МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий

Кафедра информационно-измерительных систем

Направление подготовки: 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

«РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЧЕСКОГО ОТСЛЕЖИВАНИЯ ПЕРЕМЕЩЕНИЯ ГРУППЫ ЛАБОРАТОРНЫХ РЫБ»

Липкин Евгений Олегович

«К защите допущен»
Заведующий кафедрой,
д.т.н.
Потатуркин О.И./……..
(фамилия , И., О.) / (подпись, МП)
«……»……2015г.

Научный руководитель
н.с. ИАиЭ СО РАН,
к.т.н.
Куликов В.А./……...
(фамилия , И., О.) / (подпись, МП)
«……»……2015г.

Дата защиты: «……» ……2015г.

Новосибирск, 2015г.

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)
Факультет информационных технологий

Кафедра информационно-измерительных систем

Направление подготовки: 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

УТВЕРЖДАЮ
Зав. Кафедрой Потатуркин О.И.
(фамилия, И., О.)
…….
(подпись, МП)
«……»……2015г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

Студенту Липкину Евгению Олеговичу

Тема: Разработка системы автоматического отслеживания перемещения группы лабораторных рыб

Цель работы: разработка инструментария получения трехмерных координат лабораторных рыб.

Структурные части работы: проанализировать проблему; выявить требования к системе; разработать и сконструировать аппаратную часть системы; реализовать программное обеспечение, способное вычислять трехмерные координаты рыб; разработать визуализацию перемещения лабораторных рыб в трехмерном пространстве.

Научный руководитель
н.с. ИАиЭ СО РАН,
к.т.н.
Куликов В. А./……..
(фамилия , И., О.) / (подпись)
«…»……2015г.

Задание принял к исполнению
Липкин Е.О./……..
(ФИО студента) / (подпись)
«…»……2015г.

Оглавление

ВВЕДЕНИЕ 4
ГЛАВА 1. Анализ задачи, требования к системе 6
1.1 Анализ задачи 6
1.2 Требования к системе 10
1.2.1 Требования к аппаратной части системы 10
1.2.2 Требования к программной части системы 11
1.2.3 Сценарии работы программного обеспечения 12
1.3 Программные средства 12
ГЛАВА 2. Аппаратная часть системы 14
2.1 Комплектующие 14
2.2 Стойка для камер 15
2.3 Система освещения 15
ГЛАВА 3. Методы цифровой обработки изображения 16
3.1 Глобальная система координат 16
3.2 Построение фона 17
3.3 Устранение шума 19
3.4 Пороговая бинаризация 19
3.5 Поиск связных областей 20
3.6 Фильтрация связных областей по площади 21
3.7 Поиск центров масс связных областей 21
3.8 Устранение оптических искажений 22
3.9 Сопоставление объектов 23
3.10 Получение трехмерных координат объектов 23
ГЛАВА 4. Реализация программной части системы 27
4.1 Сценарий пользования 27
4.2 Архитектура 27
4.3 Методы цифровой обработки изображений 28
4.4 Сохранение и загрузка данных 31
4.5 Интерфейс 31
4.6 Визуализация данных 33
ГЛАВА 5. Результаты 35
ЗАКЛЮЧЕНИЕ 36
ЛИТЕРАТУРА 37

ВВЕДЕНИЕ

В современном понимании этология – наука о поведении животных. Необходимость проведения этологических исследований главным образом определена сходством в поведении животных и поведении людей, как автономных субъектов, обладающих индивидуальной психической мотивацией, а также как элементов толпы, или как носителей массовых инстинктов, коллективных устремлений и чувств.

Биологи нуждаются в автоматизации этологических экспериментов. Разные виды животных и разные виды экспериментов определяют сложность разработки универсальной системы для решения задач подобного рода.

В лаборатории цифровых методов обработки изображений ИАиЭ СО РАН разработана зарекомендовавшая себя на рынке система автоматизации этологических тестов EthoStudio[1][2], которая представляет собой аппаратно-программный комплекс, включающий в себя многофункциональную установку для тестирования животных и компьютер с программным обеспечением. Однако, в этой системе отсутствует инструментарий, позволяющий автоматизировать наблюдение за лабораторными рыбами.

У ЛИН СО РАН существует задача автоматического отслеживания перемещения группы лабораторных рыб для изучения акустической чувствительности. Эта задача сводится к определению трехмерных координат отдельных особей во время эксперимента.

На данный момент большинство систем, автоматизирующие этологические эксперименты на лабораторных рыбах, построены на использовании одной единственной камеры. Задача получения трехмерных координат рыб решена только в виде системы, способной автоматизировать наблюдение лишь за одной лабораторной рыбой.

Рыбы двигаются в трехмерном пространстве, поэтому требуется реализовать инструментарий получения трехмерных координат объектов. Для решения задачи получения трехмерных координат объекта используют системы из нескольких камер, расположенных на фиксированном расстоянии друг от друга, для получения стереопары (изображений с разных ракурсов), с помощью которой определяют трехмерные координаты, используя алгоритм построения карты диспарантности или метод триангуляции.

Цель дипломной работы состоит в разработке инструментария получения трехмерных координат лабораторных рыб.

Для достижения цели мной выделены следующие задачи:

- Проанализировать проблему;

- Выявить требования к системе;
- Разработать и сконструировать аппаратную часть системы;
- Реализовать программное обеспечение, способное вычислять трехмерные координаты рыб;
- Разработать визуализацию перемещения лабораторных рыб в трехмерном пространстве.

В работе был реализован программный модуль, вычисляющий методом триангуляции трехмерные координаты аквариумных рыб по их двухмерным координатам на кадрах с видеокамер. Этот программный модуль позволяет системе EthoStudio автоматизировать этологические исследования с лабораторными рыбами.

Работа изложена в пяти главах. В первой главе описан анализ задачи и требования к системе. Во второй главе описана аппаратная часть системы. В третьей главе описаны алгоритмы, использованные при разработке программной части системы. В четвертой главе описана реализация программной части системы. В пятой главе описаны результаты работы.

Дипломная работа выполняется в лаборатории цифровых методов обработки изображений ИАиЭ СО РАН в рамках заказного проекта «Система автоматического отслеживания перемещения группы лабораторных рыб».

ГЛАВА 1. Анализ задачи, требования к системе

1.1 Анализ задачи

На текущий момент системы, автоматизирующие этологические эксперименты, построены на использовании одной единственной камеры. Задача получения трехмерных координат рыб решена только в виде системы, способной автоматизировать наблюдения лишь за одной лабораторной рыбой.

На мировом рынке системы автоматизации этологических исследований на аквариумных рыбах разрабатывает компания «ViewPoint». В России дистрибьютором продуктов этой компании является компания «Vivariy.com». Компания «ViewPoint» разработала системы для отслеживания перемещения рыб «ZebraLab» и «ZebraLab3D». Эти системы специализируются на автоматическом определении координат взрослых особей вида данио-рерио. Система «ZebraLab» позволяет получать двухмерные координаты аквариумных рыб и работать с ними. На рисунке 1.1 изображен скриншот работы программы «ZebraLab». Система «ZebraLab3D» позволяет получать трехмерные координаты одной единственной аквариумной рыбы, так как система использует пару камер, одна из которых помещается над аквариумом для наблюдения за лабораторной рыбой сверху, а вторая камера помещается с фронтальной стороны аквариума для наблюдения за лабораторной рыбой сбоку. На рисунке 1.2 изображен скриншот работы программы «ZebraLab3D».

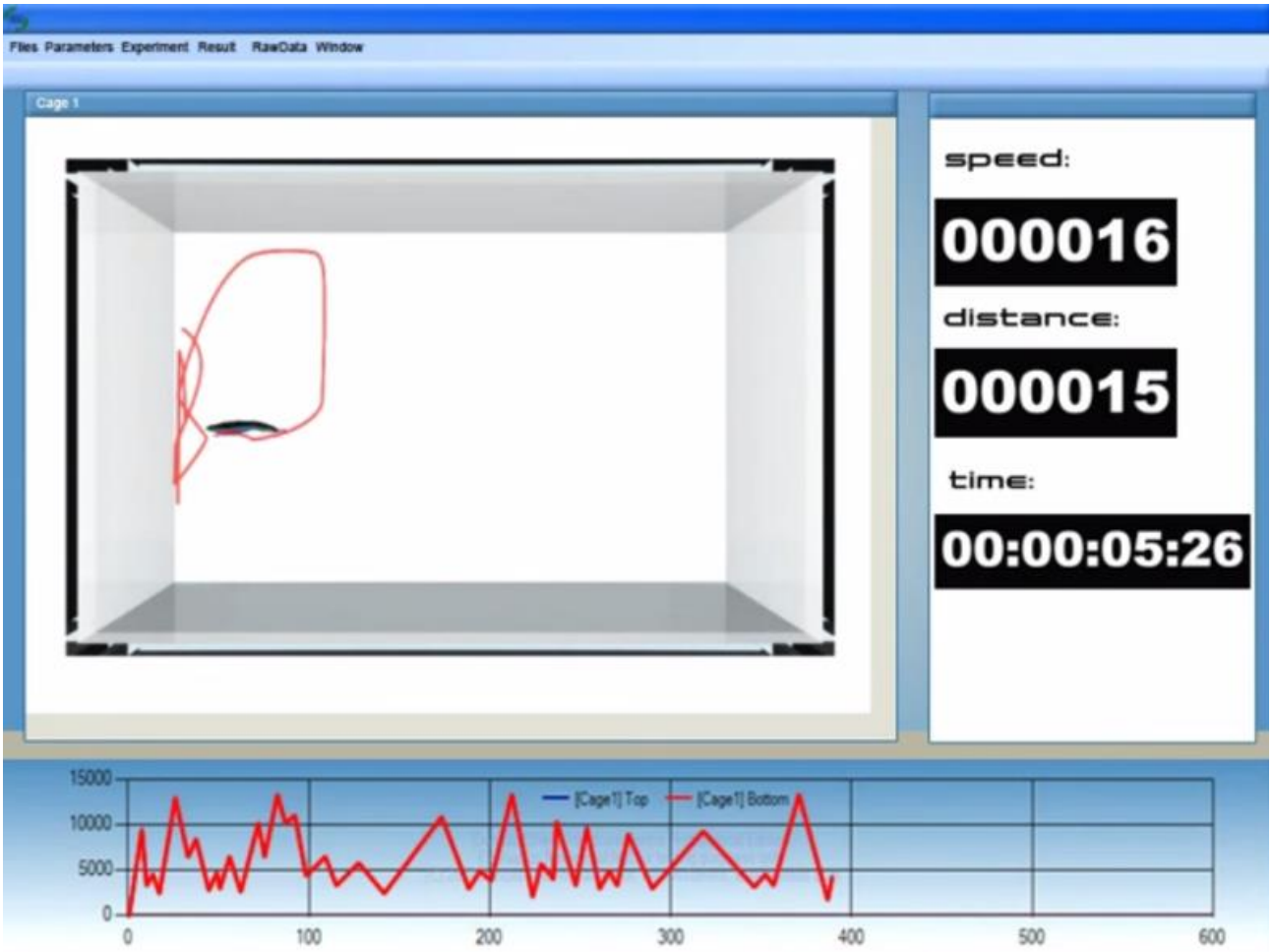

Рисунок 1.1. Скриншот программы «ZebraLab». В центральной части окна программы отображается аквариум, лабораторная рыба и траектория движения лабораторной рыбы. В правой части окна программы выводятся параметры эксперимента: скорость; пройденная дистанция; промежуток времени, прошедший с начала старта эксперимента

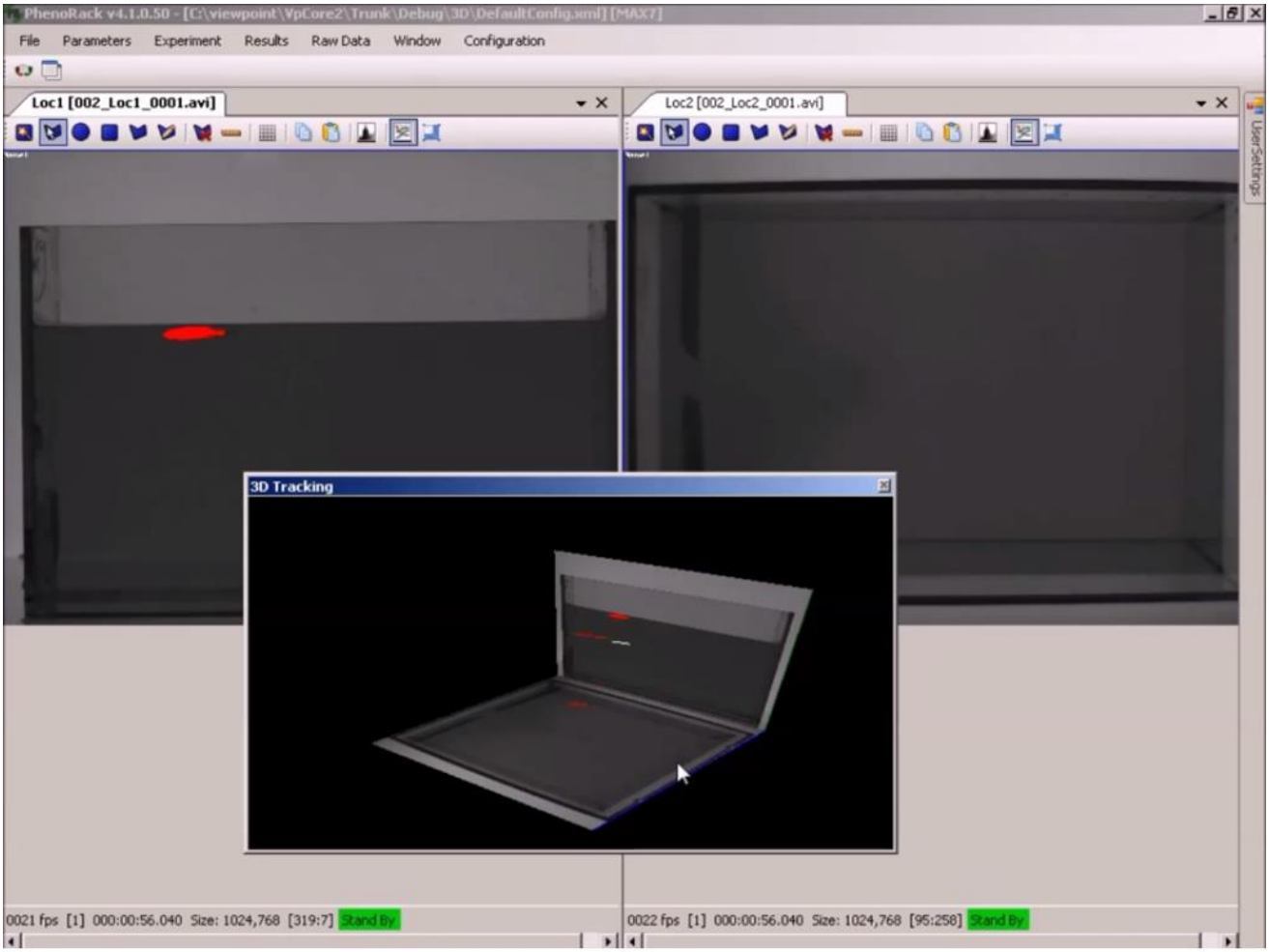

Рисунок 1.2. Скриншот программы «ZebraLab3D». В левой части окна программы отображается последовательность кадров с камеры, захватывающей видеопоток с фронтальной стороны, а в правой части окна отображается последовательность кадров с камеры, захватывающей видеопоток над аквариумом. Также можно видеть на скриншоте наложение кадров с камер в трехмерном пространстве для создания визуализации перемещения лабораторной рыбы

Для задач этологических исследований используют модельный организм биологии развития – вид данио-рерио(zebrafish)[3]. Это популярная лабораторная рыбка, являющаяся стандартным модельным организмом в исследованиях по эмбриологии и стволовым клеткам. Геном данио-рерио секвенирован, что упрощает ее генетическую модификацию. Благодаря тому, что биологи очень хорошо знают внутреннее строение лабораторных рыб этого вида, многие биологи предпочитают использовать их для своих экспериментов. Спектр решаемых задач достаточно широк: пищевое поведение, акустическое поведение, апробация

медицинских препаратов, тестирование реакции на химические вещества (экология), бихевиоризм (лидерство, стадность) и другие различные типы экспериментов. Для задачи изучения акустической чувствительности подходят взрослые особи вида данио-рерио.

Биологи ЛИН СО РАН проводят исследования акустической чувствительности лабораторных рыб в аквариумах. Распространенным решением задачи получения трехмерных координат объектов является стереоскопия. На основе двух кадров, сделанных камерой или камерами, можно определить относительное положение объектов. Так как нам необходимо отслеживать перемещение рыб, требуется работа с последовательностями кадров. Мы используем пару камер, расположенных на фиксированном расстоянии друг от друга.

Датчики глубины с инфракрасной подсветкой (в частности, Microsoft Kinect) не подходят для решения задачи по причине искажений инфракрасной сетки в среде аквариума. На рисунке 1.3 изображен кадр с камеры глубины Microsoft Kinect. А на рисунке 1.4 изображена карта глубины поверх кадра цветной видеокамеры.

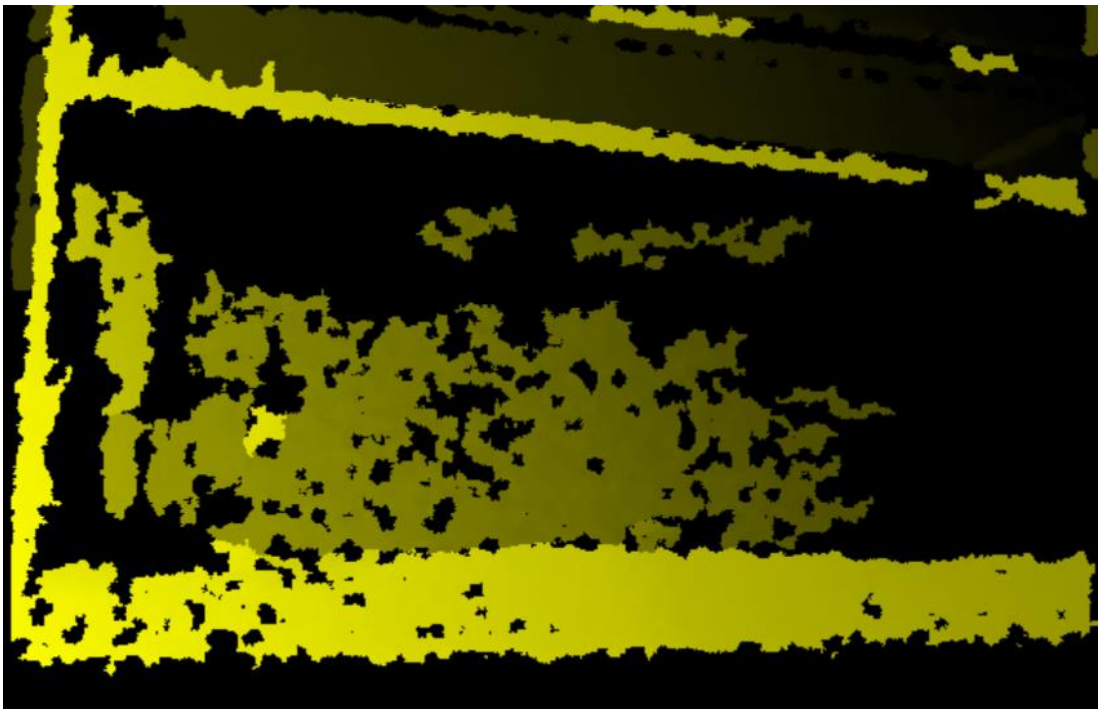

Рисунок 1.3. Кадр датчика глубины Microsoft Kinect. Темные точки – неизвестные расстояния, шумы. Светлые точки – расстояния, чем темнее – тем дальше

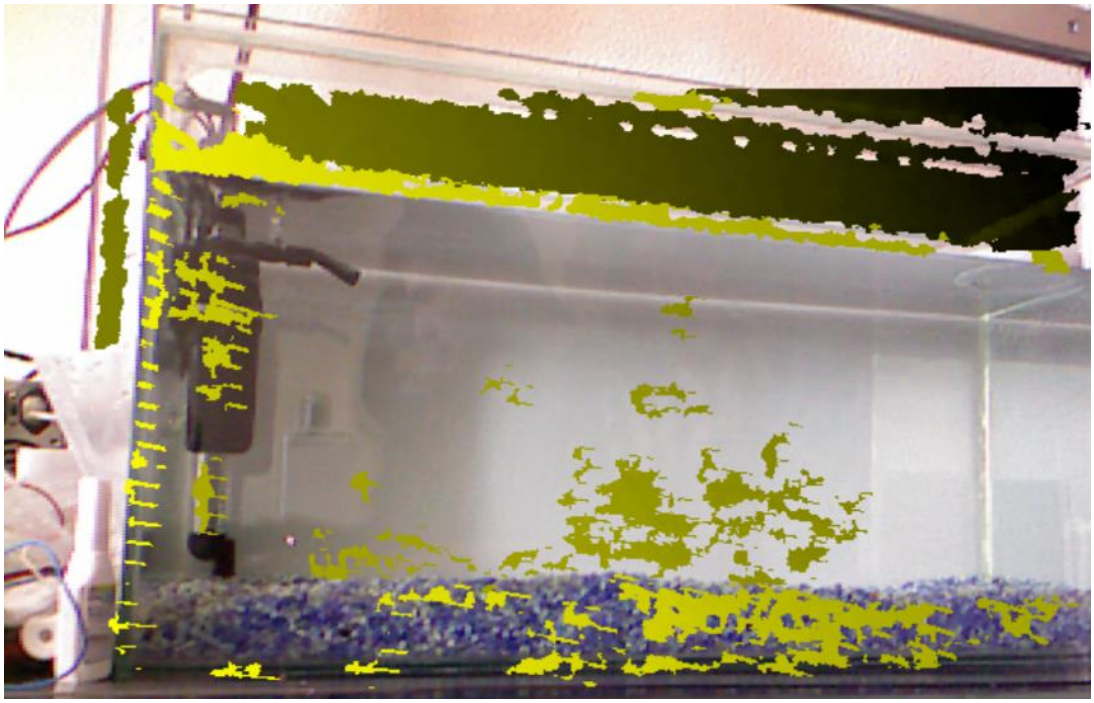

Рисунок 1.4. Аналогично с предыдущим изображением вместо черных точек с неизвестной глубиной подставлено цветное изображение. Видно, что большая часть кадра датчика глубины зашумлена

Задачу можно решить, используя алгоритм построения карты диспарантности или метод триангуляции. Так как решение задачи не нуждается в реконструкции трехмерной структуры сцены, а нужно только определять трехмерные координаты наблюдаемых объектов, то для решения задачи больше подойдет метод триангуляции точек. Алгоритм построения карты диспарантности будет производить лишние операции при восстановлении всего кадра, что скажется на скорости работы системы.

1.2 Требования к системе

1.2.1 Требования к аппаратной части системы

Выявлены следующие требования к аппаратной части системы:

- Аппаратная часть системы должна представлять собой установку, состоящую из стойки для камер, системы освещения, аквариума, оборудования для обеспечения жизнедеятельности в аквариуме и компьютера;
- Стойка для камер должна уметь регулировать положение камер для возможности калибровки камер под произвольный аквариум;

- Система освещения должна обеспечивать контрастность между фоном и наблюдаемыми объектами.

На рисунке 1.3 изображена модель макета аппаратной части системы.

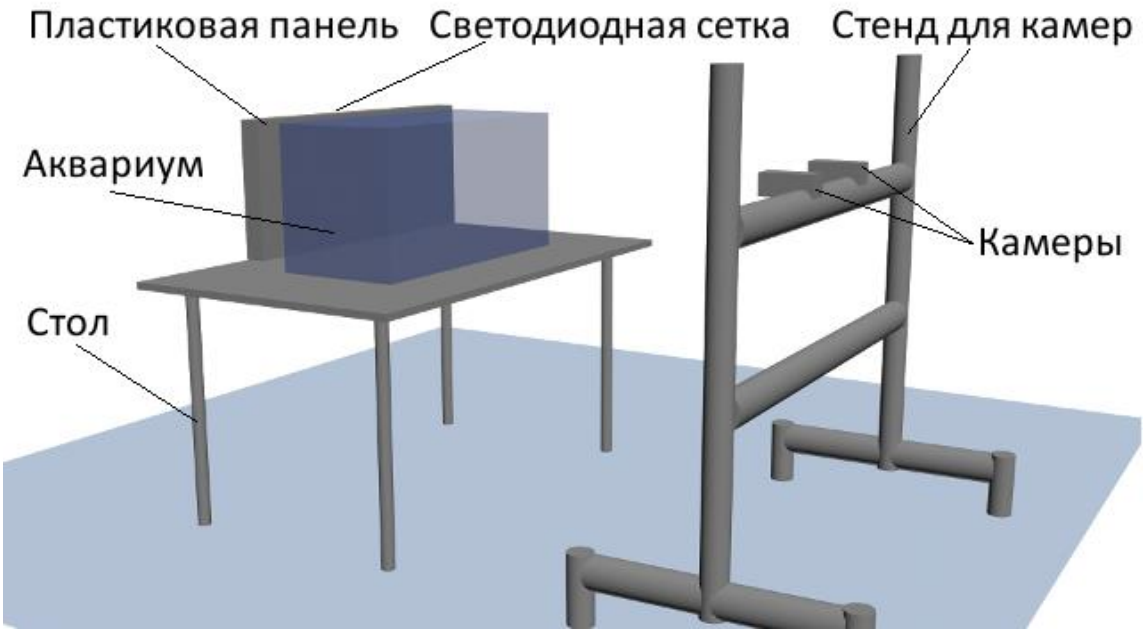

Рисунок 1.5. Модель макета аппаратной части системы

1.2.2 Требования к программной части системы

Программная часть системы должна представлять собой программное обеспечение, обладающее инструментарием для получения трехмерных координат трех взрослых особей вида данио-рерио и визуализацией их в трехмерном пространстве. При перекрытии лабораторных рыб на изображении, будем принимать за их двухмерные координаты центр масс связной области, образованной пересечением лабораторных рыб на изображении.

Получение трехмерных координат объектов должно быть реализовано с помощью метода триангуляции. Система должна определять трехмерные координаты объектов с точностью до 2 см.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

- функции запуска отслеживания перемещения группы лабораторных рыб системой;
- функции остановки запуска отслеживания перемещения группы лабораторных рыб системой;

- функции визуализации данных о местоположении лабораторных рыб в произвольный момент времени на трехмерной карте.

Пользовательский интерфейс программы должен обеспечивать возможность выполнения перечисленных функций.

1.2.3 Сценарии работы программного обеспечения

Основной сценарий работы программного обеспечения состоит из нескольких шагов:

- запуск программы;
- выбор параметров работы алгоритма автоматического отслеживания перемещения;
- калибровка системы при надобности;
- запуск алгоритма;
- сохранение стереоизображения группы лабораторных рыб при надобности;
- визуализация объектов в трехмерном пространстве;
- по истечению времени эксперимента останавливаем алгоритм отслеживания перемещения.

Другой сценарий работы алгоритма заключается в следующем:

- загрузка стереоизображения группы лабораторных рыб;
- визуализация объектов в трехмерном пространстве.

1.3 Программные средства

Для реализации программного обеспечения мы использовали следующие средства разработки:

- **C++98** - компилируемый статически типизированный язык программирования общего назначения.
- **Qt 5.2.1** - кроссплатформенный инструментарий разработки ПО на языке программирования C++[4][5].
- **OpenCV 2.4.10** - библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++.
- **Point Grey FlyCapture 2.0 SDK** - библиотека, разработанная специально для упрощения работы с камерами Point Grey.

- **EthoStudio SDK** - система автоматизации этологических тестов EthoStudio представляет собой аппаратно-программный комплекс, включающий в себя многофункциональную установку для тестирования животных и компьютер с программным обеспечением. Реализована на C++ с использованием кроссплатформенной библиотеки Qt, библиотеки компьютерного зрения OpenCV и другого инструментария.
- **OpenGL 2.0** - спецификация, определяющая независимый от языка программирования платформонезависимый программный интерфейс для написания приложений, использующих двухмерную и трехмерную компьютерную графику[6][7].
- **Microsoft Visual C++ Compiler 10.0** - компилятор языка программирования C++ от компании Microsoft.

ГЛАВА 2. Аппаратная часть системы

2.1 Комплектующие

Для тестовой системы мы используем следующее оборудование:

- Стол;
- Аквариум 700x250x400 мм^3;
- Фильтр, компрессор, терморегулятор;
- Система освещения;
- Стойка для камер;
- Две камеры Point Grey FL3-U3-32S2 M-CS с кадровой частотой – 60 кадров в секунду, разрешаемой способностью – 1600x1200 пикселей и восьми разрядным АЦП;
- Компьютер со следующими техническими характеристиками: операционная система Windows 7, процессор AMD Phenom II x6 1055T 2.8 ГГц, 8 ГБ ОЗУ, видеокарта NVIDIA GeForce GTS 450 с 4 ГБ памяти и 100 МБ свободного места на жестком диске.

На рисунке 2.1 изображена фотография системы.

Рисунок 2.1. Фотография системы

2.2 Стойка для камер

При сборке стойки для камер была использована система алюминиевых профилей для торгово-выставочного оборудования "Consta-Sib" компании ЗАО "СИБ.ПРОФИЛЬ", предназначенного для изготовления профильных конструкций. Был спроектирован и собран стенд, согласно проекту.

2.3 Система освещения

Была изготовлена система освещения, представляющая собой корпус из пористого ПВХ 700x400x100 мм^3. Передняя панель системы освещения состоит из белого акрилового оргстекла для рассеивания света. Весь корпус, не включая переднюю панель, обклеен изнутри фольгой. Поверх фольги по контуру корпуса изнутри размещена светодиодная сетка. Система освещения разработана согласно методу теневой проекции и размещается за аквариумом для обеспечения контрастности между наблюдаемыми объектами и фоном.

ГЛАВА 3. Методы цифровой обработки изображения

Общий алгоритм работы программы следующий. Есть две камеры, с каждой берется видеопоток. Для каждого видеопотока выполняется следующий алгоритм: находятся объекты и определяются двухмерные координаты этих объектов. Далее, методом триангуляции определяются трехмерные координаты объектов и визуализируются в трехмерном пространстве.

На рисунке 3.1 изображена блок-схема алгоритма работы программы.

Рисунок 3.1. Алгоритм работы программы

3.1 Глобальная система координат

Глобальная система координат задается следующим образом. У нас есть калибровочная доска – доска с закрашенными кругами в прямоугольной сетке размером 9 * 8. Размещаем калибровочную доску так, чтобы все круги полностью попадали в кадры видеопотоков обеих камер с известным расстоянием в 23.6 миллиметра. Центр верхнего левого круга мы принимаем за начало глобальной системы координат. Эта доска задает систему координат аквариума. На рисунке 3.2 видно, как задана система координат. Оси *OX* и *OY* направлены в плоскости калибровочной доски, а ось *OZ* направлена перпендикулярно доске.

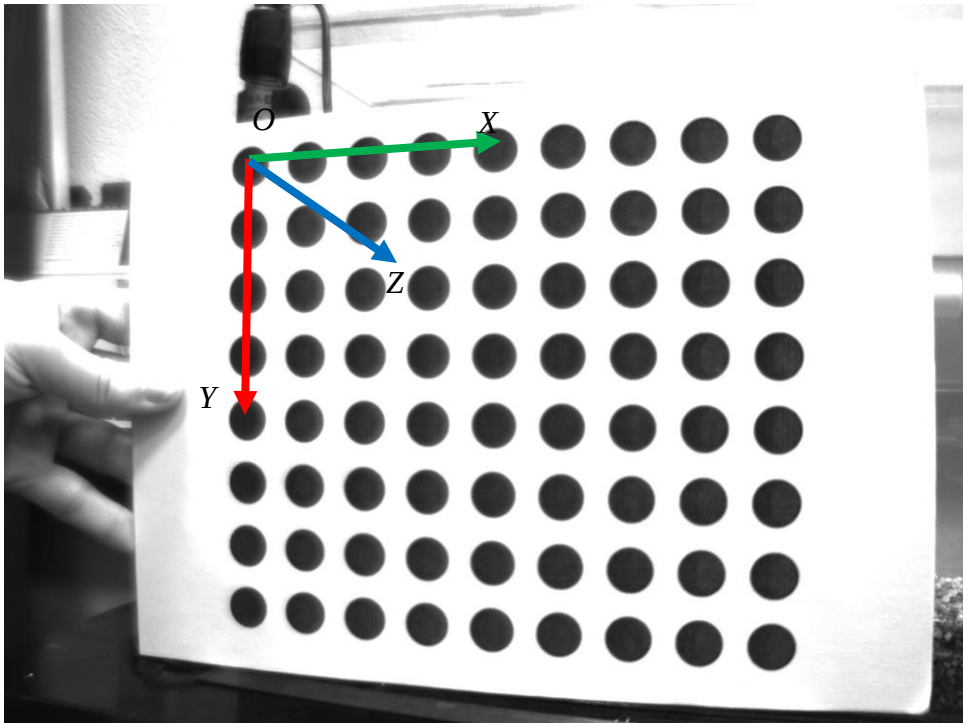

Рисунок 3.2. Калибровочная доска с нарисованной системой координат

3.2 Построение фона

Нам нужно выделить наблюдаемые объекты, для этого нужно построить фон. Пусть есть серия серых изображений $I_l(x, y, t)$ и $I_r(x, y, t)$ для левой и правой камер соответственно. Фон строится следующим образом: Для каждой точки фона мы вычисляем медианный элемент последовательности из 50 кадров с частотой 5 кадров в секунду:

$$BG(x, y) = median_{t=1..50}(I(x, y, t)),$$

где $I(x, y, t)$ – яркость t-того изображения в координатах x и y, $BG(x, y)$ – яркость фона в координатах x и y, а *median* – функция вычисления медианного элемента.

Вычитая фон из исходного изображения, мы получаем изображение движущихся объектов, так как лабораторные рыбы постоянно двигаются. На рисунке 3.3 изображен пример кадра с видеокамеры системы. На рисунке 3.4 изображено разностное изображение движущихся объектов.

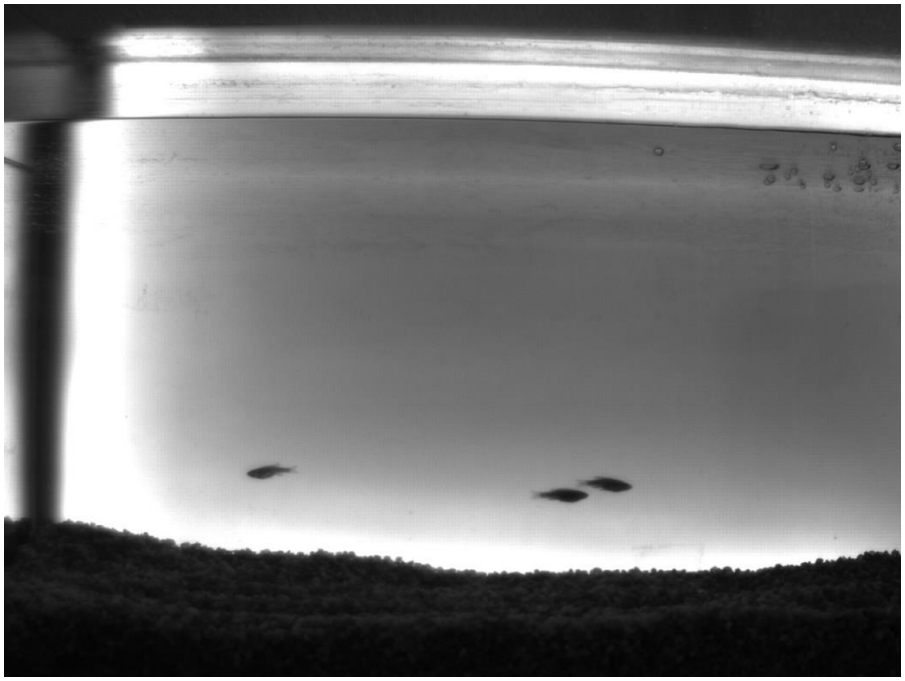

Рисунок 3.3. Пример изображения с видеокамеры. На кадре заснят аквариум с тремя лабораторными рыбами. На дне аквариума лежит гравий, а справа вверху можно заметить пузыри от компрессора

Рисунок 3.4. Разностное изображение движущихся объектов. На изображении хорошо видны лабораторные рыбы, а справа вверху чуть заметны пузыри от компрессора

3.3 Устранение шума

Следующий этап алгоритма – получение двухмерных координат объектов. В качестве координат объектов мы используем центры масс связных областей. Связные области ищутся на бинаризованном изображении. Прежде, чем бинаризовать изображение, нам нужно устранить шум на разностном изображении движущихся объектов. Для этого нам потребуется размыть изображение.

Размытие изображения является усреднением по прямоугольной области размером *(w * h)*. Каждый пиксель на выходе является средним арифметическим пикселей в области *(w * h)*, где *w* и *h* – размеры прямоугольной области размытия.

3.4 Пороговая бинаризация

Проводим бинаризацию разностного изображения движущихся объектов после удаления части шума:

$$B(x,y,t) = \begin{cases} 1, & \text{если } |I(x,y,t) - BG(x,y)| \geq m \\ 0, & \text{иначе} \end{cases},$$

где *x* и *y* – координаты точки фона, *I(x, y, t)* – яркость *t*-того изображения в координатах *x* и *y*, $B(x,y,t)$ – точка бинаризованного *t*-того изображения с координатами *x* и *y*, а m – порог бинаризации, который выбирается методом Оцу[8]. На рисунке 3.5 изображено бинаризованное изображение движущихся объектов.

Рисунок 3.5. Бинаризованное изображение движущихся объектов по порогу *m*. Единицами заполнены точки изображения, в которых находятся движущиеся объекты. Нулями заполнены остальные точки изображения

3.5 Поиск связных областей

Для определения двухмерных координат объектов нам нужно вычислить центры масс связных областей[9]. Для вычисления центров масс связных областей нужно произвести поиск связных областей на бинаризованном изображении. В результаты мы положим маркированные изображения:

$$L(x,y,t) = \begin{cases} 0, \text{если } B(x,y,t) = 0 \\ k, \text{если } B(x,y,t) \in \text{связной области с индексом } k' \end{cases}$$

где *L(x, y, t)* – точка *t*-того изображения связных областей в координатах *x* и y, *B(x, y, t)* – точка бинаризованного *t*-того изображения в координатах *x* и *y*, *k* – номер связанной области. На рисунке 3.6 изображены контуры связных областей.

Рисунок 3.6. Изображение контуров связных областей

3.6 Фильтрация связных областей по площади

Чтобы убедиться, что на изображении остались движущиеся объекты, необходимо отсечь случайные выбросы. Для этого зададим возможный диапазон площадей связных областей:

$$L'(x,y,t) = \begin{cases} k, \text{если } Smax > S(L,k,t) > Smin \\ 0, \text{иначе} \end{cases},$$

где $L'(x,y,t)$ – точка t-того изображения связных областей в координатах x и y, прошедшего фильтрацию; k – номер связанной области; $Smin$ и $Smax$ – границы наименьшей и наибольшей площадей соответственно, а $S(L,k,t)$ – площадь связной области, которая вычисляется по формуле:

$$S(L,k,t) = \sum_x \sum_y (L(x,y,t) = k).$$

3.7 Поиск центров масс связных областей

Центры масс связных областей, прошедших фильтрацию, мы принимаем за двухмерные координаты объектов[9]:

$$x_c(k,t) = \frac{1}{S(L',k,t)} \sum_x \sum_y x(L'(x,y,t) = k),$$

$$y_c(k,t) = \frac{1}{S(L',k,t)} \sum_x \sum_y y(L'(x,y,t) = k).$$

где $x_c(k,t)$ и $y_c(k,t)$ – координаты центра масс *k*-той связной области на *t*-том изображении связных областей, прошедшем фильтрацию; $L'(x,y,t)$ – множество связных областей, прошедших фильтрацию; $S(L,k,t)$ – площадь *x* и *y* – координаты центра связной области. На рисунке 3.7 изображены контуры связных областей и их центры масс.

Рисунок 3.6. Изображение контуров связных областей и их центров масс

3.8 Устранение оптических искажений

В теории, можно определить объектив, который воспроизводит без искажений. На практике линзы не идеальны. Будем учитывать два основных типа искажения: радиальное искажение (radial distortions), которое возникает из-за формы линзы; и тангенциальное искажение (tangential distortions), которое возникает в процессе сборки камеры.

Запишем формулу[10], позволяющую устранить оптические искажения:

$$x_{corrected}(k,t) = x_c(k,t)(1 + k_1r^2 + k_2r^4 + k_3r^6) + \left(2p_1y_c(k,t) + p_2(r^2 + 2x_c(k,t)^2)\right)$$

$$y_{corrected}(k,t) = y_c(k,t)(1 + k_1r^2 + k_2r^4 + k_3r^6 + (p_1(r^2 + 2y_c(k,t)^2) + 2p_2x_c(k,t)))$$

где $x_{corrected}(k,t)$ и $y_{corrected}(k,t)$ – координаты *k*-того центра масс связной области t-того изображения с устраненными оптическими искажениями, $x_c(k,t)$ и $y_c(k,t)$ – координаты *k*-того центра масс связной области t-того изображения, $k_1,\ k_2,\ k_3,$ – параметры радиального искажения, $p_1,\ p_2$ – параметры тангенциального искажения.

3.9 Сопоставление объектов

Для алгоритма триангуляции нужны пары координат одного объекта с разных камер. Так как стенд спроектирован таким образом, что камеры расположены примерно вдоль оси *OX* (Рисунок 3.2), то можно записать формулу взаимно однозначного отображения индексов объектов на паре изображений с разных камер по их двухмерным координатам:

$$\pi(k,t) = \begin{pmatrix} 1 & \dots & N \\ g(1,t) & \dots & g(n,t) \end{pmatrix},$$

где $\pi(k,t)$ – подстановка индексов, N – количество объектов,

$$g(k,t) = argmin_{\pi(k,t)} \left| x^{(l)}_{corrected}(k,t) - x^{(r)}_{corrected}(\pi(k,t),t) \right|.$$

Это означает, что каждому объекту в координатах $\left(x^{(l)}_{corrected}(k,t), y^{(l)}_{corrected}(k,t)\right)$ с кадра одной камеры соответствует уникальный объект в координатах $\left(x^{(r)}_{corrected}(k,t), y^{(r)}_{corrected}(k,t)\right)$, и пары этих точек будут приняты за двухмерные координаты одних и тех же объектов с кадров разных камер.

3.10 Получение трехмерных координат объектов

Зная параметры камер и двухмерные координаты объектов, мы можем решить задачу триангуляции, которая сводится к решению системы линейных уравнений методом наименьших квадратов[11]. На рисунке 3.8 изображена стереоскопическая система.

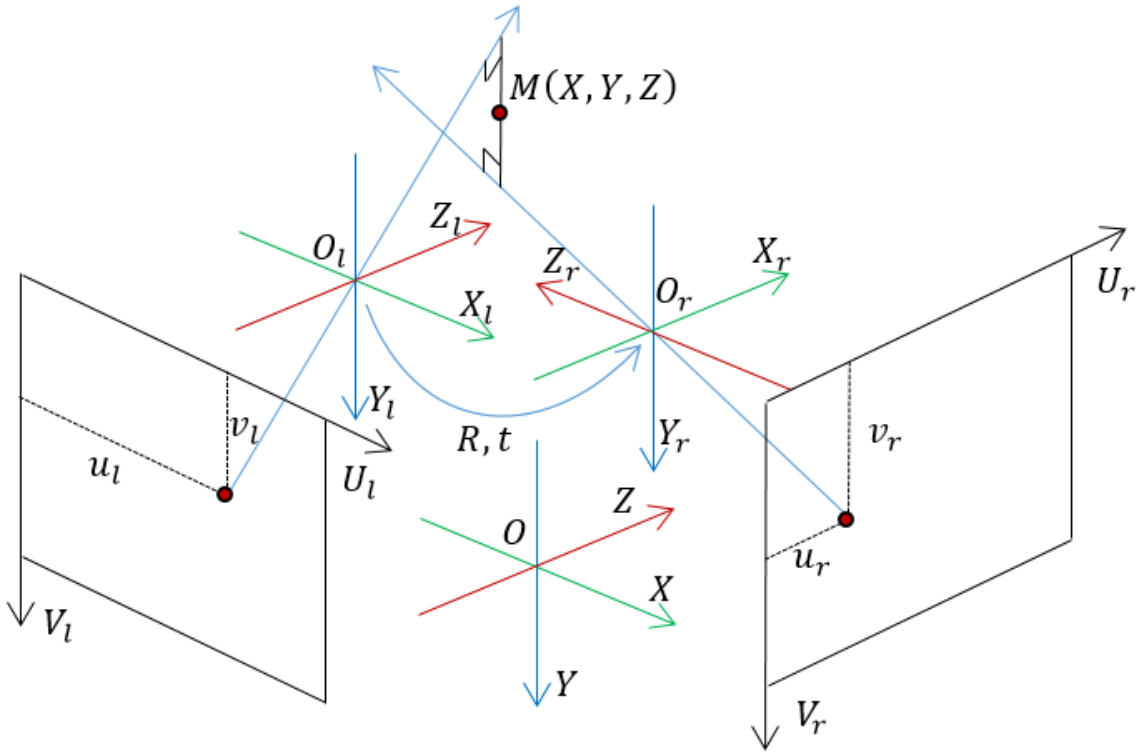

Рисунок 3.8. Изображение стереоскопической системы. $OXYZ$ – глобальная система координат, $O_lX_lY_lZ_l$ и $O_rX_rY_rZ_r$ – системы координат левой и правой камер, (u_l, v_l) и (u_r, v_r) – двухмерные координаты объекта в системах координат плоскостей изображений левой и правой камер, $M = (X, Y, Z)^T$ – трехмерные координаты объекта

Имеем пару камер. Будем их различать, как левая камера и правая камера. К каждой камере известны внутренние A_l, A_r и внешние параметры R_l, R_r, t_l, t_r в глобальной системе координат.

Введем для каждой камеры свою стандартную систему координат: левой камере соответствует система координат $O_lX_lY_lZ_l$, а правой – $O_rX_rY_rZ_r$. Пусть вектор $M_l = (X_l, Y_l, Z_l)^T$ определяет координаты некоторой точки $M = (X, Y, Z)^T$ трехмерного пространства в системе координат левой камеры, а вектор $M_r = (X_r, Y_r, Z_r)^T$ – в системе координат правой камеры.

Легко убедиться, что в системе координат левой камеры проекцией точки M_l является точка в плоскости изображения с координатами (u_l, v_l):

$$u_l = \frac{f_l X_l}{w_l Z_l} + u_{l0}, \; v_l = \frac{f_l Y_l}{w_l Z_l} + v_{l0},$$

где f_l – расстояние, на котором находится плоскость изображения от оптического центра, w и h – масштабы вдоль осей U_l и V_l, (u_{l0}, v_{l0}) – координаты главной точки относительно начала координат фотоприемника.

Используя следующие обозначения:

$$A_l = \begin{pmatrix} f_l/w_l & 0 & u_{l0} \\ 0 & f_l/h_l & v_{l0} \\ 0 & 0 & 1 \end{pmatrix},$$

$$p_l = \begin{pmatrix} u_l & v_l & 1 \end{pmatrix}^T;$$

можно записать соотношение, определяющее проекцию точки, в виде:

$$Z_l p_l = A_l M_l.$$

Аналогично и для правой камеры:

$$A_r = \begin{pmatrix} f_r/w_r & 0 & u_{r0} \\ 0 & f_r/h_r & v_{r0} \\ 0 & 0 & 1 \end{pmatrix},$$

$$p_r = \begin{pmatrix} u_r & v_r & 1 \end{pmatrix}^T,$$

$$Z_r p_r = A_r M_r.$$

К каждой камере известны внешние параметры R_l, R_r, t_l, t_r. R_l, R_r – матрицы размерности 3 * 3, описывающие повороты стандартных систем координат левой и правой камер соответственно относительно глобальной системы координат. t_l, t_r – трехмерные вектора смещения начала координат глобальной системы относительно начала координат стандартных систем координат левой и правой камер соответственно.

Переход от глобальной системы координат к стандартным системам координат левой и правой камер осуществляется с помощью следующих преобразований:

$$M_l = R_l M + t_l, \; M_r = R_r M + t_r.$$

Учитывая это, связь между векторами M_l и M_r задается соотношением:

$$M_r = R M_l + t,$$

где

$$R = R_r {R_l}^T,$$

$$t = -R t_t + t_r.$$

R - ортогональная матрица, описывающая ориентацию системы координат правой камеры относительно левой, а t - вектор трансляции, определяющий положение оптического центра правой камеры в системе координат левой.

Используя соотношения

$$Z_l p_l = A_l M_l,\ Z_r p_r = A_r M_r \text{ и } M_r = R M_l + t$$

запишем

$$Z_r p_r {A_r}^{-1} = R Z_l p_l {A_l}^{-1} + t.$$

Поскольку компоненты векторов p_l и p_r могут содержать ошибки, реальное соотношение принимаем вид:

$$-R Z_l p_l {A_l}^{-1} + Z_r p_r {A_r}^{-1} - t = e,$$

где e – вектор невязки, обусловленный наличием ошибок измерений.

Для оценивания неизвестных Z_l и Z_r можно воспользоваться методом наименьших квадратов, который может использоваться для «решения» системы линейных уравнений:

$$\mathcal{A}x = \mathcal{b},$$

$$x = \mathcal{A}^{+}\mathcal{b},$$

$$x = (\mathcal{A}^T\mathcal{A})^{-1}\mathcal{A}^T\mathcal{b}.$$

Оценим Z_l и Z_r методом наименьших квадратов, подставив следующие величины:

$$x = \begin{pmatrix} Z_l \\ Z_r \end{pmatrix},\ \mathcal{A} = \begin{pmatrix} -R p_l {A_l}^{-1} & p_r {A_r}^{-1} \end{pmatrix},\ \ \mathcal{b} = t.$$

Получаем:

$$\begin{pmatrix} Z_l \\ Z_r \end{pmatrix} = \begin{pmatrix} -R p_l {A_l}^{-1} & p_r {A_r}^{-1} \end{pmatrix}^{+} t,$$

или

$$\begin{pmatrix} Z_l \\ Z_r \end{pmatrix} = \begin{pmatrix} {p_l}^T \left({A_l}^{-1}\right)^T {A_l}^{-1} p_l & -{p_l}^T \left({A_l}^{-1}\right)^T R^T {A_r}^{-1} p_r \\ -{p_l}^T \left({A_l}^{-1}\right)^T R {A_r}^{-1} p_r & {p_r}^T \left({A_r}^{-1}\right)^T {A_r}^{-1} p_r \end{pmatrix}^{-1} \begin{pmatrix} -{p_l}^T {A_l}^{-1} \\ p_r \left({A_r}^{-1}\right)^T \end{pmatrix} t.$$

В системе координат калибровочной доски:

$$\begin{pmatrix} M_l \\ M_r \end{pmatrix} = \begin{pmatrix} {R_l}^T (Z_l {A_l}^T p_l - t_l) \\ {R_r}^T (Z_r {A_r}^T p_r - t_r) \end{pmatrix}.$$

За трехмерную координату объекта мы принимаем точку, лежащую по центру между M_l и M_r:

$$M = (M_l + M_r)/2.$$

ГЛАВА 4. Реализация программной части системы

4.1 Сценарий пользования

После запуска программы для работы пользователю необходимо выбрать данные, с которыми он будет работать. Нужно выбрать захват видеопотоков с камер или загрузить стереоизображение. Далее нужно выбрать файл внутренних и внешних параметров камер. Теперь пользователь может нажать кнопку «Start», и при успешном получении доступа к исходным данным он сможет начать работу с системой.

Пользователь может:

- установить параметры метода триангуляции;
- откалибровать систему;
- отобразить не более трех стереоизображений с выбранными методами обработки;
- сохранить стереоизображение и параметры камер.

4.2 Архитектура

Архитектура программного обеспечения построена на паттерне Модель-Представление (Model-View), как и большинство программ, написанных с использованием библиотеки Qt. На рисунке 4.1 изображена архитектура программы.

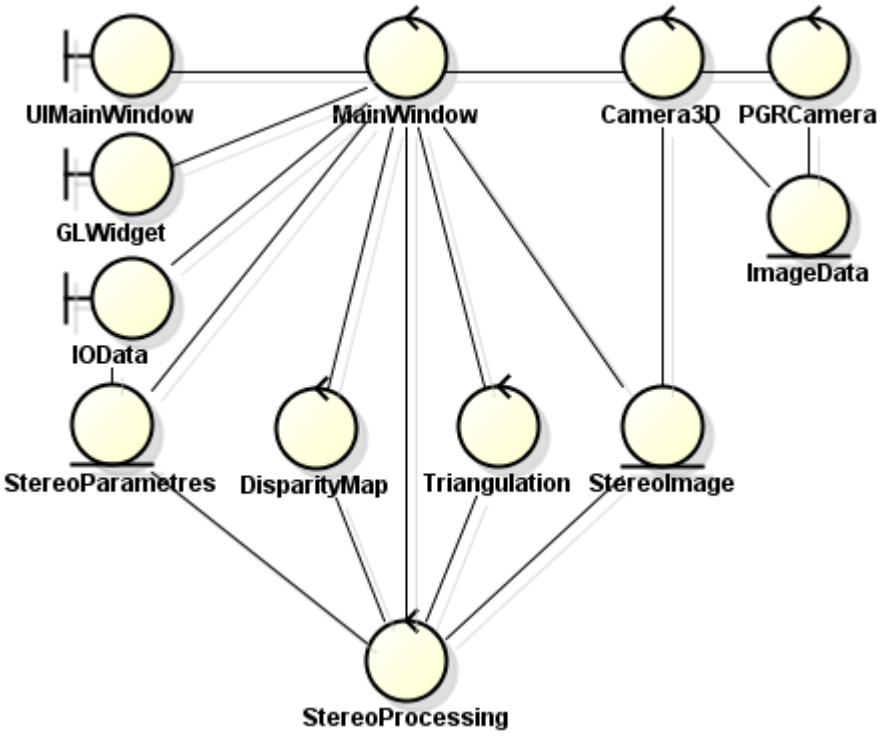

Рисунок 4.1. Изображение архитектуры программы

Класс **MainWindow** отвечает за логику представления программы. Классы **UIMainWindow** и **GLWidget** отвечают за представления основного окна и окна визуализации соответственно. Классы **DisparityMap** и **Triangulation** управляют данными алгоритма построения карты диспарантности и метод триангуляции соответственно. Класс **StereoImage** хранит стереопару. Класс **StereoParametres** содержит параметры камер, параметры методов цифровой обработки изображений. Класс **IOData** отвечает за сохранение и загрузку данных. Класс **StereoProcessing** содержит в себе реализацию методов цифровой обработки изображений. **PGRCamera** отвечает за подключение и передачу данных камер Point Grey. Класс **Camera3D** отвечает за подключение камер, передачу стереопары и конвертацию данных из **ImageData** в **StereoImage**. Класс **ImageData** является промежуточным классом данных, служащим для работы с данными видеопотока камеры Point Grey.

4.3 Методы цифровой обработки изображений

После построения модели фона, вычитаем модель фона из исходного изображения методом ***cv::absdiff,***

void absdiff(InputArray src1, InputArray src2, OutputArray dst)

Далее размываем изображение простым размытием с параметром *ksize*, заданным в программе (по умолчанию, *cv::Size(10, 10)*), методом ***cv::blur***,

void blur(InputArray src, OutputArray dst, Size ksize, Point anchor=Point(-1,-1), int borderType=BORDER_DEFAULT)

проводим бинаризацию изображения по порогу методом ***cv::threshold***,

double threshold(InputArray src, OutputArray dst, double thresh, double maxval, int type)

находим связные области с помощью метода ***cv::findContours***,

void findContours(InputOutputArray image, OutputArrayOfArrays contours, OutputArray hierarchy, int mode, int method, Point offset=Point())

и фильтруем их по площади, которую определяем методом ***cv::contourArea***.

double contourArea(InputArray contour, bool oriented=false)

Находим центры масс связных областей, прошедших фильтрацию следующим образом:

mu[i] = moments(goodcontours[i], false);

mc[i] = Point2f(mu[i].m10 / mu[i].m00 , mu[i].m01 / mu[i].m00);

Где *goodcontours* – связные области, прошедшие фильтрацию, *mu[i]* – момент i-й связной области, *mc[i]* – центр i-й связанной области, который мы принимаем за двухмерные координаты i-го объекта.

Устраняем оптические искажения методом ***cv::undistortPoints***.

void undistortPoints(InputArray src, OutputArray dst, InputArray cameraMatrix, InputArray distCoeffs, InputArray R=noArray(), InputArray P=noArray())

Метод триангуляции выполнен операциями над ***cv::Mat***.

У нас есть параметры *A1, A2, p1, p2, R1, t1, R2, t2* типа *cv::Mat*, где:

- *A1, A2* - Внутренние параметры левой и правой камер соответственно.
- *p1, p2* - Двухмерные координаты объекта для левой и правой камер соответственно.
- *R1, t1* - Внешние параметры левой камеры.
- *R2, t2* - Внешние параметры правой камеры.

Вычисляем *Ainv1*, *Ainv2*, где *Ainv1* и *Ainv2* - обратные матрицы *A1* и *A2* соответственно.

```
cv::Mat Ainv1, Ainv2;
cv::invert(A1, Ainv1);
cv::invert(A2, Ainv2);
```

Вычисляем *R* и *t*:

```
cv::Mat R = R2 * R1.t();
```

```
cv::Mat t = -R * t1 + t2;
```

Вычисляем матрицу $\begin{pmatrix} {p_l}^T \left({A_l}^{-1}\right)^T {A_l}^{-1} p_l & -{p_l}^T \left({A_l}^{-1}\right)^T R^T {A_r}^{-1} p_r \\ -{p_l}^T \left({A_l}^{-1}\right)^T R {A_r}^{-1} p_r & {p_r}^T \left({A_r}^{-1}\right)^T {A_r}^{-1} p_r \end{pmatrix}$:

```
cv::Mat F = cv::Mat(2,2,CV_64FC1,cv::Scalar::all(0));
F.at<double>(0,0) = cv::Mat(p1.t() * Ainv1.t() * Ainv1 * p1).at<double>(0,0);
F.at<double>(0,1) = cv::Mat(-p1.t() * Ainv1.t() * R.t() * Ainv2 * p2).at<double>(0,0);
F.at<double>(1,0) = cv::Mat(-p1.t() * Ainv1.t() * R.t() * Ainv2 * p2).at<double>(0,0);
F.at<double>(1,1) = cv::Mat(p2.t() * Ainv2.t() * Ainv2 * p2).at<double>(0,0);
```

Вычисляем матрицу $\begin{pmatrix} {p_l}^T \left({A_l}^{-1}\right)^T {A_l}^{-1} p_l & -{p_l}^T \left({A_l}^{-1}\right)^T R^T {A_r}^{-1} p_r \\ -{p_l}^T \left({A_l}^{-1}\right)^T R {A_r}^{-1} p_r & {p_r}^T \left({A_r}^{-1}\right)^T {A_r}^{-1} p_r \end{pmatrix}^{-1}$:

```
cv::Mat Finv;
cv::invert(F,Finv);
```

Вычисляем матрицу $\begin{pmatrix} -{p_l}^T {A_l}^{-1} \\ p_r \left({A_r}^{-1}\right)^T \end{pmatrix}$:

```
cv::Mat J = cv::Mat(2,3,CV_64FC1,cv::Scalar::all(0));
cv::Mat J1 = -p1.t() * Ainv1.t() * R.t();
cv::Mat J2 = p2.t() * Ainv2.t();
for (int k = 0; k < 3; k++) {
    J.at<double>(0, k) = J1.at<double>(0, k);
    J.at<double>(1, k) = J2.at<double>(0, k);
}
```

Вычисляем матрицу $\begin{pmatrix} Z_l \\ Z_r \end{pmatrix}$:

```
cv::Mat Z = Finv * J * t;
cv::Mat pp1 = Z.at<double>(0,0) * Ainv1 * p1;
cv::Mat pp2 = Z.at<double>(1,0) * Ainv2 * p2;
```

Вычисляем координаты объекта в системах координат левой и правой камер:

```
cv::Mat Mresult1 = R1.t() * (pp1 - t1);
cv::Mat Mresult2 = R2.t() * (pp2 - t2);
```

Вычисляем точку, лежащую по центру между *Mresult1* и *Mresult2:*

```
cv::Mat Mresult = (Mresult1 + Mresult2) / 2;
```

Mresult - точка, которую мы принимаем за трехмерную координату объекта в глобальной системе координат.

4.4 Сохранение и загрузка данных

Система позволяет сохранять и загружать изображения и внутренние и внешние параметры камер. Изображения сохраняются и загружаются в формате JPEG. Внутренние и внешние параметры камер хранятся в виде:

<название> "<количество столбцов>, <количество строк>;<данные матрицы>".

Например:

cameraMatrix1 "3,3;1680.9,0,772.854;0,1672.09,614.316;0,0,1;".

4.5 Интерфейс

Интерфейс состоит из пяти блоков: меню, блок управления, основной блок, блок вывода данных, блок вывода стереоизображений.

Меню позволяет выбрать:

- стереоизображения отображать в программе;
- методы применять к стереоизображениям;
- методы применять к параметрам камер.

Также, из меню пользователь может закрыть программу.

Блок управления позволяет начать работу системы, начать захват видео и остановить захват видео.

Основной блок представляет собой панель из трех сегментов. Первый сегмент позволяет пользователю выбрать источники исходных данных, необходимых для работы системы, а именно: захват видеопотоков с камер системы, загрузка стереоизображения, загрузка параметров камер. Второй сегмент позволяет пользователю сохранить выбранное стереоизображение и параметры камер. Третий сегмент позволяет пользователю задать параметры метода триангуляции.

Блок вывода данных отображает данные о выполненных и невыполненных операциях.

Блок вывода изображений отображает выбранные пользователем стереоизображения с примененными к ним методами, также выбранными пользователем.

На рисунке 4.2 изображен скриншот работы программы.

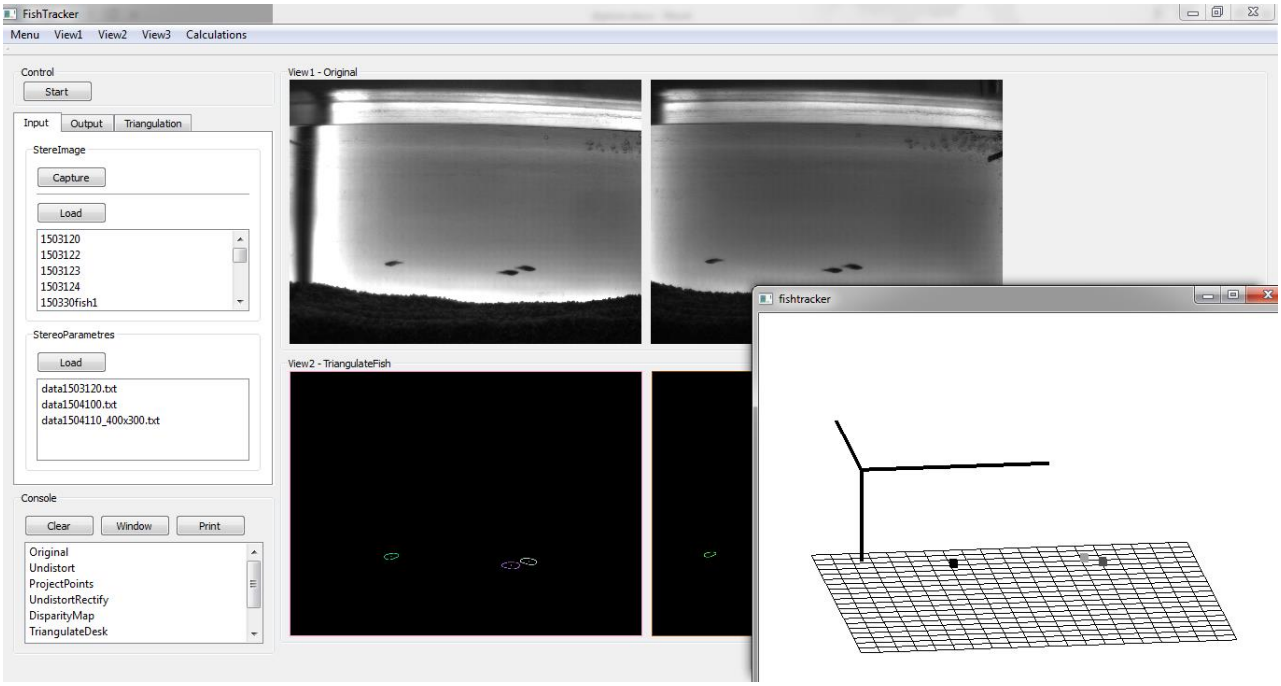

Рисунок 4.2. Скриншот работы программы

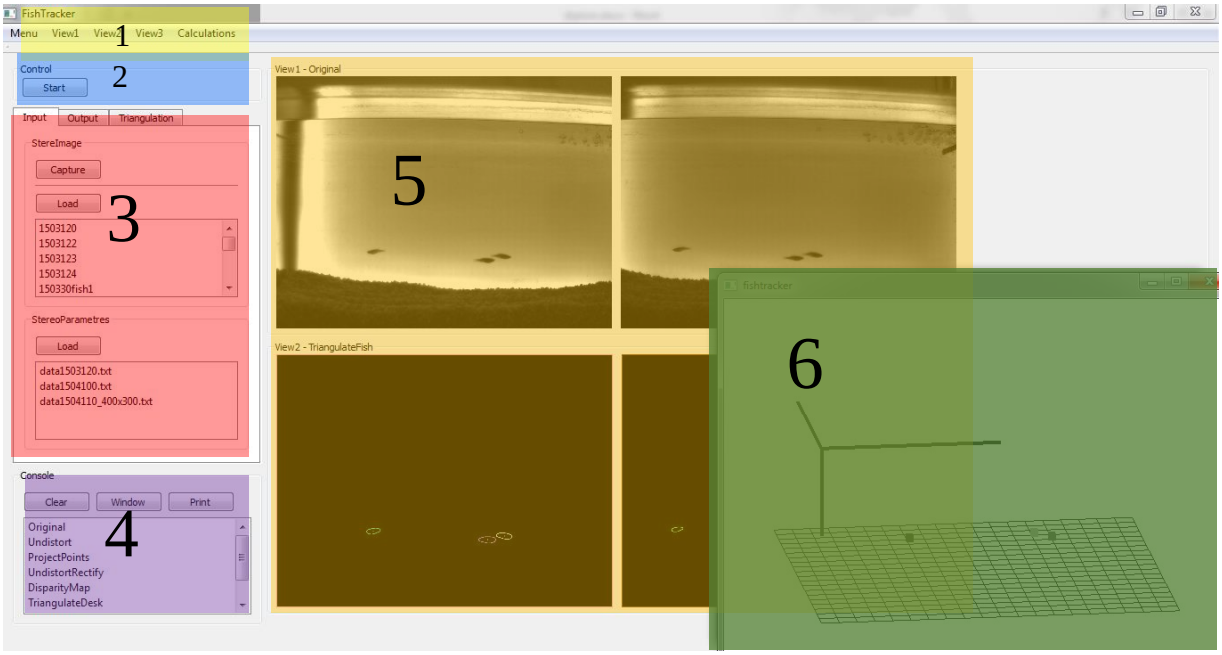

Рисунок 4.3. Скриншот работы программы с размеченными блоками. 1 - меню, 2 - блок управления, 3 - основной блок, 4 - блок вывода данных, 5 - блок вывода стереоизображений, 6 – окно визуализации

4.6 Визуализация данных

Для визуализации будем использовать инструментарий OpenGL, который позволяет визуализировать трехмерные объекты.

Результат триангуляции визуализируется программой в отдельном окне с помощью библиотеки OpenGL.

Пользователь может приближать, отдалять и поворачивать сцену.

Объекты отображаются в виде цветных кубиков, у каждой особи свой цвет. На рисунках 4.3 и 4.4 изображены кадры визуализации объектов.

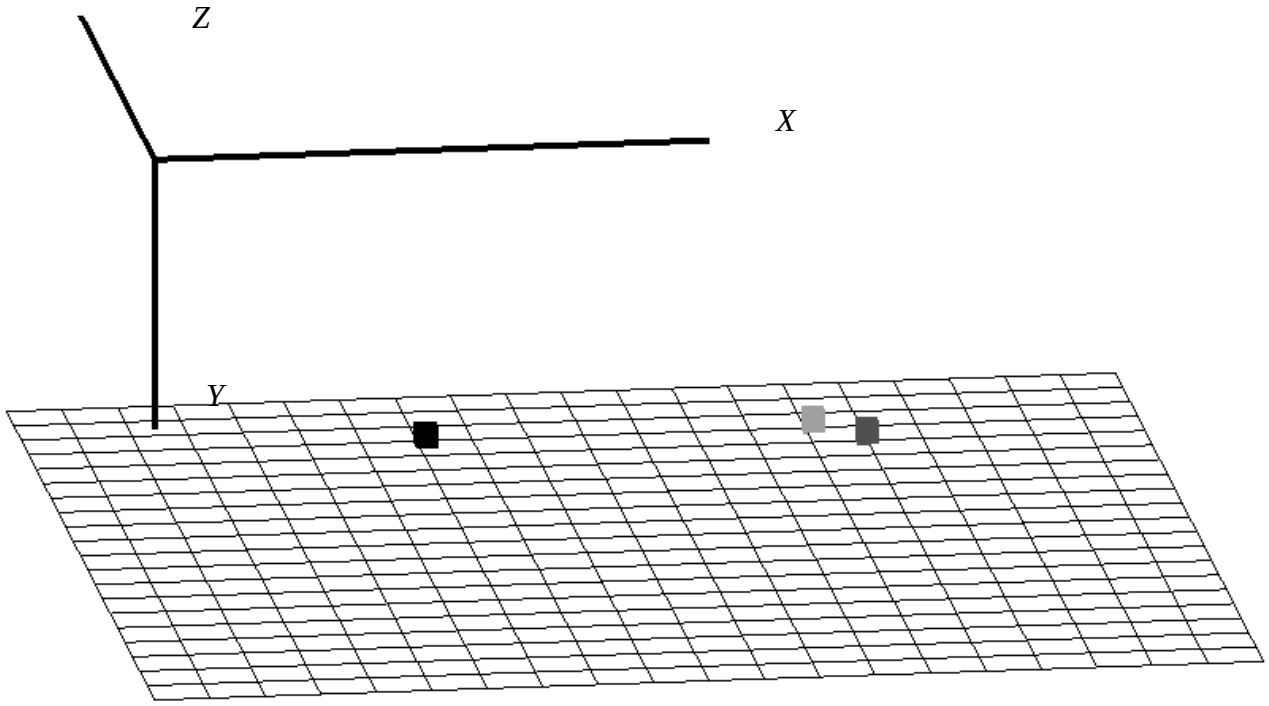

Рисунок 4.3. Кадр визуализации объектов

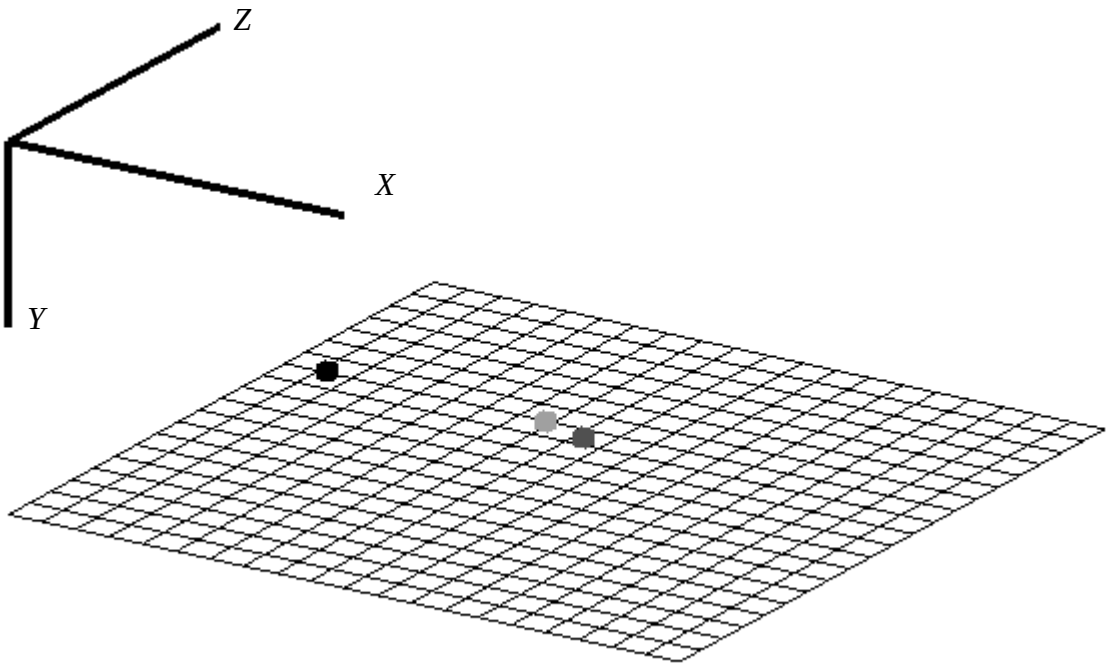

Рисунок 4.4. Кадр визуализации объектов

ГЛАВА 5. Результаты

Разработана система, позволяющая получать трехмерные координаты трех аквариумных рыб с точностью до 0.5 миллиметров и визуализирующая их в трехмерном пространстве. Частота работы алгоритма составляет 25 кадров в секунду на компьютере со следующими техническими характеристиками: операционная система Windows 7, процессор AMD Phenom II x6 1055T 2.8 ГГц, 8 ГБ ОЗУ, видеокарта NVIDIA GeForce GTS 450 с 4 ГБ памяти и 100 МБ свободного места на жестком диске.

ЗАКЛЮЧЕНИЕ

Для достижения цели проделана следующая работа: проанализирована проблема, выявлены требования к системе, разработана и сконструирована аппаратная часть системы, реализовано программное обеспечение, позволяющее получать трехмерные координаты объекта методом триангуляции, разработана визуализация перемещения лабораторных рыб в трехмерном пространстве. Работа получила диплом третьей степени на Международной научной студенческой конференции 2015 года.

В дальнейшем планируется оттестировать и отладить алгоритмы, внедрить систему в ЛИН СО РАН. Также, планируется реализовать разделение особей при их пересечении на кадрах с видеокамер.

ЛИТЕРАТУРА

1. Kulikov A. V., Tikhonova M. A., Kulikov V. A. Automated measurement of spatial preference in the open field test with transmitted lighting //Journal of neuroscience methods. – 2008. – Т. 170. – №. 2. – С. 345-351.
2. Куликов А. В., Куликов В. А. EthoStudio – новая компьютерная система измерения поведения // Российский Физиологический Журнал - 2004. – Т. 90. – №. 1. – С. 73-74.
3. Ribas L., Piferrer F. The zebrafish (Danio rerio) as a model organism, with emphasis on applications for finfish aquaculture research //Reviews in Aquaculture. – 2014. – Т. 6. – №. 4. – С. 209-240.
4. Qt Project [Электронный ресурс]. – Режим доступа: http://qt-project.org/wiki/ (дата обращения: 08.02.2015).
5. Шлее М. Е. Qt4. 8. Профессиональное программирование на C++ – БХВ-Петербург, 2012.
6. All about OpenGL ES 2.x [Электронный ресурс]. – Режим доступа: http://blog.db-in.com/ (дата обращения: 20.03.2015).
7. OpenGL. Официальное руководство программиста: Пер. с англ./Мейсон Ву, Джеки Нейдер, Том Девис, Дейв Шрайнер. – СПб: ООО «ДиаСофтЮП», 2002. - 592 с. – ISBN 5-93772-041-5.
8. Гонсалес Р., Вудс Р. Мир цифровой обработки Цифровая обработка изображений. – ТЕХНО-СФЕРА, 2005.
9. Шапиро Л. Компьютерное зрение/Шапиро Л., Стокман Дж.;[пер. с англ. АВ Назаренко, ИЮ Дорошенко] //М.: Бином. – 2006.
10. Learning OpenCV: Computer vision with the OpenCV library. / Bradski G., Kaehler A.: " O'Reilly Media, Inc.", 2008.
11. Косых, В.П. Цифровая обработка изображений: Учеб. пособие / Новосиб. гос. ун-т. - Новосибирск, 2006, 96.с.