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

1. Направление подготовки: 230100 Информатика и вычислительная техника

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ БАКАЛАВРСКАЯ РАБОТА**

«РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЧЕСКОГО ОТСЛЕЖИВАНИЯ ПЕРЕМЕЩЕНИЯ

ГРУППЫ ЛАБОРАТОРНЫХ РЫБ»

Липкин Евгений Олегович

**«К защите допущен» Научный руководитель**

Заведующий кафедрой, н.с. ИАиЭ СО РАН,

д.т.н. к.т.н.

Потатуркин О.И./………….. Куликов В.А./………...

(фамилия , И., О.) / (подпись, МП) (фамилия , И., О.) / (подпись, МП)

«……»………………2015г. «……»………………2015г.

Дата защиты: «……» ………………2015г.

Автор Липкин Е.О./………...

(фамилия, И., О.) / (подпись)

Новосибирск, 2015г.

Оглавление

[ВВЕДЕНИЕ 3](#_Toc421071665)

[ГЛАВА 1. Анализ задачи, требования к системе 4](#_Toc421071666)

[1.1 Анализ задачи 4](#_Toc421071667)

[1.2 Требования к системе 8](#_Toc421071668)

[1.2.1 Требования к аппаратной части системы 8](#_Toc421071669)

[1.2.2 Требования к программной части системы 8](#_Toc421071670)

[1.2.3 Сценарии работы программного обеспечения 9](#_Toc421071671)

[1.3 Программные средства 9](#_Toc421071672)

[ГЛАВА 2. Аппаратная часть системы 10](#_Toc421071673)

[2.1 Комплектующие 10](#_Toc421071674)

[2.2 Стойка для камер 11](#_Toc421071675)

[2.3 Система освещения 11](#_Toc421071676)

[ГЛАВА 3. Методы цифровой обработки изображения 12](#_Toc421071677)

[3.1 Построение фона 12](#_Toc421071678)

[3.2 Простое размытие 13](#_Toc421071679)

[3.3 Пороговая бинаризация 14](#_Toc421071680)

[3.4 Поиск связных областей 14](#_Toc421071681)

[3.5 Фильтрация связных областей по площади 14](#_Toc421071682)

[3.6 Поиск центров масс связных областей 15](#_Toc421071683)

[3.7 Андисторсия центров масс связных областей 15](#_Toc421071684)

[3.8 Получение трёхмерных координат объектов 16](#_Toc421071685)

[ГЛАВА 4. Реализация программной части системы 18](#_Toc421071686)

[4.1 Сценарий пользования 18](#_Toc421071687)

[4.2 Архитектура 18](#_Toc421071688)

[4.3 Методы цифровой обработки изображений 19](#_Toc421071689)

[4.4 Сохранение и загрузка данных 22](#_Toc421071690)

[4.5 Интерфейс 22](#_Toc421071691)

[4.6 Визуализация данных 23](#_Toc421071692)

[ГЛАВА 5. Результаты 24](#_Toc421071693)

[ЗАКЛЮЧЕНИЕ 24](#_Toc421071694)

[ЛИТЕРАТУРА 24](#_Toc421071695)

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, 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 рыбами.

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

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

Рыбы двигаются в трёхмерном пространстве, поэтому требуется реализовать инструментарий получения трёхмерных координат объектов. Для решения задачи получения трёхмерных координат объекта используют 3D камеры, которые представляют собой пару камер, расположенных на фиксированном расстоянии друг от друга, для получения стереопары (пары изображений с 3D камеры), с помощью которой определяют координаты, используя алгоритм построения карты диспарантности или метод триангуляции.

Цель дипломной работы состоит в разработке инструментария получения трёхмерных 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 трёхмерные координаты рыб;
* Разработать визуализацию перемещения рыб в трёхмерном пространстве.

В работе был создан программный модуль, определяющий трёхмерные координаты объектов методом триангуляции и позволяющий пользователю задавать параметры фильтра размытия по Гаусса, количество кадров и частота кадров для построения модели фона и границы площадей связных компонент для фильтрации случайных выбросов. Этот программный модуль позволяет системе EthoStudio автоматизировать этологические исследования с аквариумными 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 четвёртой 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. Также, существуют отдельные системы, заточенные под решение определенной задачи. Задача получения трёхмерных 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, а система «ZebraLab3D» позволяет получать трёхмерные координаты одной единственной аквариумной рыбы.



Рисунок 1: Скриншот программы ZEBRALAB

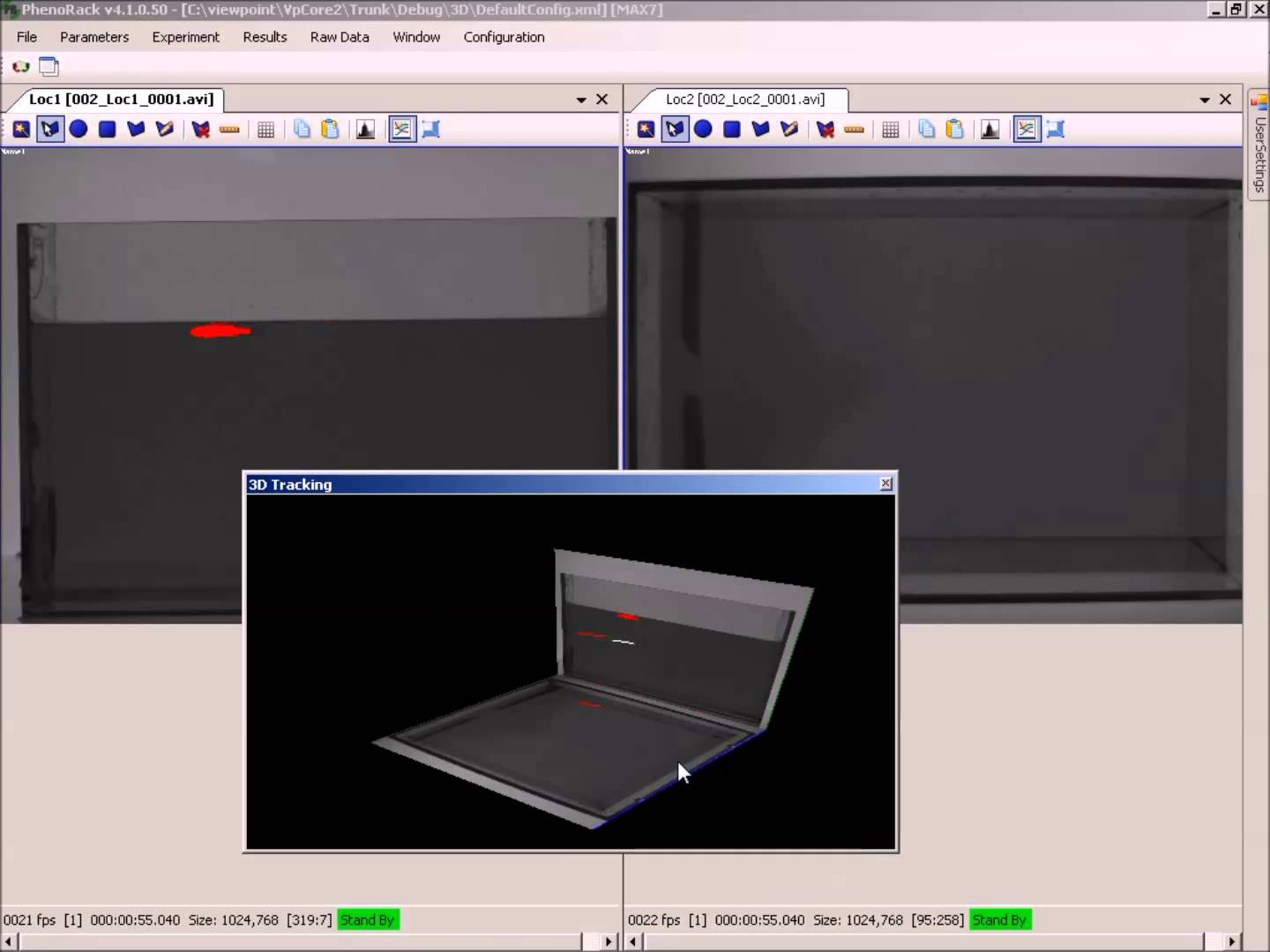


Рисунок 2: Скриншот программы ZEBRALAB3D

Для задач этологических исследований используют модельный организм биологии развития – вид данио-рерио(zebrafish). Её геном на 80% совпадает с человеческим. Благодаря тому, что биологи очень хорошо знают их внутреннее строение, этих рыбок многие биологи предпочитают использовать 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 данной задачи подойдут взрослые особи вида данио-рерио.

Биологи ЛИН СО РАН проводят исследования акустической чувствительности лабораторных рыб в аквариумах. Распространённым решением задачи получения трёхмерных 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. Нам потребуется пара камер, расположенных на фиксированном расстоянии друг от друга.

Датчики глубины (в частности, Microsoft Kinect) не подходят для решения задачи по причине искажений инфракрасной сетки в среде аквариума.



Рисунок 3: Кадр с камеры глубины Microsoft Kinect



Рисунок 4: Шумы поверх кадра Microsoft Kinect

Задачу можно решить, используя алгоритм построения карты диспарантности или метод триангуляции. Так как решение задачи не нуждается в реконструкции трёхмерной структуры сцены, а нужно только определять трёхмерные координаты наблюдаемых объектов, то для решения задачи больше подойдёт метод триангуляции. Алгоритм построения карты диспарантности будет производить лишние операции, 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.



Рисунок 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 трёхмерных координат трёх взрослых особей вида данио-рерио и визуализацией их в трёхмерном пространстве. При пересечении наблюдаемых объектов на кадре, принимаем центр масс связной компоненты за их двухмерные координаты на изображении.

Получение трёхмерных координат объектов должно быть реализовано с помощью метода триангуляции. Система должна определять трёхмерные 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 трёхмерной 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:

* запускаем программу;
* запускаем алгоритм автоматического отслеживания перемещения;
* по истечению эксперимента останавливаем алгоритм отслеживания перемещения;
* выбираем визуализацию данных удобным нам образом;
* при надобности сохраняем полученный результат.

Другой сценарий работы алгоритма заключается в следующем:

* загружаем данные о перемещении группы лабораторных рыб;
* визуализируем удобным нам образом.

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++.
* **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 использования для 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 трёхмерную компьютерную графику.
* **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 мм3;
* Фильтр, компрессор, терморегулятор;
* Система освещения;
* Стойка для камер;
* Две камеры Point Grey FL3-U3-32S2 M-CS с кадровой частотой – 60 кадров в секунду и разрешаемой способностью – 1600x1200 пикселей;
* Компьютер.



Рисунок 6, 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 твёрдого ПВХ 700x400x100 мм3. Передняя панель системы освещения состоит из акрилового оргстекла. Весь корпус, не включая переднюю панель, обклеен изнутри фольгой. Поверх фольги по контуру корпуса изнутри размещена светодиодная сетка. Система освещения 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 берётся видеопоток. Для каждого видеопотока выполняется следующий алгоритм: находятся объекты и определяются двухмерные координаты этих объектов. Далее, методом триангуляции определяются трёхмерные координаты объектов и визуализируются в трёхмерном пространстве.



Рисунок 7, Алгоритм работы программы

3.1 Построение фона

Нам нужно выделить наблюдаемые объекты, для этого нужно построить фон. Фон строится следующим образом: Для каждой точки фона мы вычисляем медианный элемент последовательности из 50 кадров с частотой 5 кадров в секунду:

,

где *t* – номер кадра, *x* и *y* – координаты точки фона, *I* –точка фона *t*-того кадра с координатами *x* и *y*, – точка фона с координатами *x* и *y*, а *median* – функция вычисления медианного элемента.

Вычитая фон из исходного изображения, мы получаем изображение наблюдаемых объектов.



Рисунок 8, Исходное изображение с видеокамеры



Рисунок 9, Изображение наблюдаемых объектов

3.2 Простое размытие

Следующий этап алгоритма – получение двухмерных координат объектов. В качестве координат объектов мы используем центры масс связных компонент. Для того, чтобы получить центры масс связных компонент, нам нужно размыть изображения и провести бинаризацию изображения наблюдаемых объектов.

Размытие изображения является простым размытием, которое представляет собой суммирование области *(w \* h)*. с последующим масштабированием на *1 / (w \* h)*. Каждый пиксель на выходе является средним арифметическим пикселей в области *(w \* h)*, где *w* и *h* – параметры простого размытия.

3.3 Пороговая бинаризация

Проводим бинаризацию изображения наблюдаемых объектов:

,

где *x* и *y* – координаты точки фона, *I(x, y)* –точка текущего кадра с координатами *x* и *y*, – точка бинаризованного изображения с координатами *x* и *y*, а *m* – порог бинаризации.



Рисунок 10, Бинаризованное изображение наблюдаемых объектов

3.4 Поиск связных областей

На бинаризованном изображении нужно провести поиск связных компонент.

,

где k – номер связанной области.

3.5 Фильтрация связных областей по площади

Фильтруем связные компоненты по размерам, чтобы отсечь случайные выбросы:

,

где – множество связных областей; – множество связных областей, прошедших фильтрацию; – площадь связной области ; и – границы наименьшей и наибольшей площадей соответственно.

3.6 Поиск центров масс связных областей

Центры масс связных компонент, прошедших фильтрацию, мы принимаем за двухмерные координаты объектов.

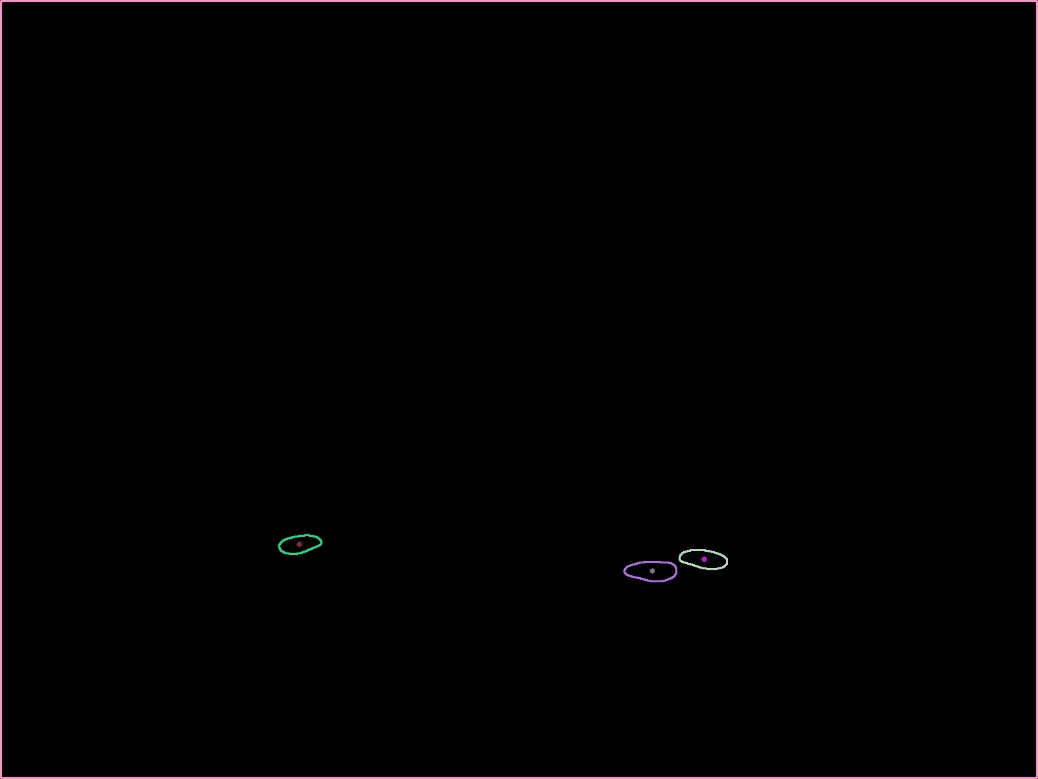


Рисунок 11, Связные компоненты и их центры масс

3.7 Андисторсия центров масс связных областей

Для учета искажения камеры, мы должны провести андисторсию двухмерных координат объектов.

,

где *x* и *y* – координаты точки, а *undistort* – функция.

3.8 Получение трёхмерных координат объектов

Зная параметры камер и двухмерные координаты объектов, мы можем решить задачу триангуляции, которая сводится к решению системы линейных уравнений методом наименьших квадратов.

Имеем пару камер. К каждой камере известны внутренние , и внешние параметры , , , .

К каждой камере известны внешние параметры R\_l R\_r t\_l t\_(r ).

Тогда системы камер будут иметь свои параметры перехода

,

,

где *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.

Ошибка. – вектор невязки, обусловленный наличием ошибок измерений.

Метод наименьших квадратов:

Оценим и методом наименьших квадратов, подставив следующие величины:

Получаем:

В глобальной системе координат:

За трёхмерную координату объекта мы принимаем точку, лежащую по центру между и :

.

ГЛАВА 4. Реализация программной части системы

4.1 Сценарий пользования

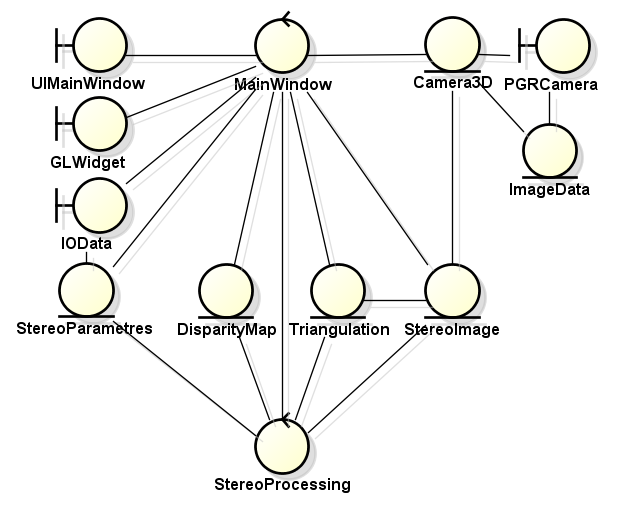
После запуска программы для работы пользователю необходимо выбрать данные, с которыми он будет работать. Нужно выбрать захват видеопотоков с камер или загрузить стереоизображение. Далее нужно выбрать файл внутренних и внешних параметров камер. Теперь пользователь может нажать кнопку «Start», и при успешном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

Пользователь может:

* установить параметры алгоритма построения карты диспарантности и метода триангуляции;
* произвести выбранную операцию над параметрами камер;
* отобразить не более трёх стереоизображений с выбранными методами обработки;
* сохранить стереоизображение и параметры камер.

4.2 Архитектура

Программная часть системы имеет следующую архитектуру:



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

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), intborderType=BORDER\_DEFAULT )***

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

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

находим связные компоненты 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 компонент, 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.

Вычисляем андисторсию 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 :

*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 :

*cv::Mat Finv;*

*cv::invert(F,Finv);*

Вычисляем матрицу :

*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 :

*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 трёхмерную 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:

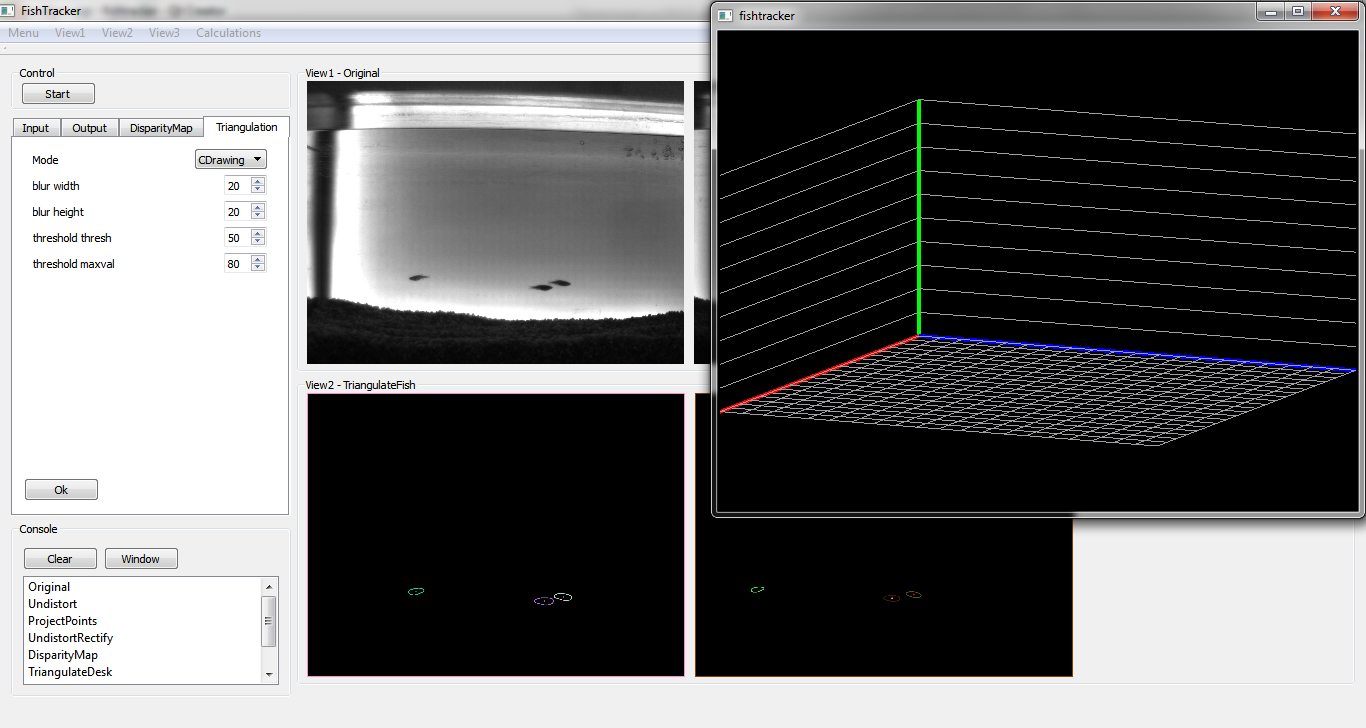
* выбрать, какие стереоизображения отображать в программе;
* выбрать, какие методы применять к стереоизображениям;
* выбрать, какие методы применять к параметрам камер;
* закрыть программу.

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

Основной блок представляет собой панель из четырёх 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 и четвёртый сегменты позволяют пользователю задать параметры алгоритма построения карты диспарантности и метода триангуляции соответственно.

Блок вывода данных отображает данные о выполненных и невыполненных операциях.

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



4.6 Визуализация данных

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

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

Пользователь может приближать, отдалять и поворачивать сцену.

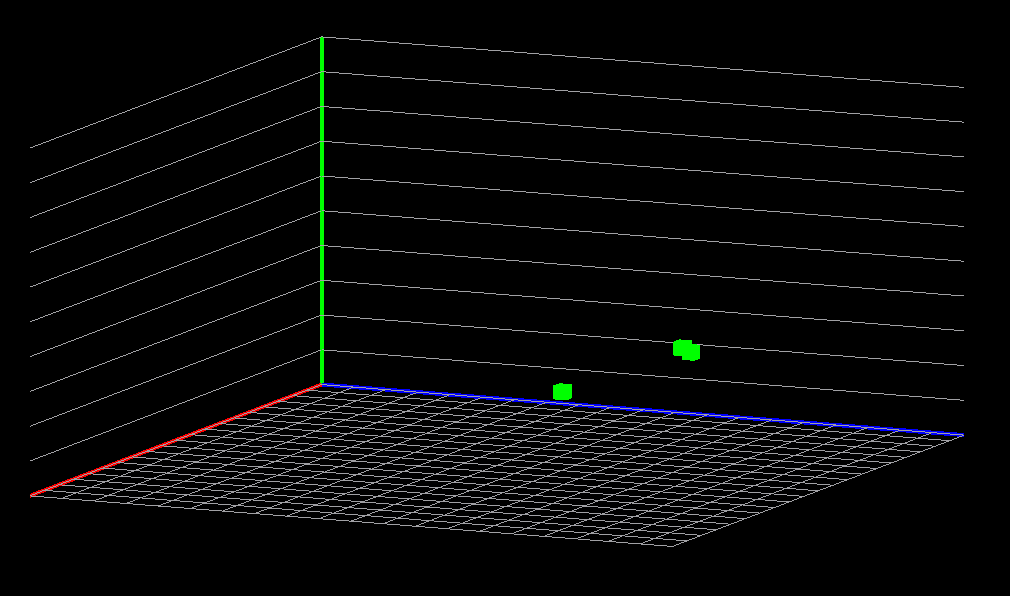


Рисунок 13, Кадр визуализации объектов

ГЛАВА 5. Результаты

Получены следующие результаты.

Разработана система.

ЗАКЛЮЧЕНИЕ

Для достижения цели проделана следующая работа: проанализирована проблема, выявлены требования к системе, разработана и сконструирована аппаратная часть системы, реализовано программное обеспечение, позволяющее получать трёхмерные координаты объекта методом триангуляции, разработана визуализация перемещения рыб в трёхмерном 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.

ЛИТЕРАТУРА

1. Gary Bradasky, Adrian Kaehler, Learning OpenCV: Computer Vision using the OpenCV Library
2. Richard Hartley, Andrew Zisserman, Multiple View Geometry in Computer Vision
3. Документация библиотеки OpenCV
4. Документация Point Grey FlyCapture 2.0 SDK