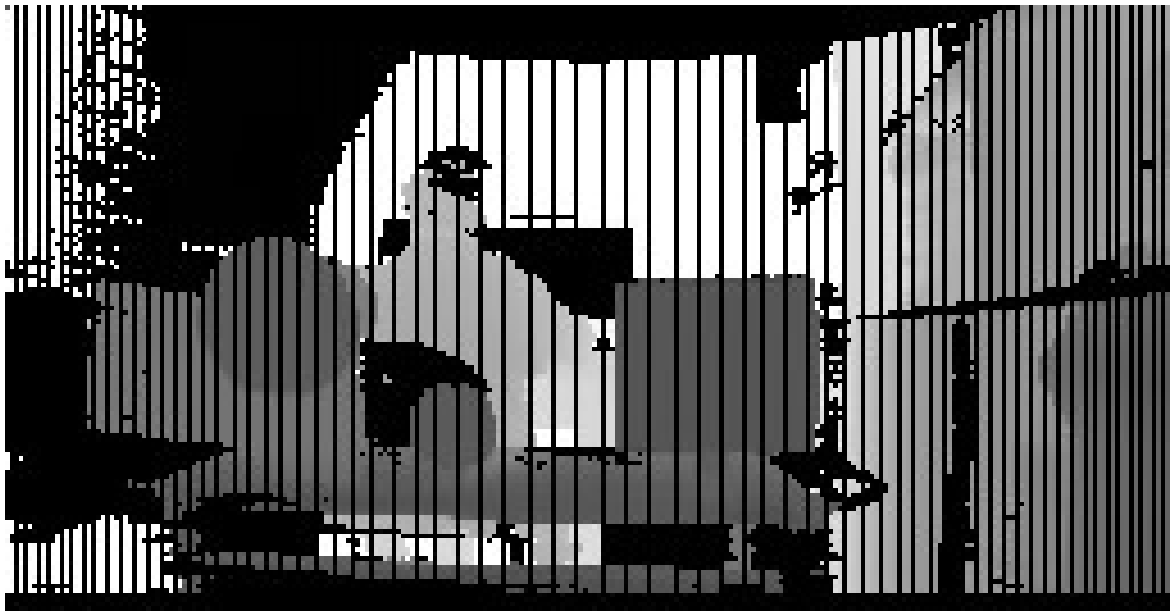


Рисунок 17. Карта глубины при f = 200.

Можно сделать вывод, что изменение фокусного расстояния *f* в проекции по модели камеры с точечной диафрагмой приводит к следующим результатам:

- увеличение параметра *f* приводит к большему масштабированию изображения и увеличению разрешения, однако создает области на плоскости, куда не проецируется ни одна точка;

- уменьшение фокусного расстояния f позволяет уменьшить количество пустых областей и создать более общую неразрывную проекцию облака трехмерных точек, но при этом уменьшается разрешение отдельных деталей.

Параметр f должен быть подобран таким образом, чтобы обеспечить как можно большее разрешение в области интереса, но без разрывов, которые могут существенно ухудшить работы алгоритмов поиска особенностей и приводить к неверным результатам. Для проекции областей, лежащих на значительном отдалении от главной оси, необходимо вносить изменения в матрицу A для того, чтобы избежать деформации изображения при проекции. Если говорить более конкретно, то необходимо повернуть плоскость проекции таким образом, чтобы область интереса лежала приблизительно по центру плоскости.

3.2 Поиск особенностей

Особенности на изображениях можно разделить на две группы: низкоуровневые и высокоуровневые[30, 31]. К низкоуровневым особенностям относятся контуры объектов, углы или связанные области пикселей. Эти особенности сами по себе не представляют интереса, поскольку описывают отношения между пикселями, однако на их основе строятся алгоритмы поиска высокоуровневых объектов. К высокоуровневым объектам можно отнести геометрические фигуры (линии, круги и окружности, прямоугольники, треугольники, многоугольники), шаблонные изображения (задача поиска на изображении заранее заданного шаблона), каркасы объектов [32]. Непосредственно перед процессом поиска особенностей выполняется предварительная обработка изображения, которая усиливает ключевые параметры для алгоритмов поиска, и уменьшает или вовсе удаляет объекты, несущественные для поиска. Предварительная обработка повышает надежность поиска, а так же уменьшает время работы высокоуровневых алгоритмов, поскольку они не обрабатывают удаленные области.

Предварительная обработка и поиск особенностей проводится в среде LabView при помощи пакета Vision Assistant. Этот инструмент предназначен для создания и тестирования приложений обработки изображений, построения комплексных алгоритмов компьютерного зрения.

Блок-схема общего алгоритма поиска представлена на рисунке 16. Механизм построения карты глубины раскрыт в предыдущем разделе.

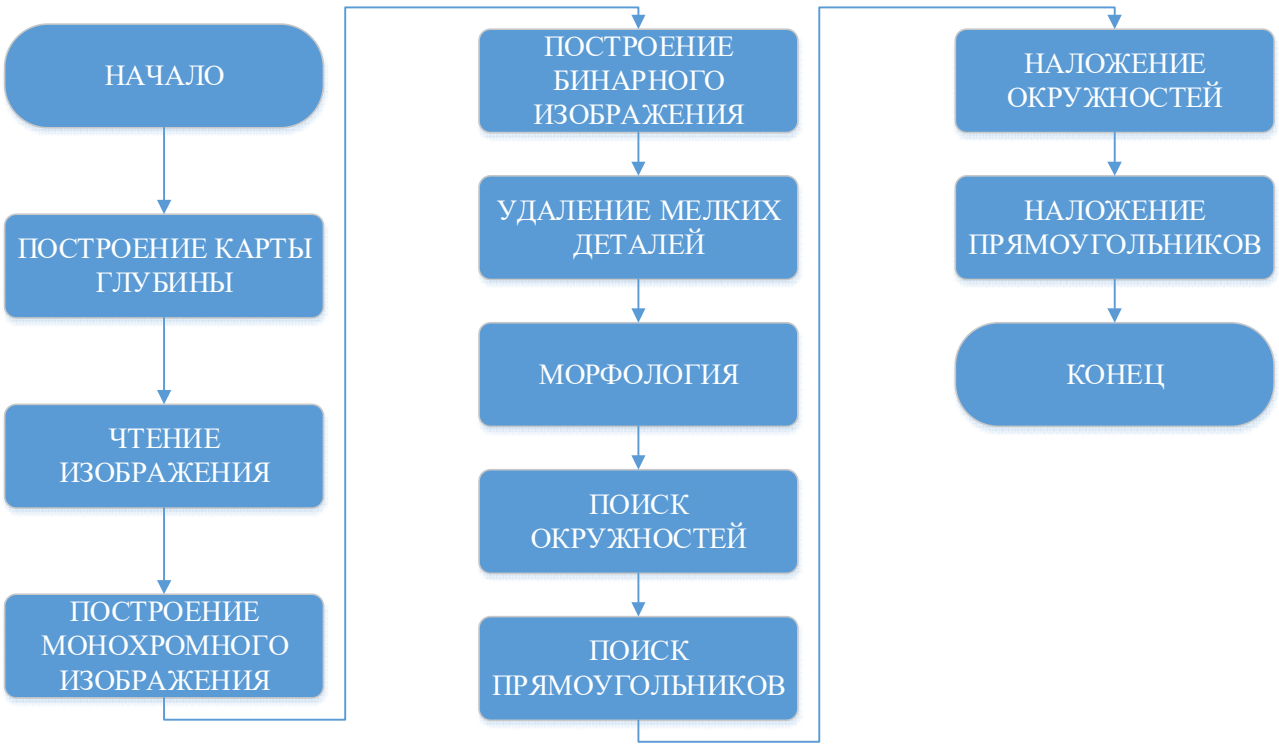


Рисунок 16. Алгоритм поиска геометрических фигур

3.2.1 Предварительная обработка

Первым этапом предварительной обработки является преобразование исходного цветного изображения, закодированного 32 битами в монохромное восьмибитовое. Несмотря на то, что карта глубины фактически являлась восьмибитовой монохромной, изображение было сохранено как цветное, где интенсивности красного, зеленого и синего канала были равны. Дальнейший шаг обработки требует именно восьмибитовое изображение, поэтому такое преобразование необходимо. При этом никакая информация не теряется.

Следующим этапом является перевод монохромного изображения в бинарное. Такое изображение содержит всего две градации серого – черный цвет и белый. Бинарное представление изображения требуется для дальнейшего анализа. Причина использования именно бинарного представления кроется в том, что алгоритмы, работающие всего с двумя градациями серого, хорошо изучены и описаны, также они менее требовательны к вычислительным ресурсам (память и процессорное время). Процесс построения бинарного изображения называется пороговым представлением яркости изображения. Главной целью этого процесса является выделение областей, которые представляют реальные объекты. Такое разделение изображения называется сегментацией. Формально это может быть описано как разделение изображения *F(u, v)* на множество подизображений, называемых регионами $R_1,\ldots,R_k$ таких, что каждое подизображение является потенциальным объектом[33].

Если значения интенсивности пикселей региона, представляющего объект, лежат в определенном интервале, а интенсивность пикселей, являющихся фоном, лежат вне этого интервала, то бинарное изображение может быть получено путем использования алгоритма порогового представления яркости, в котором значения внутри интервала представляются единицей, а значения вне интервала – нулем. Метод порогового представления яркости преобразует монохромное изображение в бинарное таким образом, что регионы, представляющие интерес, отделены от фона. Для этого необходимо, чтобы фон и области, представляющие объекты, имели достаточные различия яркости, а также были известны уровни интенсивности пикселей фона и объектов. Предположим, что бинарное изображение *B(u, v)* то же самое, что и изображение $F_T(u,v)$, полученное методом порогового представления яркости, путем применения границы T на исходное монохромное изображение *F(u, v).* Таким образом $F_T(u,v)= B(u, v),$ где для темного объекта на светлом фоне:

$$F_T(u,v) = \begin{cases} 1, & \text{если } F(u,v) \le T \\ 0, & \text{если } F(u,v) > T \end{cases}$$

Если известно, что интенсивность пикселей, представляющих объект интереса, лежат на отрезке $[T_1, T_2]$, то мы можем записать:

$$F_T(u,v) = \begin{cases} 1, & \text{если } T_1 \le F(u,v) \le T_2 \\ 0, & F(u,v) \notin [T_1, T_2] \end{cases}$$

Важно отметить, что информация о границах для одного набора может оказаться бесполезной для другого изображения и привести к неверной сегментации изображения[33].

Поскольку в данной работе интенсивность пикселей отражает удаленность объектов от лидара, то выбор отрезка $[T_1, T_2]$ позволяет выделять объекты по удаленности. На рисунках 17, 18 и 19 приведен результат сегментации изображения, показанного на рисунке 12, при разных значениях T_1 и T_2. Инструмент Vision Assistant раскрашивает единичные области в красный цвет для большей наглядности. Области в градациях серого будут удалены (заменены черным цветом).

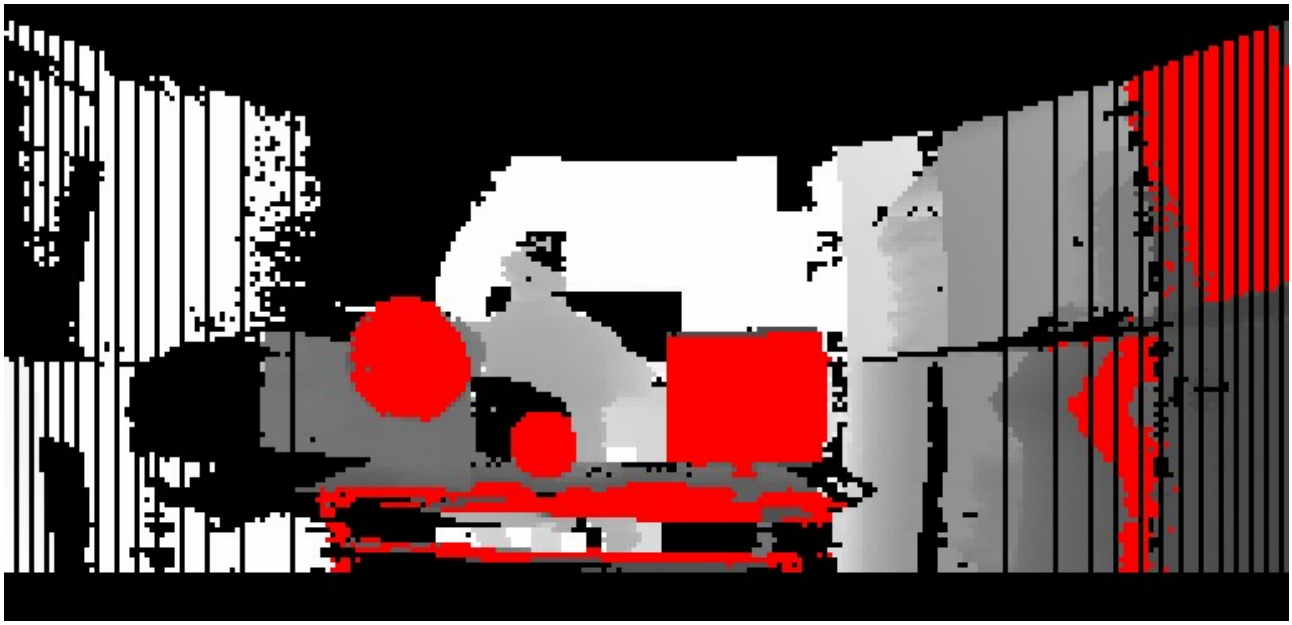


Рисунок 17. Сегментация изображения при T_1 = 83 и T_2 = 93.

Рисунок 18. Сегментация изображения при T_1 = 113 и T_2 = 129

Рисунок 19. Сегментация изображения при T_1 = 66 и T_2 = 87

Выберем для дальнейшей работы параметры T_1 = 83 и T_2 = 93, которые достоверно отображают прямоугольный и два круглых объекта. Итоговый результат приведен на рисунке 20.

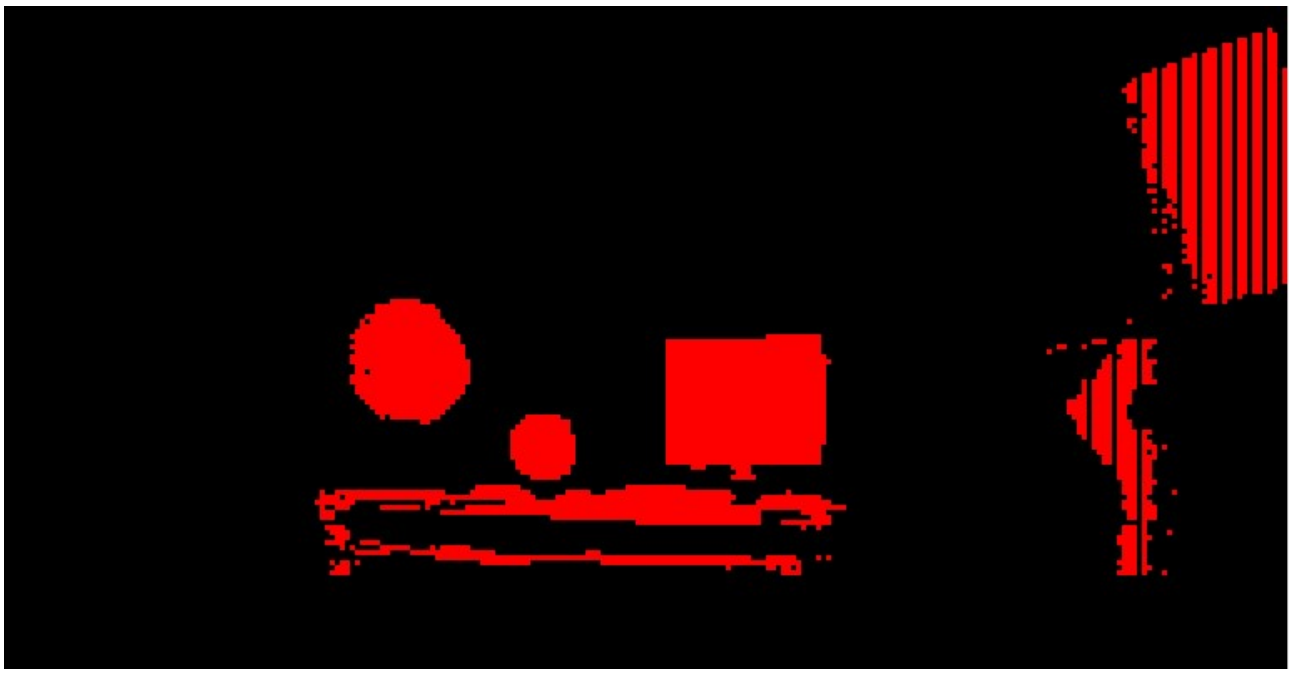


Рисунок 20. Результат порогового представления яркости изображения

Затем происходит удаление слишком мелких деталей, результат показан на рисунке 21.

Рисунок 21. Результат удаления мелких объектов.

Следующим шагом является применение методов математической морфологии к полученному бинарному изображению. Математическая морфология, применительно к компьютерному зрению – это теоретический и

практический подход к анализу и обработке изображений, основанная на теории множеств, топологии и случайных функциях. Методы математической морфологии позволяют преобразовать бинарное изображение для более качественной работы алгоритмов распознавания[34]. В бинарной морфологии в качестве исходных данных принимается изображение *B* и структурный элемент *S*, представляющий собой некоторую бинарную геометрическую форму. Чаще всего используется прямоугольник фиксированного размера или круг. Каждый такой элемент имеет точку, называемую начальной, которая расположена обычно в центре. В начале работы алгоритма морфологии результирующий массив пикселей заполняется 0, затем осуществляется зондирование исходного изображения попиксельно структурным элементом *S*, при котором происходит наложение структурного элемента на зондируемый пиксель. Затем проверяются условия наложения пикселей изображения и структурного элемента в соответствии с некоторым выражением. Если условие выполняется, то результирующее изображение будет иметь 1 в данном месте. Существуют две базовые операции – расширение и сужение, а также набор производных, таких как замыкание и размыкание.

Наращивание бинарного изображения B структурным элементом S обозначается $B \oplus S$ и определяется выражением $B \oplus S = \bigcup_{b \in S} B_s$. В результате применения наращивания исходное изображение увеличивается в размере на величину структурного элемента. Пример наращивания приведен на рисунке 22.

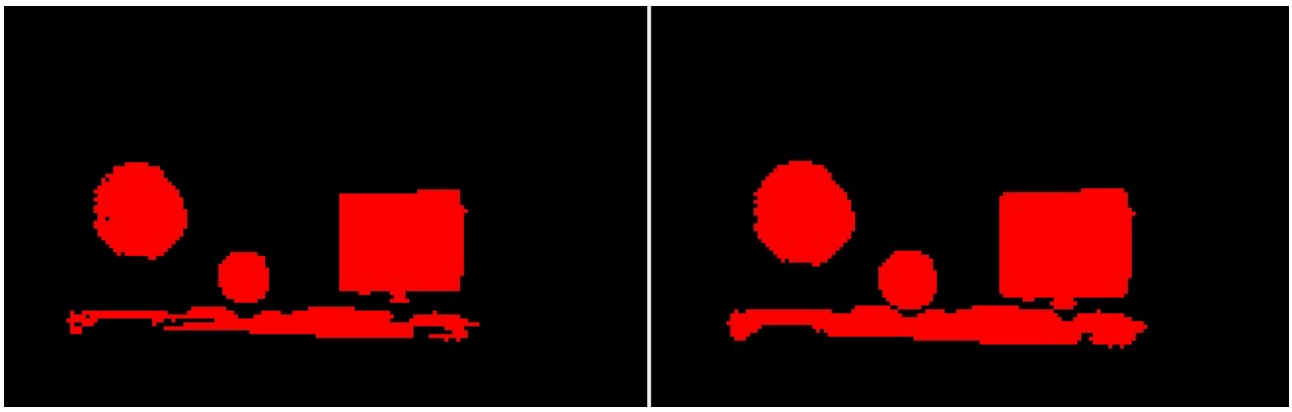


Рисунок 22. Наращивание объектов структурным элементом типа диск размером 3 пикселя. Слева исходное изображение, справа результирующее.

Эрозия изображения обозначается как $B \ominus S$ и определяется формулой $B \ominus S = \{z \in A | S_z \subseteq A\}$. В результате стираются все объекты, которые меньше структурного элемента, объекты, связанные тонкими линиями, разъединяются, а также все остальные объекты уменьшаются в размере. Результат эрозии показан на рисунке 23.

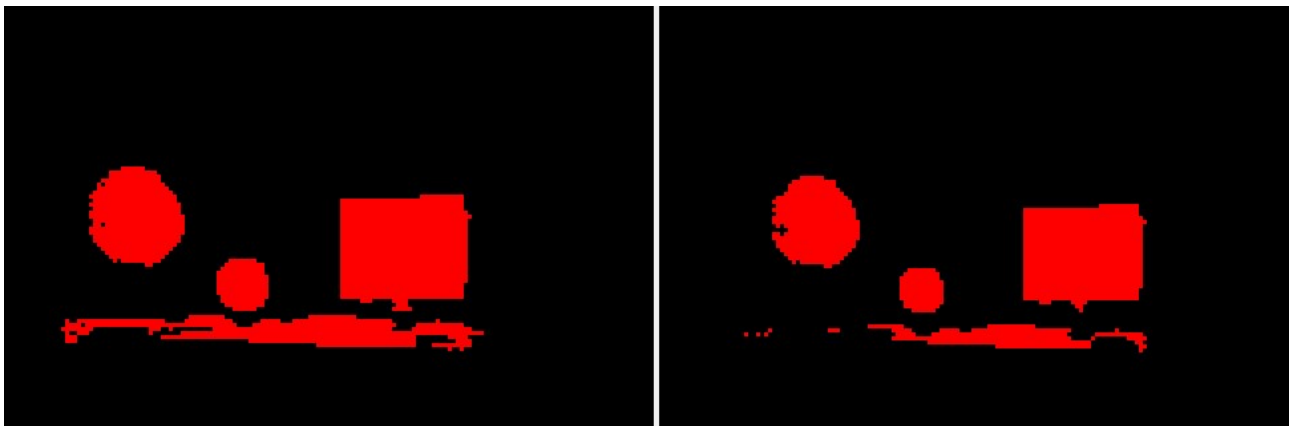


Рисунок 23. Эрозия объектов структурным элементом типа диск размером 3 пикселя. Слева исходное изображение, справа результирующее.

Замыкание задается выражением $B \bullet S = (B \oplus S) \ominus S$ и приводит к заполнению внутренних пустых областей в изображении и убирает небольшие углубления в граничных областях. Замыкание показано на рисунке 24.

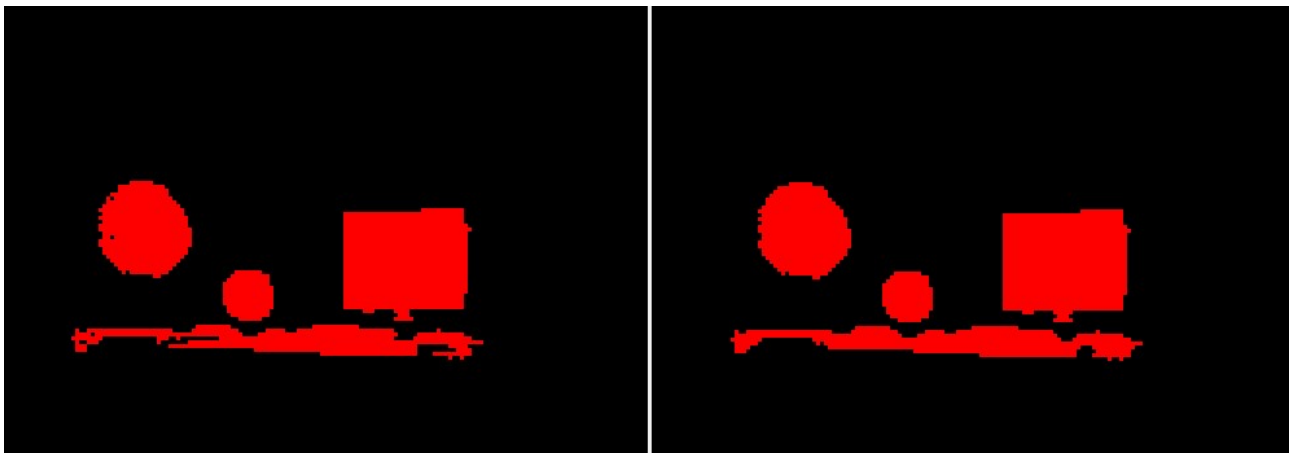


Рисунок 24. Замыкание структурным элементом типа диск размером 3 пикселя. Слева исходное изображение, справа результирующее.

Размыкание задается выражением $B°S = (B \ominus S) \oplus S$ и используется для удаления небольших объектов и шума, однако приводит к уменьшению объектов. Можно нивелировать этот недостаток применением операции наращивания с таким же структурным элементом. Размыкание убирает все объекты, которые меньше структурного элемента, а также сглаживает контуры объектов[35, 36]. В данном случае применятся операция размыкания, результат которой показан на рисунке 25. Используется структурный элемент типа прямоугольник с размером 5 пикселей.

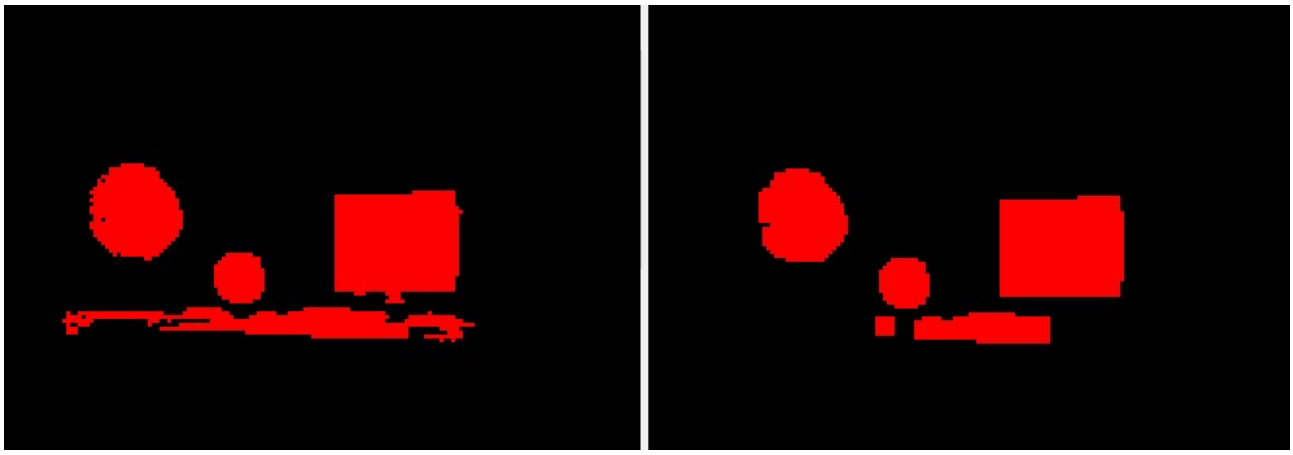


Рисунок 25. Размыкание объектов. Слева исходное изображение, справа результирующее.

Таким образом, в результате предварительной обработки из исходного изображения, мало пригодного для методов распознавания объектов, получилось бинарное изображение, которое высокоуровневые алгоритмы могут успешно обработать.

3.2.2 Высокоуровневый поиск объектов

На данном этапе выполняется поиск геометрических фигур, таких как окружность и прямоугольник. Окружность характеризуется координатами центра и радиусом, а прямоугольник описывается координатами левого верхнего и правого нижнего угла.

Поиск разделяется на два этапа: выделение граней и определение фигур по найденным граням. Гранями называются такие кривые на изображении, вдоль которых происходит резкое изменение интенсивности или других параметров. Поскольку на данном этапе изображение является бинарным, то контуром считается кривая, вдоль которой происходит изменение интенсивности с 0 на 1. Границы отражают ключевые особенности изображения, поэтому преобразование изображения в набор кривых осуществляется для выделения существенных характеристик изображения и для сокращения объема обрабатываемой информации, что ускоряет последующий анализ. Библиотека Vision Assistant предлагает детектор границ Кенни. Несмотря на то, что алгоритм, предложенный Джоном Кенни[37], был разработан в 1986, он до сих пор остается одним из лучших в данной области. Шаги работы алгоритма:

- удаление шума и лишних деталей;
- расчет градиента изображения;
- уменьшение толщины краев;
- связывание краев в контуры.

Детектор использует фильтр на основе первой производной гауссианы при = 1.4:

$$\mathbf{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A}$$

Так как он восприимчив к шуму, то не рекомендуется использовать его на необработанных изображениях. Именно по этой причине в данной работе используется тщательная предварительная подготовка карты глубины.

Краям соответствуют области, где градиент изображения имеет максимальное значение. Так как границы могут находиться в различных направлениях, то алгоритм Кенни использует четыре фильтра для обнаружения горизонтальных, вертикальных и диагональных границ. Воспользовавшись оператором Собеля, который вычисляет приближенное значение градиента интенсивности изображения[38], получаем значение для первой производной в горизонтальном G_v и вертикальном G_u направлении. Из этого градиента можно получить угол направления границы $\Theta=\operatorname{arctg}(\frac{G_v}{G_u})$. Полученный угол округляется до 0^0, 45^0, 90^0 или 135^0. Далее выполняется подавление немаксимумов, в ходе которого проверяется, достигает ли величина градиента локального максимума в заданном направлении. Для сетки 3x3:

- если угол направления градиента равен нулю, точка будет считаться границей, если её интенсивность больше чем у точки, лежащей выше и ниже рассматриваемой точки;
- если угол направления градиента равен 90 градусам, точка будет считаться границей, если её интенсивность больше чем у точек, лежащих слева и справа рассматриваемой точки;
- если угол направления градиента равен 135 градусам, точка будет считаться границей, если её интенсивность больше чем у точек на

левой верхней и правой нижней диагонали относительно рассматриваемой точки;

- если угол направления градиента равен 45 градусам, точка будет считаться границей, если её интенсивность больше чем у точек, лежащих на правой верхней и левой нижней диагонали от рассматриваемой точки.

Далее выполняется уменьшение толщины краев, суть которого заключается в удалении пикселей на границах, которые были вызваны шумом. Для этого необходимо удалить пиксели с меньшим значением градиента и сохранить пиксели с большим значением. Это достигается установкой нижнего и верхнего граничных значений. Подбор таких значений осуществляется индивидуально для каждого изображения или группы похожих изображений. Последним шагом является связывание границ в контуры. На текущий момент имеются пиксели, характеризуемые большим значением градиента, и они однозначно должны быть включены в контуры. Однако вопрос, включать или не включать пиксели с меньшим значением градиента остается открытым, поскольку такие пиксели могут быть шумовыми или в действительности представлять границу реального объекта. Для достижения более точного результата пиксели, представляющие шум, должны быть удалены. Обычно пиксели с небольшим значением градиента, представляющие реальные границы, будут связаны с пикселями, имеющие большое градиентное значение, а шумовые пиксели будут не связаны. Чтобы отследить связанность, для каждого пикселя с малым градиентным значением просматривается восемь соседних пикселей. Если в окрестности существует хотя бы один пиксель с большим градиентным значением, то целевой пиксель должен быть сохранен.

Vision Assistant предлагает незначительную модификацию исходного алгоритма Кенни, предоставляя возможность анализировать не каждый пиксель исходного изображения, а производить обработку с заданным шагом по вертикали и по горизонтали. Результаты выделения границ с различными

параметрами показаны на рисунке 26. Зелеными линиями показаны найденные контуры.

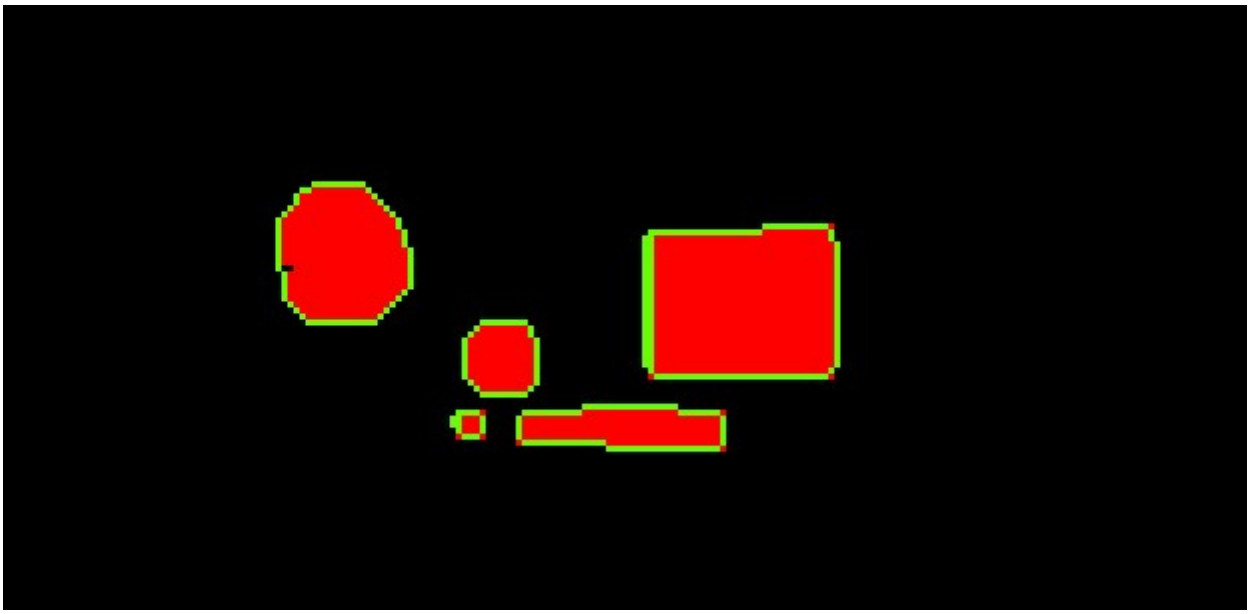


Рисунок 26. Полученные контуры.

Из-за того, что исходное изображение было бинарным, то изменение параметров построения граней не сильно влияют на результат, однако можно заметить, что грани не совсем четко совпадают с границами объектов, несмотря на высокую контрастность и отсутствие помех. При выборе слишком большого размера фильтра объекты начинают сливаться, как показано на рисунке 27. Причиной такого поведения является то, что карта глубины имеет очень низкое разрешение. Например, радиус малого круга не более 6 пикселей, а промежуток между большим прямоугольником и нижней линией не более 3 пикселей. Такие размеры близки к шумовым значениям, что представляет дополнительную сложность при анализе.

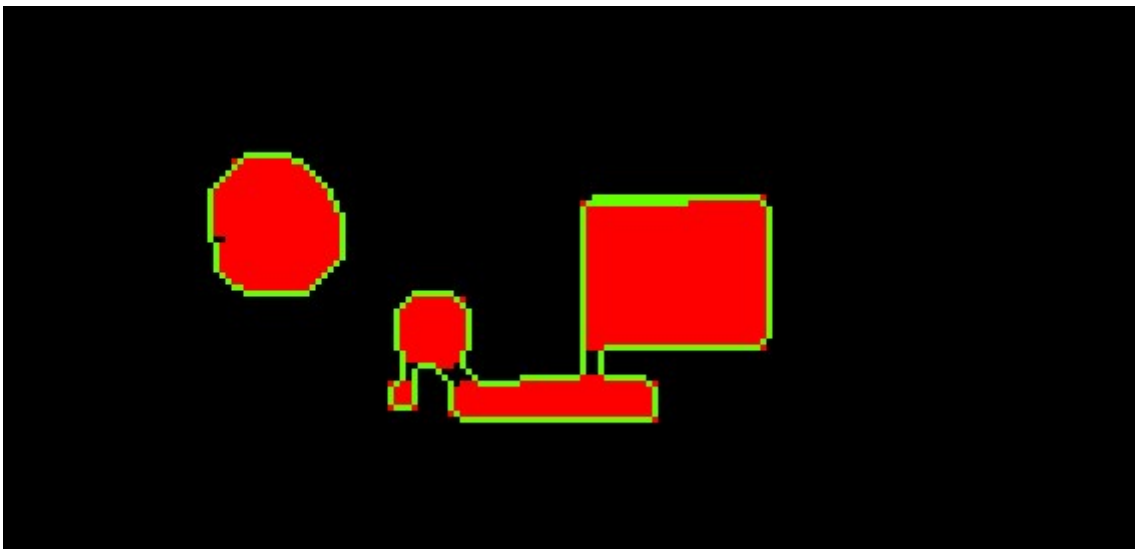


Рисунок 27. Контуры при большом размере фильтра.

Именно из-за низкого разрешения предварительный анализ имеет особенную важность. На рисунке 28 показан результат применения алгоритма поиска контуров на необработанной карте глубины.

Рисунок 27. Контуры на исходном изображении.

Хорошо видно, что контуры объектов искажены, а также выделяются фрагменты, не представляющие интерес, что может замедлить работу более высокоуровневых алгоритмов.

Последним этапом является выделение геометрических фигур на изображении, которые будут наилучшим образом характеризовать объект.

Объекты круглой формы должны быть описаны окружностью с центром, совпадающим с центром круглого объекта, и радиусом, соответствующим радиусу реального объекта. Для прямоугольных объектов соответствующей фигурой будет прямоугольник, углы которого должны совпадать с углами реального прямоугольного объекта. Для определения этих характеристик применяется контурный анализ, который является эффективным методом описания, хранения, распознавания, поиска и сравнения графических объектов[39, 40]. При проведении контурного анализа предполагается, что контур содержит достаточную информацию о форме объекта, а также информация о внутренних пикселях не рассматривается. В связи с этими положениями возникают ограничения применения контурного анализа, которые связаны в основном с выделением контура. Типовыми проблемами являются невозможность выделения контура из-за отсутствия четкой границы между объектами (применительно к картам глубины это означает, что разные объекты расположены вплотную друг к другу на одинаковом или близком расстоянии от лидара), неверное выделение контура из-за перекрытия объектов или их группировки. Однако переход к рассмотрению только контуров объектов позволяет существенно снизить сложность алгоритмов и трудоемкость вычислений. Таким образом, контурный анализ имеет низкую устойчивость к помехам, и частичная видимость объекта или нарушение контура приводит либо к неверному определению, либо к полной его невозможности. Однако простота и быстродействие контурного анализа позволяют успешно его применять. Задача правильного выделения контуров была рассмотрена ранее в текущей главе, сейчас рассмотрим операции над контуром.

Для оперирования контуром необходимо его закодировать. Одним из способов представления является цепной код Фримена[41]. В цепном коде граница представляется в виде последовательности отрезков прямых, характеризуемых длинной и направлением. В основе такого представления лежит 4-связная или 8-связная решетка, которая задает длину отрезка, а

направления задаются выбранным кодом. В 4-связной решетке все направления можно закодировать двумя битами, а в 8-связной – тремя. Пример цепного кодирования приведен на рисунке 28.

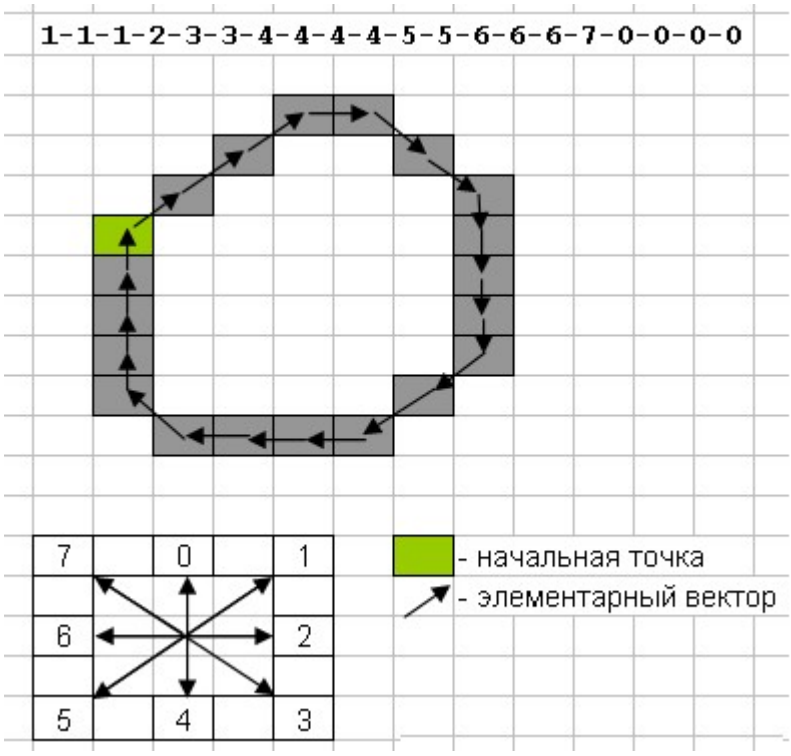


Рисунок 28. Пример цепного кода Фримена с 8-связной решеткой.

Классификация объектов как окружностей или прямоугольников показана в работах [42, 43], оперирующих контурами, представленными в цепном коде Фримена. Эти работы описывают общий подход к классификации контуров, основанный на отношении между длиной контура и площадью, заключенной внутри него. Характеристикой фигуры может служить компактность C, которая определяется следующим выражением: $C = \frac{P^2}{4\pi \cdot S}$, где P – периметр фигуры, S – площадь. Эта формула позволяет вычислять компактность инвариантно относительно размеров фигуры. Круг имеет наименьшее значение компактности среди всех фигур, равное $\pi/4$. Для прямоугольника и квадрата значение C равно примерно 1,25, причем чем больше отношение длин сторон, тем значение больше.

Для оценки качества найденной окружности используется методика, описанная в работе[44]. Если C_{jkl} – окружность, имеющая центр в точке (a_{jkl}, b_{jkl}) и радиус-вектор r_{jkl}, полученный из массива граней для объекта V_i и v_{im} – граничная точка объекта, тогда расстояние между граничными точками и радиус-вектором определяется соотношением:

$$d_{m\to jkl} = \left|\sqrt{(x_{im} - a_{jkl})^2 + (y_{im} - b_{jkl})^2} - r_{jkl}\right|$$

Если расстояние d_m меньше, чем заданное граничное значение T_d $d_{m\to jkl} > T_d$, тогда можно сказать, что точка v_{im} лежит на окружности C_{jkl}. Таким образом, уровень соответствия окружности может быть выражен выражением $\frac{\text{все точки } v_{im}, \text{удовлетворяющие условию } d_{m\to jkl} > T_d}{\text{все точки } v_{im}}$. Инструмент Vision Assistant позволяет проводить такую оценку как для окружностей, так и для прямоугольников.

Финальным шагом является наложение найденных фигур на исходное изображение для наглядной демонстрации результатов работы алгоритмов.

Таким образом, в результате работы алгоритма распознавания, на изображении были выделены два круглых объекта и один прямоугольный. Результат наложения найденных геометрических фигур на исходную карту глубины (рисунок 12) приведен на рисунке 29.

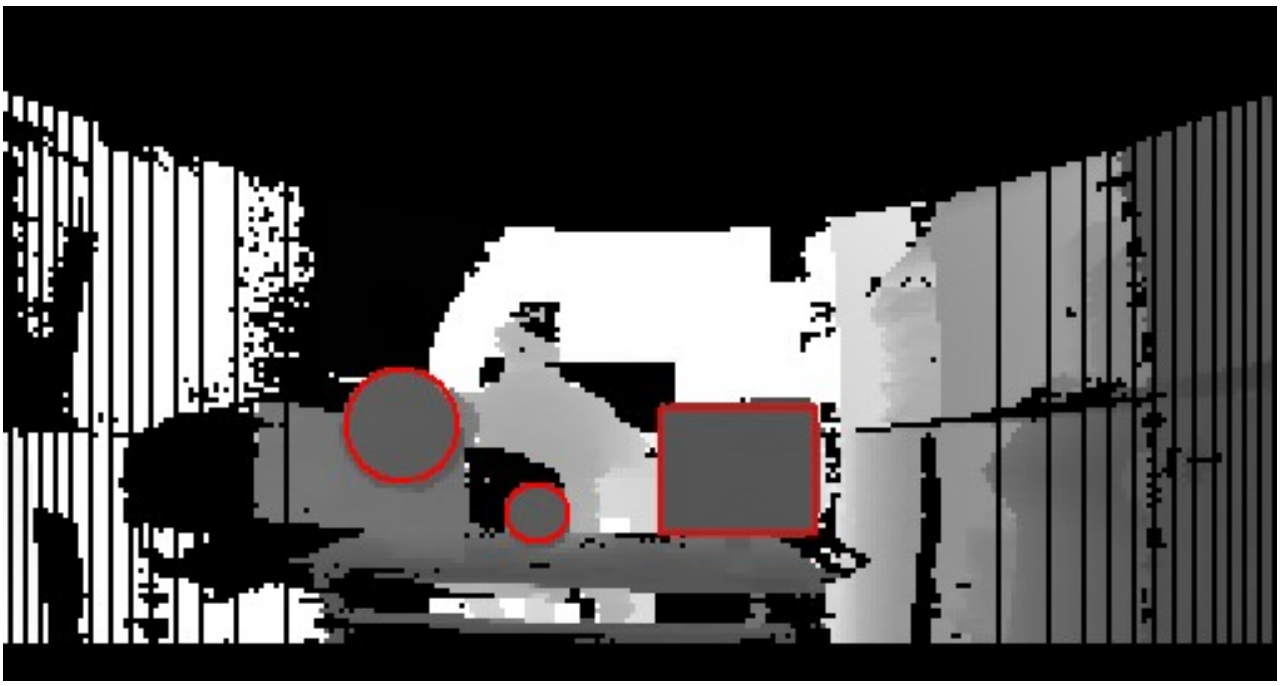


Рисунок 29. Найденные геометрические фигуры.

Численные результаты работы алгоритма приведены в таблицах 4 и 5.

Таблица 4. Характеристики окружностей

Параметр	Большая окружность	Малая окружность
Соответствие, %	95,8	96,9
Радиус, пк	11	6
Координата центра *u*, пк	80	107
Координата центра *v*, пк	85	103

Таблица 5. Характеристика прямоугольника

Параметр	
Соответствие, %	82,6
Ширина, пк	32
Высота, пк	26
Угол поворота, °	3
Координаты угла 1 *u*, пк	132
Координаты угла 1 *v*, пк	82
Координаты угла 2 *u*, пк	163

Координаты угла 2 *v*, пк	80
Координаты угла 3 *u*, пк	163
Координаты угла 3 *v*, пк	107
Координаты угла 4 *u*, пк	131
Координаты угла 4 *v*, пк	106

Рисунок 30 и 31 демонстрирует работу алгоритма на другой карте глубины, показанной на рисунке 15. На рисунке 30 выделен объект заднего плана, на рисунке 31 – переднего.

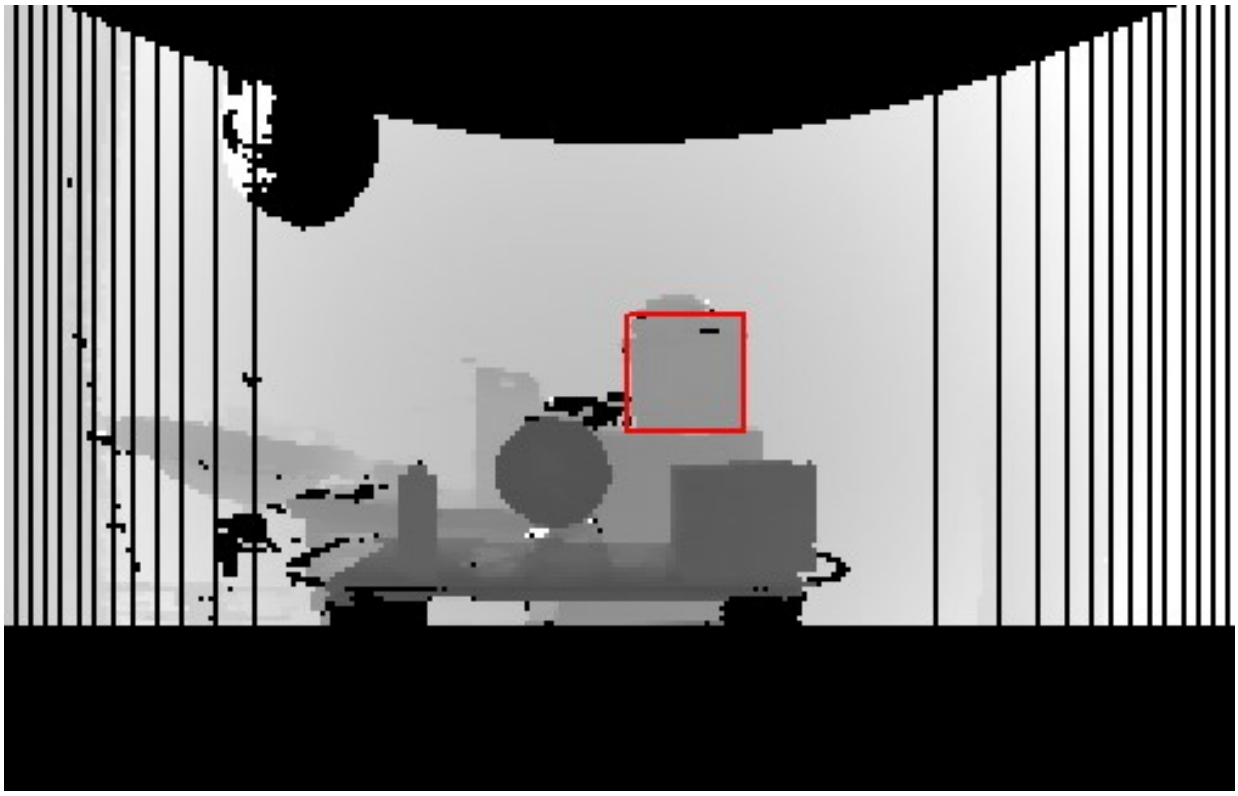


Рисунок 29. Найденная геометрическая фигура заднего плана.

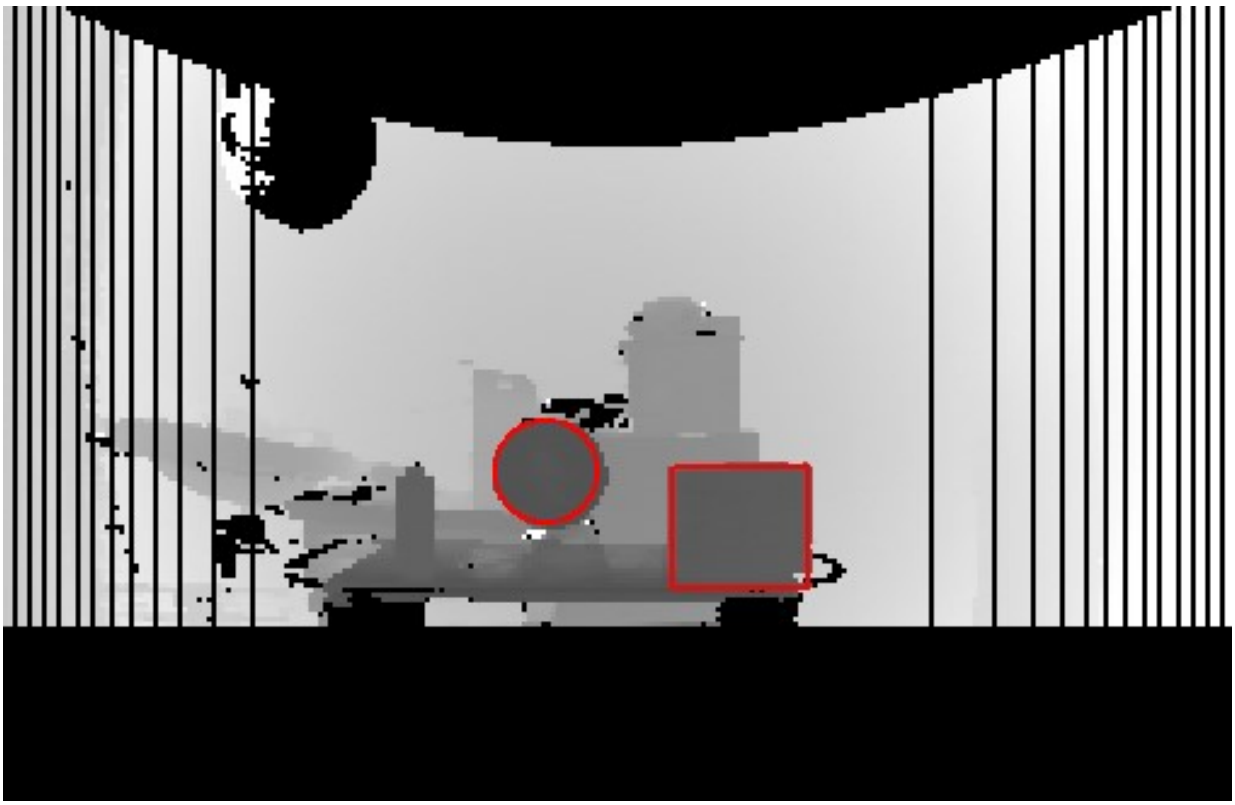


Рисунок 30. Найденные геометрические фигуры переднего плана.

Выводы по главе 3

В главе 3 был описан способ, позволяющий выделять геометрические фигуры на трехмерном облаке точек посредством представления облака точек как карты глубины и применения методов анализа плоских изображений. В результате практического применения описанных в главе методик были успешно распознаны геометрические фигуры, представляющие собой реальные объекты.

ЗАКЛЮЧЕНИЕ

В настоящей работе предложен комплексный подход к выделению и классификации объектов методами обработки двумерных изображений, полученных как проекция трехмерного облака точек на плоскость, при этом построение трехмерной картины выполняется с использованием двумерного лидара.

Решены следующие задачи:

- разработан и реализован на практике способ получения трехмерных данных измерения расстояний до окружающих объектов;
- определены параметры сканирования, при которых достигается оптимальное соотношение между разрешением, быстродействием, точностью и потребляемой памяти;
- разработан и реализован алгоритм представления трехмерных точек как плоского двумерного изображения с целью дальнейшего анализа;
- изучена работа методов обнаружения и классификации объектов на полученных данных.

В результате выполнения данной работы был предложен метод, который позволяет достоверно определять и классифицировать по форме объекты реального мира, причем информацией о таких объектах являются расстояния, полученные последовательным сканированием пространства двумерным лидаром.

Остается ряд открытых вопросов и проблем, которые можно выделить в качестве дальнейших направлений исследования. Одним из таких вопросов является изучение целесообразности повышения разрешения карты глубины искусственным или реальным способом для улучшения надежности распознавания. Другим направлением может служить изучение работы

алгоритмов получения и обработки данных «на лету», то есть без необходимости ожидания все данных сразу.

СПИСОК ИСТОЧНИКОВ

1. *Yogeswaran A., Payeur P.* Features Extraction from Point Clouds for Automated Detection of Deformations on Automotive Body Parts. Robotic and Sensors Environments, 2009.
2. *Jordan C.* Feature Extraction from Depth Maps for Object Recognition. Department of Computer Science, Stanford University, Stanford.
3. *L. Bo, K. Lai, X. Ren, D. Fox* Depth Kernel Descriptors for Object Recognition. IEEE/RSJ International Conference on Intelligent Robots and Systems. San Francisco, 2011.
4. *L. Bo, K. Lai, X. Ren, D. Fox* Object recognition with hierarchical kernel descriptors. IEEE International Conference on Computer Vision and Pattern Recognition, 2011.
5. *Dong-Geol Choi, Yunsu Bok, Jun-Sik Kim, Inwook Shim, In So Kweon* Structure-From-Motion in 3D Space Using 2D Lidars. Sensors (Basel), 2017 Feb; 17(2): 242.
6. *M. Bosse, R. Zlot, P. Flick Zebedee* Design of a spring-mounted 3-D range sensor with application to mobile mapping. IEEE Transactions on Robotics (Volume: 28, Issue: 5, Oct. 2012) p. 1104 - 1119.
7. *M. Bosse, R. Zlot* Continuous 3D scan-matching with a spinning 2D laser. IEEE International Conference on Robotics and Automation, Kobe, Japan, 2009 p. 4244-4251.
8. *A. Harrison, P. Newman* High quality 3D laser ranging under general vehicle motion. IEEE International Conference on Robotics and Automation, 2008.
9. *Teck Chew Ng, Javier Ibanez Guzman, Jin Chang Tan* Development of a 3D LADAR system for autonomous navigation. IEEE International Conference on Robotics and Automation, Singapore, 2004.
10. *James Ring*, The Laser in Astronomy. p. 672–673, New Scientist Jun 20, 1963.

11. *Klaus Tempfli, Norman Kerle, Gerrit C. Huurneman, Lucas L. F. Janssen* Principles of Remote Sensing. The International Institute for Geo-Information Science and Earth Observation, 2009.
12. *М. С. Малашин, Р. П. Каминский, Ю. Б. Борисов* Основы проектирования лазерных локационных систем. Учеб. пособие для радиотехн. спец. вузов. — М.: Высшая школа, 1983. — 207 с.
13. *Z. J. Chong, B. Qin, T. Bandyopadhyay, M. H. Ang Jr., E. Frazzoli, D. Rus* Synthetic 2D LIDAR for Precise Vehicle Localization in 3D Urban Environment. IEEE International Conference on Robotics and Automation (ICRA) Karlsruhe, Germany, May 6-10, 2013.
14. *Gumhold S., Xinlong Wang, MacLeod R.* Feature Extraction from Point Clouds. 10th International Meshing Roundtable Newport Beach, California, U.S.A. October 7-10, 2001.
15. *Weber C., Hahmann S., Hagen H.* Methods for Feature Detection in Point Clouds. Visualization of Large and Unstructured Data Sets - Applications in Geospatial Planning, Modeling and Engineering, 2011, p.90-99.
16. *Weber C., Hahmann S., Hagen H.* Sharp Feature Detection in Point Clouds. Proceedings of the 2010 Shape Modeling International Conference, Washington, DC, USA, 2010, p. 175-186.
17. *Hubeli A., Gross M.* Multiresolution feature extraction for unstructured meshes. Proccedings of IEEE Visualization, 2001, p. 287–294.
18. *Leif Kobbelt, Mario Botsch, Ulrich Schwanecke, Hans-Peter Seidel* Feature sensitive surface extraction from volume data. Proceedings of the 28th annual conference on Computer graphics and interactive techniques , pages 57–66, New York, NY, USA, 2001.
19. *Kouki Watanabe and Alexander G. Belyaev* Detection of salient curvature features on polygonal surfaces. Computer Graphics Forum, pages 385–392, 2001.

20. *Eng Zi Hao and Sutthiphong Srigrarom* Development of 3D Feature Detection and on Board Mapping Algorithm from Video Camera for Navigation. Journal of Applied Science and Engineering, Vol. 19, No. 1, pp. 23-39, 2016.
21. *Saurabh Gupta, Ross Girshick, Pablo Arbel´aez, Jitendra Malik* Learning *Rich* Features from RGB-D Images for Object Detection and Segmentation. European Conference on Computer Vision (ECCV), 2014.
22. Xiao, J., Chen, J., Yeung, D.-Y. and Quan, L., Learning Two-view Stereo Matching. Proceedings of the 10th European Conference on Computer Vision: Part III, ECCV '08, Berlin, Heidelberg, pp. 15-27 (2008).
23. *Michalis Dimitriou, Tsampikos Kounalakis, Nikolaos Vidakis, Georgios Triantafyllidis* Detection and Classification of Multiple Objects using an RGB-D Sensor and Linear Spatial Pyramid Matching. Electronic Letters on Computer Vision and Image Analysis 12(2):78-87; 2013.
24. *J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, A. Blake* Real-Time Human Pose Recognition in Parts from Single Depth Images. 24th IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, USA, June 20-25, 2011.
25. *Siddhartha Chandra, Grigorios G. Chrysos, Iasonas Kokkinos* Surface Based Object Detection in RGBD Images. British Machine Vision Conference, Sep 2016, Swansea, Wales, United Kingdom. 2015.
26. *Paul Mrstik, Kresimir Kusevic* REAL TIME 3D FUSION OF IMAGERY AND MOBILE LIDAR. ASPRS 2009 Annual Conference Baltimore, Maryland March 9-13, 2009.
27. Advantages of LabVIEW in Academic Research http://www.ni.com/white-paper/8534/en/
28. *Pavel Chmelar, Ladislav Beran, Lubos Rejfek* The Depth Map Construction from a 3D Point Cloud. International Conference on Measurement Instrumentation and Electronics, 2016.
29. The Pinhole Camera http://www.ics.uci.edu/~majumder/VC/classes/cameracalib.pdf

30. *Nikita Upadhyaya, Manish Dixit* A Review: Relating Low Level Features to High Level Semantics in CBIR. International Journal of Signal Processing, Image Processing and Pattern Recognition. Vol.9, No.3 (2016), pp.433-444.
31. *Petar S. Aleksic and Aggelos K. Katsaggelos* COMPARISON OF LOW- AND HIGH-LEVEL VISUAL FEATURES FOR AUDIO-VISUAL CONTINUOUS AUTOMATIC SPEECH RECOGNITION. IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004.
32. *Ю. В. Визильтер, С. В. Сидякин* Бициклические каркасы двумерных фигур. ТЕХНИЧЕСКОЕ ЗРЕНИЕ В СИСТЕМАХ УПРАВЛЕНИЯ, МЕХАНИКА 258-264, УПРАВЛЕНИЕ И ИНФОРМАТИКА, МОСКВА 2012г
33. *Ramesh Jain, Rangachar Kasturi, Brian G. Schunck* MACHINE VISION. McGraw-Hill, Inc., 1995
34. *Steven W. Smith* The Scientist and Engineer's Guide to Digital Signal Processing. California Technical Publishing, 1999.
35. *Л.Шапиро, Дж.Стокман*, Компьютерное зрение. изд. — М.: БИНОМ. Лаборатория знаний, 2006. — 752 с
36. *Д.Форсайт, Ж.Понс* Компьютерное зрение. Современный подход. изд. — М.: Вильямс, 2004. — 928 с.
37. *JOHN CANNY* A Computational Approach to Edge Detection. IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. PAMI-8, NO. 6, NOVEMBER 1986.
38. *Duda R., Hart P.* Pattern Classification and Scene Analysis. — John Wiley and Sons, 1973. — P. 271—272.
39. OpenCV шаг за шагом. Нахождение контуров и операции с ними http://robocraft.ru/blog/computervision/640.html
40. *Фурман Я. А., Юрьев А. Н. , Яншин В. В*. Цифровые методы обработки и распознавания бинарных изображений.— Красноярск: Изд-во Краснояр. ун-та, 1992.

41. *H. Freeman* On the encoding of arbitrary geometric configurations. IRE Transactions on Electronic Computers EC- 10(1961) p. 260-268.
42. *Azzam Talal Sleit, Rahmeh Omar Jabay* A Chain Code Approach for Recognizing Basic Shapes. Proceedings of the 4th International Multiconference on Computer Science and Information Technology - CSIT 2006, Volume: 2, p. 298-302.
43. *Ahmad Fashiha Hastawan, Indah Soesanti, Noor Akhmad Setiawan* Effective Method for Circle Detection Based on Transformation of Chain Code Direction. ADVANCES OF SCIENCE AND TECHNOLOGY FOR SOCIETY: Proceedings of the 1st International Conference on Science and Technology 2015.
44. A. Chen, G. Dong Efficient method for rapidly detecting circles based on edge-tracking, in Computational Intelligence and Design, ISCID'09. Second International Symposium on Changsha 1. (IEEE, 2009), p. 402–405
45. *Madhurima Bandyopadhyay, Jan A.N. van Aardt, Kerry Cawse*-Nicholson Classification and Extraction of Trees and Buildings from Urban Scenes Using Discrete Return LiDAR and Aerial Color Imagery. Laser Radar Technology and Applications XVIII, 2013.

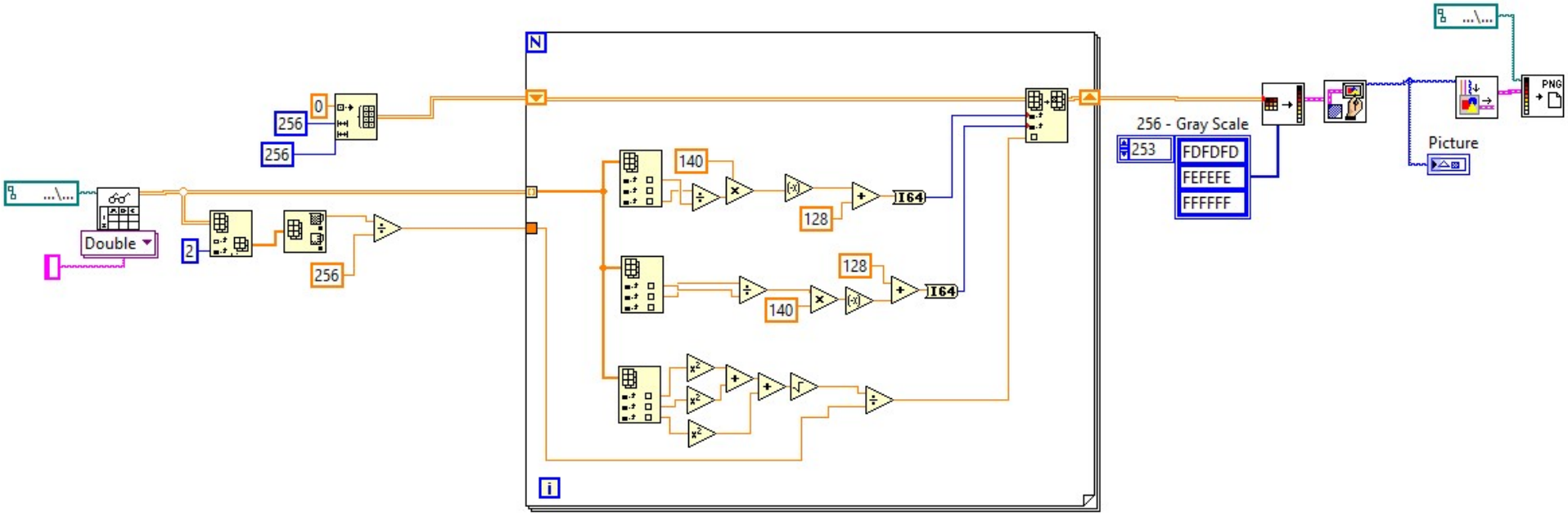
Приложение А. Исходный код программы для МК

```
/*
 * NON-ANGLE CODES DESCRIPTION
 */
 const int MOVE_TO_ZERO_CODE = 241;

 //PIN description
 int _stepP = 2;
 int _stepM = 3;
 int _dirP = 4;
 int _dirM = 5;
 int _button = 8;

 int speed = 0;
 int divider = 4;
 long pulses_per_round = 800*divider;
 long pulse;
 long step;
 int pause;

 int code;

void toZero(){
  digitalWrite(_dirP, LOW);
  while(digitalRead(_button) != 0){
    digitalWrite(_stepP, HIGH);
    delayMicroseconds(pulse);
    digitalWrite(_stepP, LOW);
    delayMicroseconds(pulse);
```

```
  }
  Serial.write(MOVE_TO_ZERO_CODE);
  digitalWrite(_dirP, HIGH);
}

void setup() {
  pinMode(_dirP, OUTPUT);
  pinMode(_dirM, OUTPUT);
  digitalWrite(_dirP, HIGH);
  digitalWrite(_dirM, LOW);
  pinMode(_stepP, OUTPUT);
  pinMode(_stepM, OUTPUT);
  digitalWrite(_stepM, LOW);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  digitalWrite(6, LOW); //motor enabled
  digitalWrite(7, LOW); //motor enabled
  pinMode(_button, INPUT);

  Serial.begin(9600);

  pulse = 25000/divider;
  toZero();
}

void makeStep(int angle){
  long one_step = (360*60)/pulses_per_round;

  long current_iteration_angle = 0;
  for(long i = 0; i < angle; i = i + one_step){
```

```
    digitalWrite(_stepP, HIGH);
    delayMicroseconds(pulse);
    digitalWrite(_stepP, LOW);
    delayMicroseconds(pulse);

    current_iteration_angle +=one_step;
  }
  delay(100);
  Serial.write(current_iteration_angle);
}

void loop() {
  if (Serial.available()) {
    code = Serial.read();
    if(code == MOVE_TO_ZERO_CODE){
      toZero();
    } else{
      makeStep(code);
    }
  }
}
```

Приложение Б. Блочная диаграмма алгоритма построения карты глубины



Приложение В. Блочная диаграмма алгоритма анализа изображения

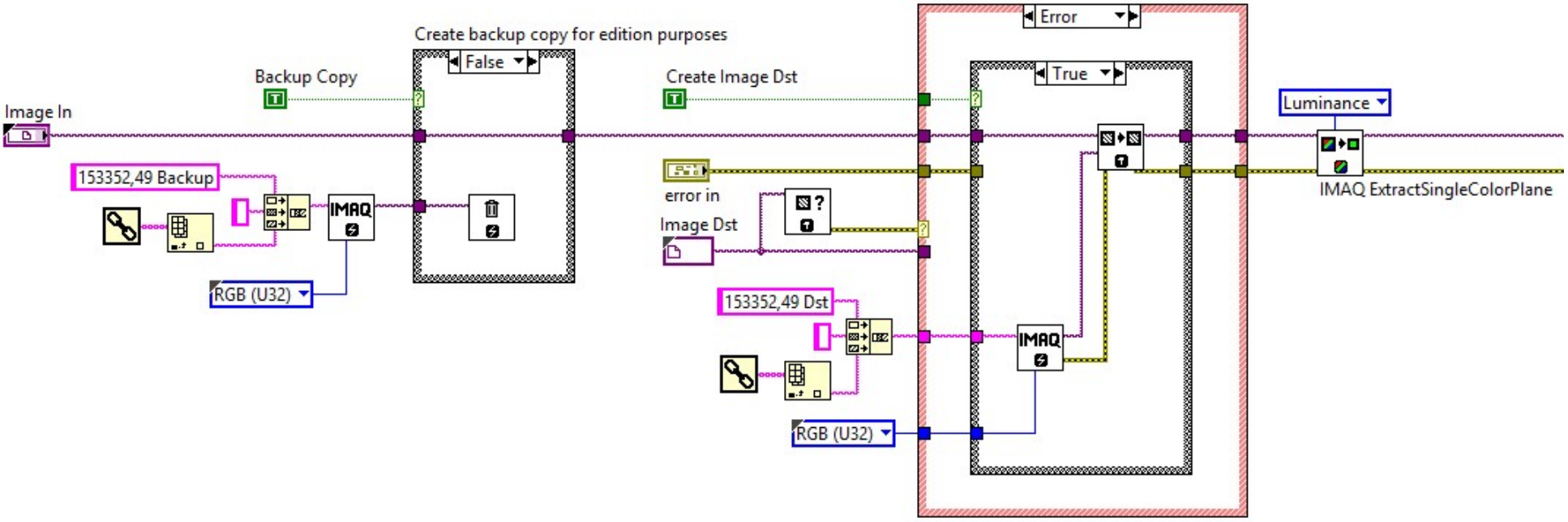


Рисунок В1. Чтение изображения и перевод его в градации серого.

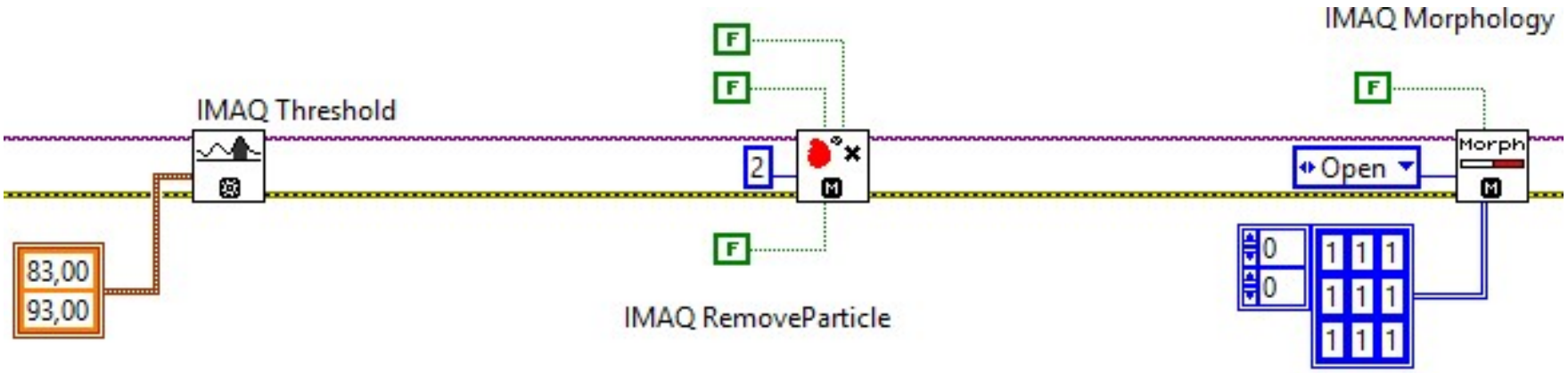


Рисунок В2. Создание бинарного изображения, удаление мелких частиц и выполнение операций морфологии.

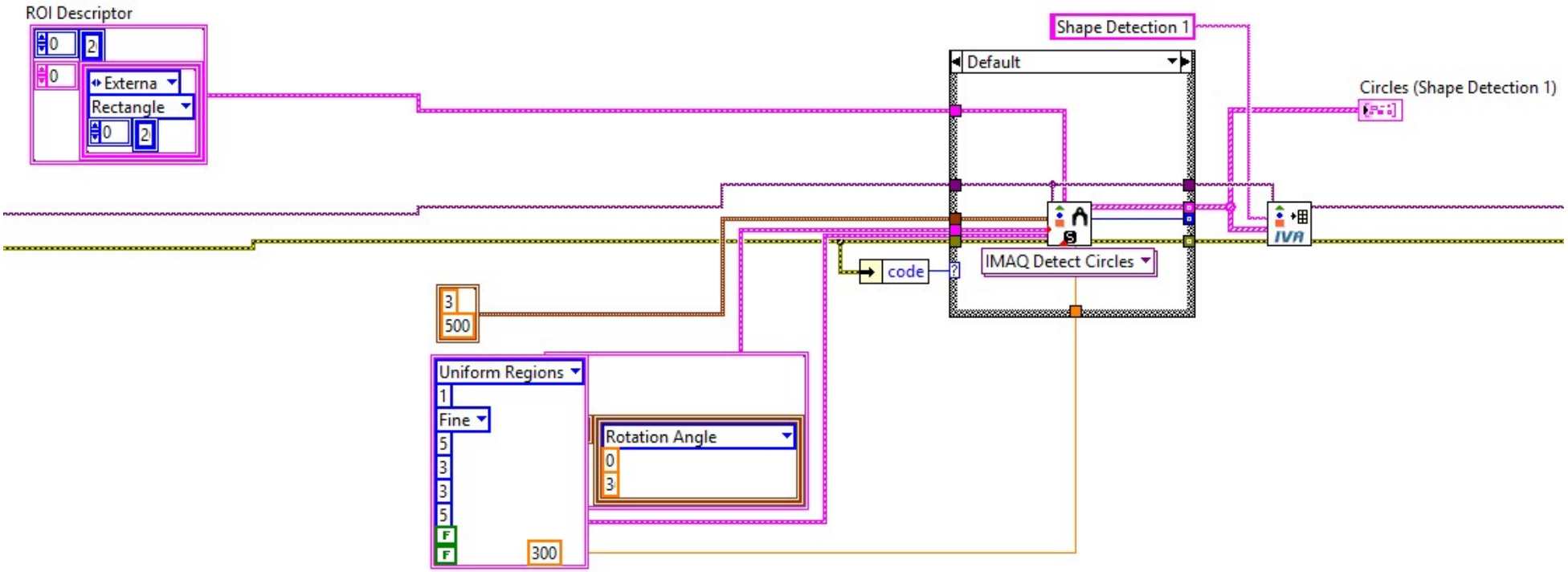


Рисунок В3. Поиск окружностей.

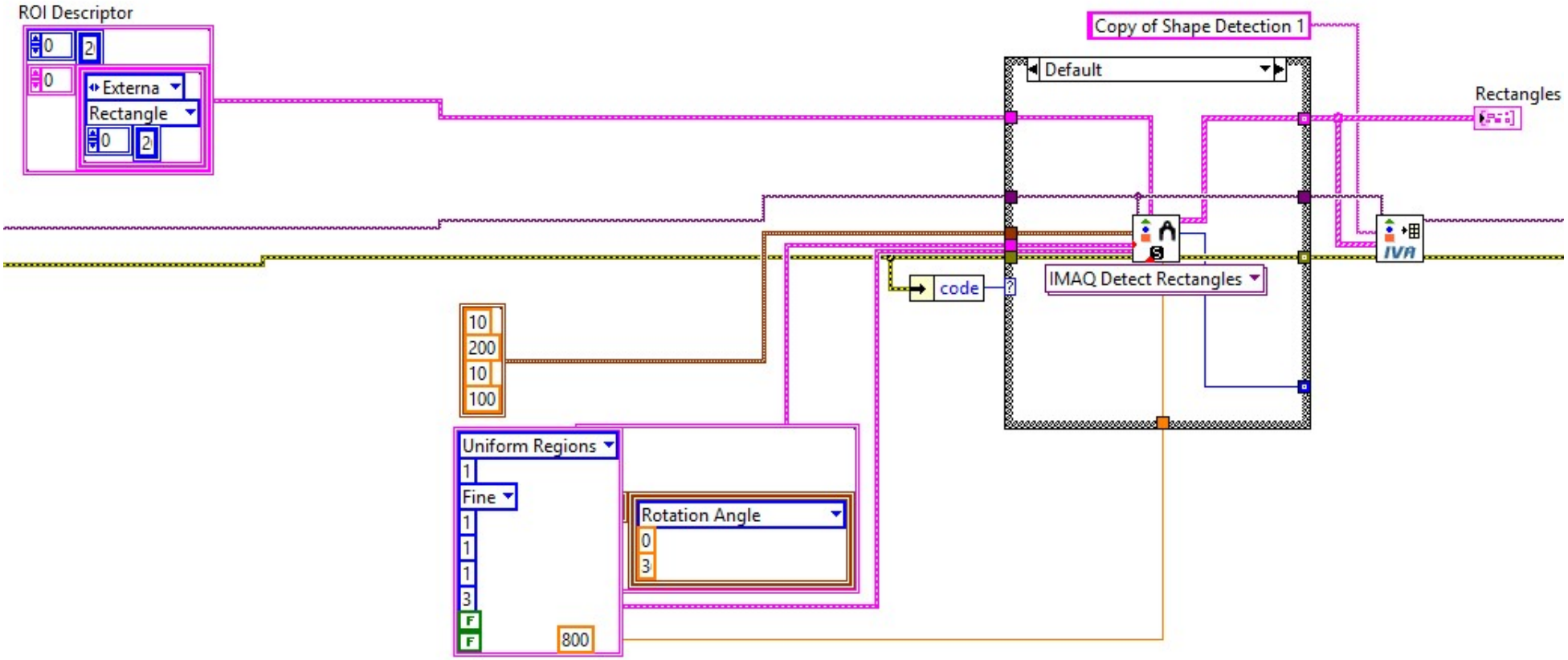


Рисунок В4. Поиск прямоугольников.

Приложение Г. Блочная диаграмма алгоритма наложения найденных фигур на исходное изображение