СОДЕРЖАНИЕ

ВВЕДЕНИЕ

За последние годы был совершён существенный прогресс в доступности и точности сенсоров, позволяющих мобильным роботам осуществлять оценку окружающего пространства. Такие информационно-измерительные устройства как лидары, сонары (и что-нибудь ещё) стали основной опорой алгоритмов для алгоритмов автономной навигации и локализации. Тем не менее в роботах по-прежнему присутствуют оптические системы, так как они дают наиболее читаемую информацию для оператора в случаях, когда его вмешательство необходимо. Существенная часть современных мобильных роботов имеют у себя на борту камеры с широким () или сверхширокими () углами обзора, так как они, хоть и вносят искажения в воспринимаемую картину, позволяют охватить больше окружающего пространства. Набор таких камер может составлять систему кругового обзора [], позволяющую оператору видеть не только в любом направлении, но даже с видом от третьего лица [].

В случае автономных мобильных роботов подобные системы включаются лишь по необходимости, но при этом могут быть весьма дорогостоящими и занимать место в корпусе. Согласно схемам на рисунке **??** широкоугольные камеры в системах кругового обзора роботов часто имеют области пересечения их полей зрения, что позволяет проводить оценку глубины / использовать алгоритмы стереозрения / использовать камеры как стереопару. Это даёт роботу вспомогательный (или единственный) источник трёхмерной информации об окружении без дополнительных расходов.

Однако значительные радиальные искажения изображения, вызванные особенностями используемых объективов, вместе с тем фактом, что области пересечения обычно расположены ближе к краям изображения, не позволяют использовать известные алгоритмы стереозрения.

Целью работы является разработка и изучение точности системы стереозрения, основанной на ортогонально расположенных сверхширокоугольных камерах.

В ходе работы решаются следующие задачи:

– Обзор современных алгоритмов стереозрения / алгоритмов калибровки изображений широкоугольных камеры.
– Обоснование выбора программного обеспечения, используемого для разработки и тестирования алгоритма.
– Разработка алгоритма устранения искажений fisheye-объектива и системы стереозрения на его основе.
– Оценка точности оценки глубины в виртуальной среде.

1 Обзор опыта построения систем стереозрения с использованием сверхширокоугольных камер

1.1 Модель сверхширокоугольной камеры

Сверхширокоугольные объективы имеют в своей основе сложную систему линз, схема которой представлена на рисунке 1.1. Особенности этой системы позволяют достигать существенного угла обзора, но также являются причиной аберрации и характерных искажений изображения. Моделировать реальный ход лучей в подобных камерах нецелесообразно, поэтому исследователи прибегают к моделям камер.

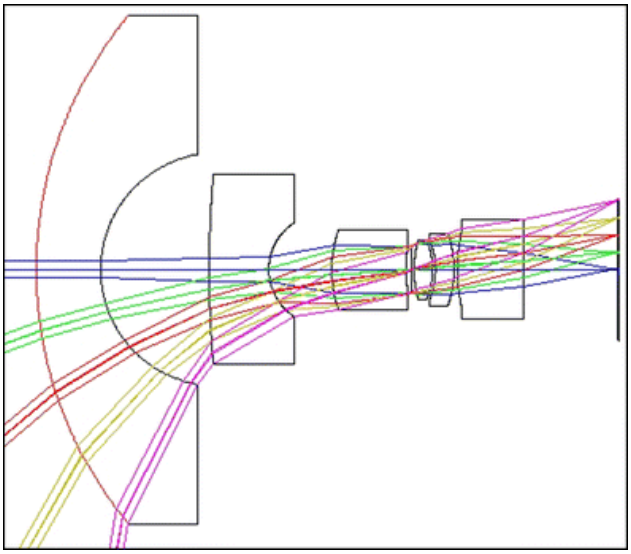

Рисунок 1.1 – Схема хода лучей объектива ”рыбий глаз”

Как видно по рисунку 1.2, модель проекции для камеры это функция, обычно обозначаемая π_c, которая моделирует преобразование из точки трёхмерного пространства ($P = [x_c, y_c, z_c]^T$) в области зрения камеры в точку на плоскости изображения ($p = [x, y]^T$). Единичная полусфера S с центром в точке O_c описывает поле зрения. На ней также лежит точка P_C, являющаяся результатом обратной проекции $\pi_c^{-1}(p)$. Угол θ является углом падения для рассматриваемой точки, а угол ϕ откладывается между положительным направлением оси x и O_ip.

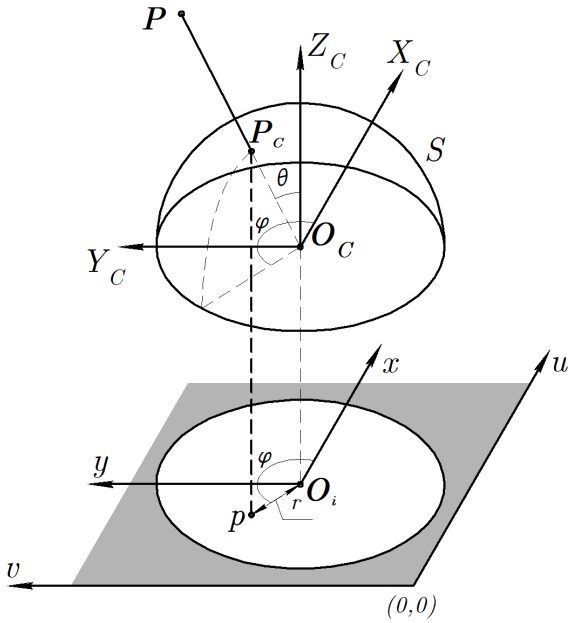

Рисунок 1.2 – Схема проекции точки трёхмерного пространства в точку на изображении

Модели камер включают в себя описания множества типов искажений, но в сверхширокоугольных объективах самыми существенными являются радиальные - искажения, проявляющиеся сильнее ближе к краям изображения. Поэтому далее в этой секции модели будут рассматриваться именно с точки зрения моделирования радиальных искажений.

Перспективная проекция, которая обычно используется в качестве простейшей модели камеры, не способна спроецировать широкоугольный снимок на кадр конечного размера. Поэтому при описании fisheye-объективов опираются на другие виды проекций **??**. Но реальные линзы не всегда в точности следуют заданным моделям, к тому же отличия в используемых параметрах усложняют процесс калибровки камер. По этой причине радиальные искажения принято аппроксимировать многочленами, например, вида

$$\delta r = k_1 r^3 + k_2 r^5 + k_3 r^7 + ... + k_n r^{n+2} \tag{1.1}$$

где k_i - коэффициенты, описывающие внутренние параметры камеры.

В настоящий момент есть несколько распространённых моделей, аппроксимирующих реальные искажения подобных объективов. Модель Канналы и Брандта [3] реализована в OpenCV и описывает радиальные искаже-

ния через угол падения луча света на линзу, а не расстояние от центра изображения до места падения, как это делалось в более ранних моделях. Авторы посчитали, что для описания типичных искажений достаточно пяти членов полинома. Таким образом, указанную модель можно записать следующими уравнениями:

$$\theta = \text{arctan}(\frac{r}{f}), \tag{1.2}$$

$$\delta r = k_1\theta + k_2\theta^3 + k_3\theta^5 + k_4\theta^7 + k_5\theta^9, \tag{1.3}$$

$$\begin{pmatrix} u \\ v \end{pmatrix} = \delta r(\theta) \begin{pmatrix} cos(\phi) \\ sin(\phi) \end{pmatrix}, \tag{1.4}$$

где θ - угол падения луча, определяемый выбранным типом проекции; ϕ - угол между горизонтом и проекцией падающего луча на плоскость изображения; r - расстояние от спроектированной точки до центра изображения; f - фокусное расстояние.

Также большое распространение получила модель Скарамуззы [5], которая легла в основу Matlab Omnidirectional Camera Calibration Toolbox. Она связывает точки на изображении с соответствующей им точкой в координатах камеры следующим образом

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \lambda \begin{pmatrix} u \\ v \\ a_0 + a_2r^2 + a_3r^3 + a_4r^4 \end{pmatrix}, \tag{1.5}$$

где $a_0 ... a_4$ - коэффициенты, описывающие внутренние параметры камеры; λ - масштабный коэффициент.

1.2 Обзор существующих систем стереозрения, использующих сверхширокоугольные камеры

Распространённые библиотеки машинного зрения (OpenCV, MATLAB CV Toolbox) предлагаю готовые к использованию классы и функции, позволяющие после калибровки камер получать с помощью них карты глубины.

Однако на практике эти методы весьма ограничены. Для стереосопоставления используются традиционные методы, приспособленные для классических камер с перспективной проекцией, что не позволяет использовать кадры с широкоугольных камер целиком. В результате у этих кадров после устранения искажений остаётся угол зрения порядка $90°$. Кроме того, библиотечные функции не позволяют задать область интереса для каждой камеры, что ограничивает их область применения только для копланарного расположения камер.

Исследователи предложили несколько реализаций стереозрения, опирающихся на снимки со сверхширокоугольных объективов и лишённых недостатков распространённых решений. Например, метод, предложенный в [1], позволяет создать кольцевую область стереозрения с вертикальным полем зрения $65°$. Для этого используются две $245°$ камеры, закреплённые на противоположных концах жёсткого стержня. Это позволяет достигнуть панорамного обзора глубины с качеством, достаточным для осуществления автономной навигации и локализации [2], но доступна такая схема расположения камер только летательным аппаратам.

Другие авторы [4] решили отказаться от типичных для стереозрения этапов устранения искажений и ректификации и извлекать информацию о глубине напрямую по двум снимкам fisheye-камер. Для производства карт глубины используется свёрточная неиронная сеть, что требует существенных вычислительных мощностей - для достижения производительности в реальном времени разработчикам понадобилось использовать компьютер с ЦПУ i7-4770 и ГПУ NVIDIA GTX 1080Ti. В мобильном автономном роботе такой вычислитель разместить может быть проблематично. Кроме того, метод так же предполагает, что обе камеры направлены в одном направлении.

1.3 Обоснование выбора ПО

Разработку и первоначальные испытания алгоритма стереозрения целесообразно проводить в виртуальной среде. Это позволяет значительно упростить разработку, так как уменьшает время на проверку гипотез и рас-

ходы на реальное оборудование, особенно в случае неудачных испытаний. Из-за этих факторов виртуальное моделирование в робототехнике приобрело широкое распространение и активно применяется, например, для разработки систем локализации и навигации беспилотного транспорта []. Возросшее качество компьютерной графики к тому же позволило моделировать реалистичное окружение, что особенно важно при работе с системами технического зрения.

Для разработки алгоритма, описанного в этой работе, нужна виртуальная среда, в которой можно симулировать несколько широкоугольных камер и настраивать их параметры, легко интегрировать алгоритмы технического зрения и создать окружение, приближенное к тому, в котором будет работать алгоритм. На данный момент исследователю доступен широкий выбор программного обеспечения, подходящего для этой задачи. В таблице 1.1 представлено сравнение имеющихся предложений по основным изложенным выше требованиям.

Таблица 1.1 – Сравнение ПО для симуляции

Название	**Симуляция fisheye-камер**	**Реалистичное моделирование**	**Интеграция кода**	**Доступ**
Gazebo	Возможна	Затруднено	Возможна посредством ROS	Бесплат
RoboDK	Нет	Затруднено	Нет	От 145€
Webots	Затруднена	Возможно	Возможна	Бесплат
CoppeliaSim	Затруднена	Затруднено	Возможна	Бесплат
NVIDIA Isaac Sim	Возможна	Возможно	Возможна	Бесплат
CARLA	Затруднена	Возможно	Возможна	Бесплат
Unity	Возможна	Возможно	Возможна	Бесплат

По результатам оценки собранные сведений было принято решение проводить разработку в симуляторе Unity. Он позволяет подробно настра-

ивать камеру и эмулировать fisheye-объектив, строить реалистичные сцены благодаря свободному импорту моделей, а при программировании в симуляторе можно использовать сторонние программы в виде динамически подключаемых библиотек. По функционалу так же подходит NVIDIA Isaac Sim, но от него пришлось отказаться из-за высоких системных требований и малой изученности продукта.

Разрабатываемое решение должно иметь возможность внедрения в ПО робота, поэтому должно реализовываться на одном из популярных и быстродейственных языков программирования. Учитывая необходимость интеграции с Unity и потребность использовать популярные библиотеки , был выбран язык C++. Другим важным фактором является библиотека обработки изображений. В качестве основы для программной части была выбрана библиотека OpenCV, являющаяся стандартом при разработке систем технического зрения. Она доступна к использованию со множеством языков программирования, но наилучшую производительность показывает именно с C++ [].

1.4 Выводы по главе

Обзор современных моделей сверхширокоугольных камер позволил выбрать наиболее точную и удобную для калибровки. Был осуществлён обзор существующих систем стереозрения, применяющих камеры типа ”рыбий глаз” с целью ознакомления с мировым опытом. Было принято решение разрабатывать и тестировать предлагаемую систему стереозрения с применением виртуального моделирования.

Для моделирования выбрана среда разработки Unity. Для обработки изображений с камер выбрана библиотека OpenCV для языка программирования C++.

2 Система стереозрения

2.1 Алгоритм устранения искажений

Как уже упоминалось в секции 1.1, существующие способы устранения радиальных искажений не позволяют работать близко к краям изображений, поэтому для реализации предлагаемой системы стереозрения был разработан алгоритм устранения искажений на основе модели [5]. Схема геометрического принципа, лежащего в основе этого алгоритма, представлена на рисунке 2.1.

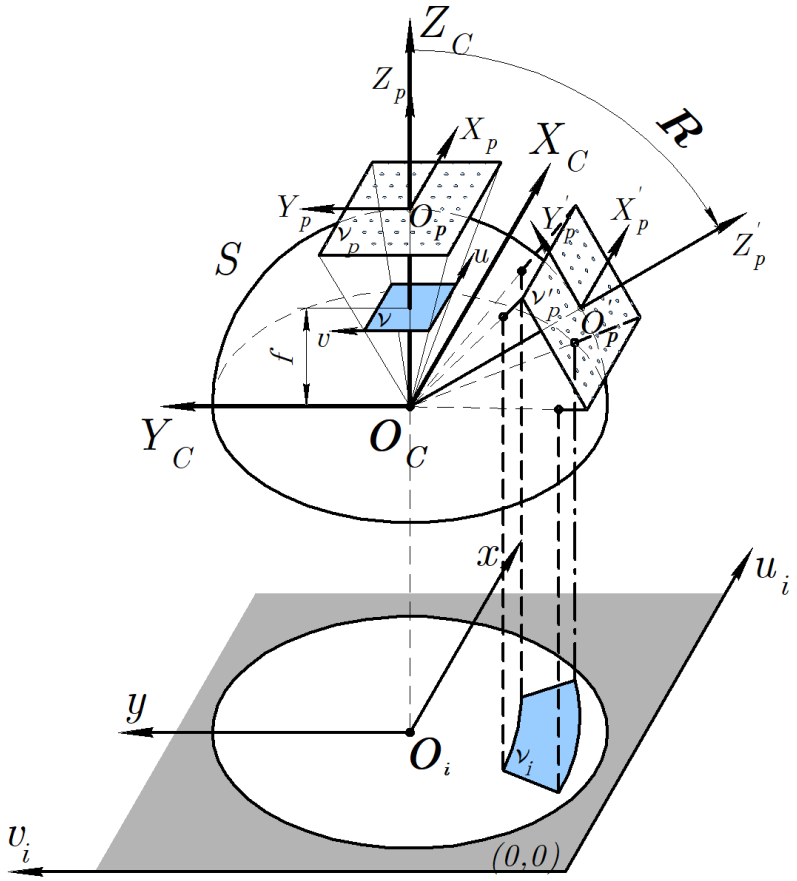

Рисунок 2.1 – Схема принципа устранения искажений

Цель алгоритма - найти, куда на выходном изображении проектируются все пиксели из выбранного участка входного сверхширокоугольного изображения. Сделать это можно, выполнив обратное преобразование 1.5 для каждого пикселя fisheye-снимка и затем прямое преобразование модели камеры-обскуры для получения результирующей проекции. Однако такой подход приведёт к возникновению дефектов из-за несовпадения частот дис-

кретизации двух изображений. Поэтому вместо этого алгоритм выполняет обратное преобразование для каждого пикселя итогового изображения ν, находя таким образом соответствующую ему точку в системе координат камеры (X_c, Y_c, Z_c)

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} (u + c_x) * z_c / f \\ (v + c_y) * z_c / f \\ z_c \end{bmatrix}, \tag{2.1}$$

где c_x, c_y - координаты центра изображения; f - фокусное расстояние.

Набор таких точек формирует прямоугольную область ν_p с центром в точке O_p и является виртуальной камерой-обскурой с оптической осью Z_p. Поворот точек, входящих в ν_p, с помощью матрицы вращения $\boldsymbol{R}$ образует ν'_p и позволяет таким образом задать направление обзора и ориентацию виртуальной камеры.

$$\begin{bmatrix} x'_p \\ y'_p \\ z'_p \end{bmatrix} = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} \boldsymbol{R}. \tag{2.2}$$

$$\boldsymbol{R} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & \sin\beta \\ 0 & -\sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\gamma & 0 & -\sin\gamma \\ 0 & 1 & 0 \\ \sin\gamma & 0 & \cos\gamma \end{bmatrix}, \tag{2.3}$$

где α, β, γ - углы Эйлера.

Тогда обратная fisheye-проекция точек из ν'_p позволяют получить область ν_i исходного изображения с искомыми пикселями. Таким образом, зная как геометрически проектируется каждая точка из ν в ν_i, можно перенести информацию о цвете и получить изображение с устранёнными радиальными искажениями в любой части поля зрения камеры.

Так как в используемой модели искажения в точке считаются центрально симметричными и зависят только от её удаления от центра изображения, рассмотрим ход падающего луча в координатах (Z_c, ρ), изображённый на рисунке 2.2.

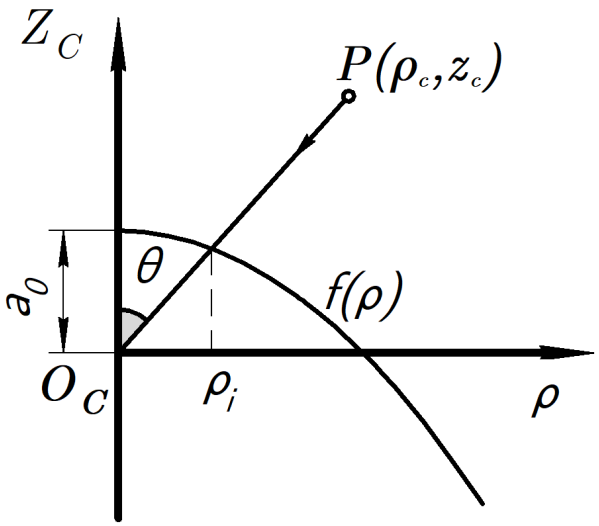

Рисунок 2.2

Для модели (1.5) обратная проекция записывается как

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} \frac{x_c}{\lambda} \\ \frac{y_c}{\lambda} \end{bmatrix}, \tag{2.4}$$

где $\lambda = \rho_c / \rho_i$ - масштабный коэффициент.

Для нахождения ρ_i был применён метод последовательных приближений. Блок-схема алгоритма приведена на рисунке 2.3.

Рисунок 2.3 – Блок-схема алгоритма поиска ρ

Очевидно, весь процесс преобразования $\nu \rightarrow \nu_i$ требует выполнения существенного количества математических операций, что негативно сказывается на скорости, с которой алгоритм может обрабатывать изображения в реальном времени. Однако при неизменных параметрах модели алгоритм достаточно выполнить лишь один раз, записав результат в таблицу поиска - структуру данных, которая позволяет дальнейшие преобразования проводить по уже известным соотношениям между пикселями. Время построения таблицы поиска для выходного изображения разрешением 540*540 пикселей

и входного изображения 1080*1080 пикселей составляет $\sim 2.2c$. Это время можно уменьшить, применив различные оптимизации и параллельные вычисления (проекция каждого пикселя может рассчитывать независимо). Время отображения с использованием таблиц поиска составляет для того же изображения ..., что позволяет применять алгоритм для устранения искажений в реальном времени.

2.2 Описание системы стереозрения

Предлагаемая система стереозрения состоит из двух или более камер с объективами типа "рыбий глаз" $\geqslant 180^\circ$, расположенных ортогонально. В таком случае поля зрения соседних камер могут пересекаться, образуя области, точки пространства в которых видны на обеих камерах. Это позволяет получить информацию о глубине в указанных областях.

Пример такой системы для робота "Капитан" представлен на рисунке 2.4. Здесь 4 камеры, изображённых полукругами, с центрами в точках C_{0-3} размещены спереди, сзади и по бортам корпуса робота. Пересечения их полей зрения образуют 4 области объёмного зрения, закрашенных голубым.

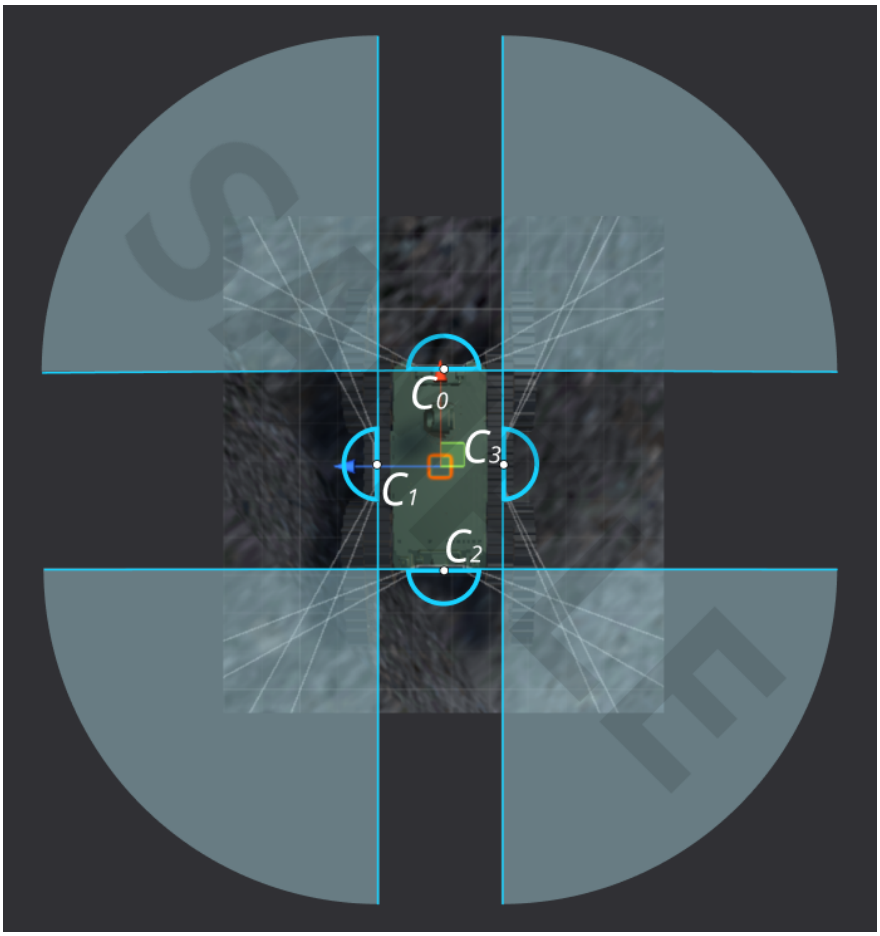

Рисунок 2.4 – Система стереозрения из четырёх камер на примере робота ”Капитан”

Применение традиционных алгоритмов стереозрения предполагает наличие стереопары - двух камер с известным взаимным положением, наблюдающих одну область пространства с разных ракурсов. Описанный в предыдущей секции метод устранения искажений сверхширокоугольной камеры позволяет создавать виртуальные камеры-обскуры и регулировать их направление обзора. Таким образом, стереопару можно сформировать из двух таких виртуальных камер. Далее для упрощения рассмотрения системы будет считаться, что оптические оси всех камер находятся в одной плоскости, а на ориентацию виртуальных камер влияет только один угол.

На рисунке 2.5 изображён вариант системы с двумя камерами под углом $90°$, соответствующий фрагменту схемы, представленной на рисунке 2.4.

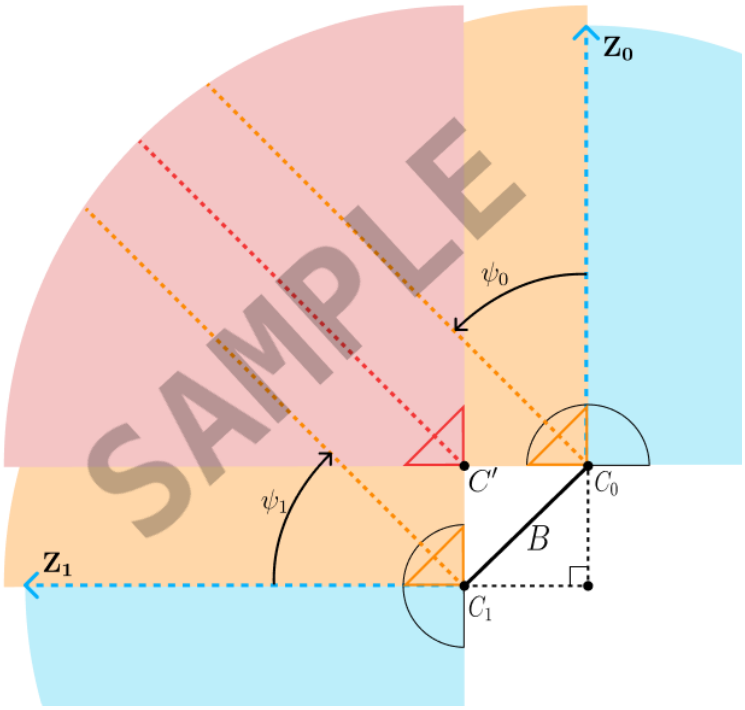

Рисунок 2.5 – Геометрическая модель бинокулярной системы стереозрения

Здесь у полей зрения камер C_0 и C_1 есть область пересечения, обозначенная красным. Эта область эквивалентна области пересечения полей зрения двух камер с полями зрения $90°$ (обозначены оранжевым), повёрнутых на $45°$ в сторону области интереса. Примеры изображений, полученных в такой конфигурации, изображены на рисунке **??**.

Рисунок 2.6 – Пример исходных изображений и снимков виртуальной стереопары

2.3 Виртуальное моделирование системы

Описанная система была смоделирована в среде Unity, её внешний вид представлен на рисунке 2.7. Мир Unity предназначен для базовой проверки работоспособности испытываемого принципа, поэтому не содержит подробной модели какого-либо робота. В нём присутствуют: плоскость земли, кронштейн, на котором сверхширокоугольные камеры закреплены под углом

90 deg, и различные объекты-цели, предназначенные для оценки расстояния. В силу особенностей работы многих алгоритмов стереозрения эти объекты должны сильнотекстурированы [].

Рисунок 2.7 – Внешний вид модели системы стереозрения

В Unity создание сцены происходит с использованием встроенных примитивов, импортированных файлов моделей популярных форматов или моделей из магазина ассетов, предлагающего обширную библиотеку объектов и текстур, повторяющих различные реальные объекты. В данной работе использовались как и примитивы для создания простых объектов, так и модели из магазина для имитации препятствий и окружения.

Для моделирования камеры ”рыбий глаз” использовался аддон Dome Tools из магазина Unity Asset Store. Он позволяет моделировать сверхширокоугольные объективы с разным углом зрения в эквидистантной проекции []. При этом с точки зрения других искажений, не относящихся к моделированию правильной проекции, снимки с этой камеры получаются идеальными. Окно настроек камеры представлено на рисунке 2.8.

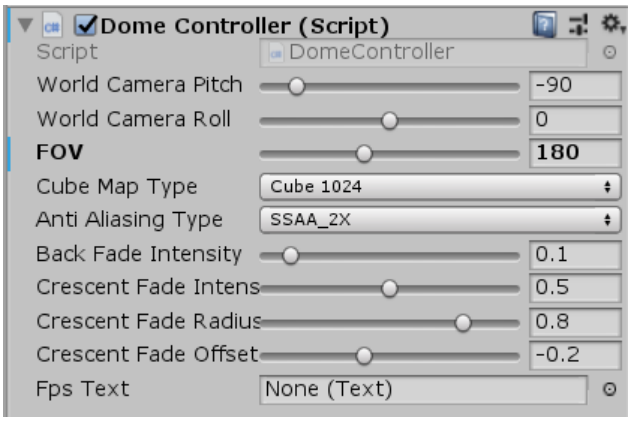

Рисунок 2.8 – Окно настроек виртуальной камеры в Unity

Для виртуальной камеры доступны настройки угла зрения и виньетки по краям изображения, а также различных параметров рендеринга, влияющих на качество получаемого изображения.

2.4 Передача изображений с виртуальных камер для обработки

Unity в качестве основного языка программирования использует C# и не позволяет импортировать выбранную библиотеку технического зрения OpenCV напрямую. Однако как уже упоминалось в секции 1.3, данная среда позволяет интегрировать плагины, в том числе написанные на других языках программирования, в виде динамически подключаемых библиотек (DLL). Именно этот подход был использован для передачи изображений - была разработана библиотека с функциями для обработки снимков, которая компилировалась отдельно и затем импортировалась в Unity.

Для реализации описанного в этой работе принципа и упрощения разработки в библиотеке были реализованы следующие функции:

- initialize - выделяет память под нужные структуры, заполняет таблицу поиска и создаёт окна для отображения будущих изображений.
- getImages - передаёт изображения в библиотеку и производит обработку изображений в соответствии с аргументами.
- takeScreenshot - производит ту же обработку входного изображения, что и предыдущая функция, но результат сохраняет в файл.
- processImage - передаёт обратно в скрипт уже обработанные изображения для удобного отображения в интерфейсе Unity.
- terminate - высвобождает память по завершении работы симуляции.

Обработка, упомянутая в пунктах с функциями getImages и takeScreenshot, заключается в конвертации цветового пространства изображений (из RGBA, используемого в Unity, в BGR, используемый в OpenCV), горизонтальном зеркальном отображении (из-за разного положения точки отсчёта координат пикселей) и, наконец, устранение искажений в области интереса. Так как для наилучшей работы стереосопоставления снимки должны быть синхронизированы (то есть получены камерами в один момент времени), обработка производится сразу для двух изображений. Программный код файлов, входящих в динамически подключаемую библиотеку, представлен в приложении **??**.

Также написан специальный скрипт, который управляет процессом взаимодействия симуляции и библиотеки, его код приведён в приложении **??**. Эта программа имеет окно настроек, которое позволяет задать используемые камеры и элементы управления параметрами, выбрать область интереса и …Внешний вид этого окна представлен на рисунке 2.9.

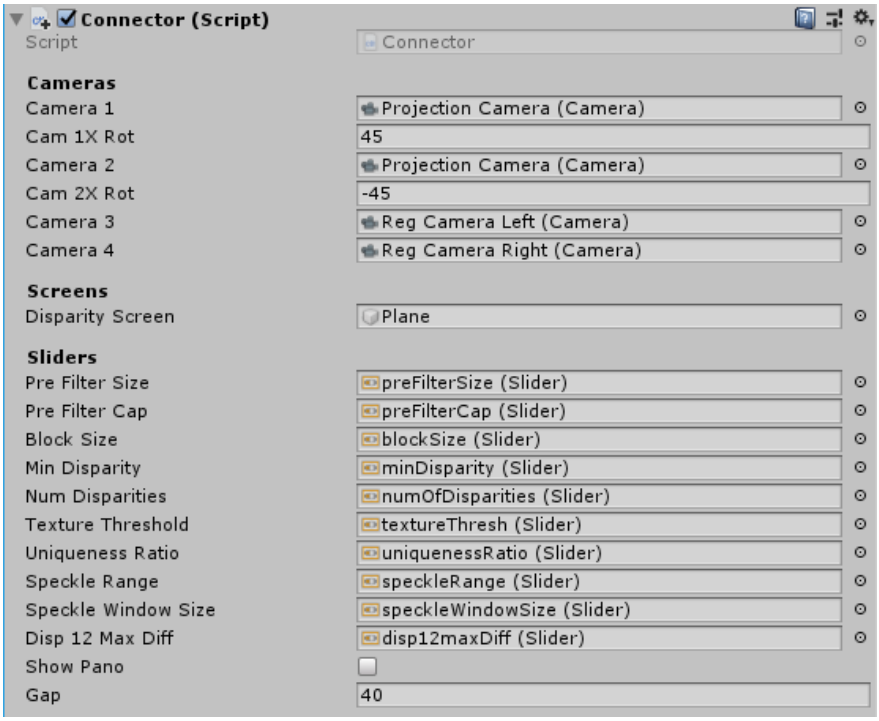

Рисунок 2.9 – Внешний вид окна настроек скрипта

Сразу после запуска симуляции происходит запуск дополнительных вычислительных потоков для обработки кадров с каждой пары камер и передача настроек скриптом в библиотеку для построения таблиц поиска. нескольких потоков позволяет не блокировать работу симуляции на время обработки изображений. Далее каждое обновление кадра скрипт считывает изображения с камер и помещает их в память соответствующего потока. Поток же работает независимо и обрабатывает каждую следующую пару изображений после готовности предыдущей.

2.5 Выводы по главе

Описан принцип устранения искажений сверхширокоугольных линз с выбором области интереса. Разработан алгоритм нахождения обратной проекции для fisheye-изображения. Описано устройство системы стереозре-

ния. Разработана её виртуальная модель в среде Unity вместе с алгоритмом передачи изображений с виртуальной камеры в программу обработки изображений... Виртуальная модель системы позволяет перейти к её испытаниям.

3 Экспериментальное исследование системы стереозрения

Описанная в предыдущей главе виртуальная модель системы стереозрения позволяет проводить с ней испытания для оценки работоспособности и сравнения с аналогами в контролируемой среде.

ЗАКЛЮЧЕНИЕ

3 Список литературы

1. Wenliang Gao and Shaojie Shen. Dual-fisheye omnidirectional stereo. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6715–6722, 2017.

2. Wenliang Gao, Kaixuan Wang, Wenchao Ding, Fei Gao, Tong Qin, and Shaojie Shen. Autonomous aerial robot using dual-fisheye cameras. *Journal of Field Robotics*, 37(4):497–514, 2020.

3. J. Kannala and S.S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1335–1340, 2006.

4. Menandro Roxas and Takeshi Oishi. Real-time variational fisheye stereo without rectification and undistortion, 2019.

5. D. Scaramuzza, A. Martinelli, and R. Siegwart. A flexible technique for accurate omnidirectional camera calibration and structure from motion. In *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, pages 45–45, 2006.

ПРИЛОЖЕНИЕ А

Программный код файла

```
#include ”opencv2/core.hpp”
#include ”opencv2/imgproc.hpp”
#include ”vector”

// Stores parameters for the camera
class Camera
{
  std::vector <double> polynom;   // Scaramuzza model
     coefficients
  cv::Vec2d centerOffset;       // Distortion center
  cv::Matx22d stretchMatrix;      // Whatever that is in the
     stretch matrix
  double lambda;            // Scale factor
  /* Structures */
  cv::Mat map1;   // x map
  cv::Mat map2;   // y map
  std::vector<cv::Point> frameBorder; //border to draw on
     original image


};

// On the second thought it is unnecessary - two fisheye
   dewarpers work as good. May be useful in a higher level class
   though.
```