Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский государственный университет»

Кафедра Системного Программирования

Попов Кирилл Владимирович

Одновременная навигация и составление карты с использованием монокулярного зрения на основе рекуррентной фильтрации

Дипломная работа

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., профессор Терехов А. Н.

Научный руководитель:
к. ф.-м. н., доцент Вахитов А. Т.

Рецензент:
аспирант Кузнецов К. О.

Санкт-Петербург
2015

SAINT-PETERSBURG STATE UNIVERSITY

Chair of Software Engineering

Kirill Popov

Monocular simultaneous location and mapping based on recursive filtering

Graduation Thesis

Admitted for defence.
Head of the chair:
professor Andrey Terekhov

Scientific supervisor:
associate professor Alexander Vakhitov

Reviewer:
assistant Kirill Kuznetsov

Saint-Petersburg
2015

Оглавление

Введение

Не смотря на повсеместное использование систем глобального позиционирования проблема определения собственного местоположения по-прежнему стоит достаточно остро. С одной стороны, есть места куда не доходит сигнал спутников, либо он приходит сильно искажённым. С другой, иногда возникает потребность определить местоположение, скорость и траекторию движения объекта, имея лишь запись видео с его камеры. Отличным примером служат записи видео с автомобильных регистраторов.

В то же время существует необходимость в построении карт местности. Это поможет роботам лучше ориентироваться в пространстве, а людям, используя огромное количество видео, находящего в отрытом доступе в интернете, автоматически создавать детальные 3D карты местности.

Проблемы определения собственного местоположения и построения карты местности очень сильно связаны между собой. Нельзя построить карту местности, не зная собственных координат и маршрута. С другой стороны, не получится построить точный маршрут, не обладая информации о том, где ты находишься. В некотором смысле это очень похоже на известную проблему "яйца и курицы". Поэтому имеет смысл решать эти две задачи одновременно.

От сюда берет своё начало отдельное направление в робототехнике и компьютерном зрении SLAM[7] (рус. Одновременная навигация и составление карты)

Так как алгоритмы построения карты должны работать в реальном времени, то, учитывая их вычислительную сложность и размеры матриц камер, приходится снижать размерность входных данных. Поэтому вместо отслеживания каждого пиксели на экране, на каждом кадре ищутся "особые точки"[5] и считается их "оптический поток"[8]. То есть определяется траектория их движения на видео.

В своей работе я реализовал открытую и расширяемую 3D модель, которая позволяет задавать параметры камеры робота и траекторию

его движения. Кроме этого в модели задаются координаты объектов в окружающем робота мире и параметры шума, связанные, к примеру, с несовершенством приборов. На выходе получаются данные о том, как робот "видит" окружающий мир и то, какие данные будут переданных на вход алгоритму SLAM.

В качестве алгоритма был выбран "Монокулярный SLAM с обратной параметризацией глубины"[2] (далее просто monoSLAM). Самими авторами данный алгоритм был реализован в виде "proof-of-concept" в тесном переплетении с другими алгоритмами. Это не позволяло оценить точность и границы применения их решения. Поэтому мной была реализована своя версия этого алгоритма с использованием модульной архитектуры. В дальнейшем это позволит со значительно меньшими трудозатратами изменять и дорабатывать алгоритм, а также с минимальными усилиями портировать его для встраиваемых систем, разрабатываемых, в том числе, и на нашей кафедре.

Так же мной был проведён анализ точности определения местоположения и построения карты при различных параметрах камеры, шума и начальных значениях дисперсии, которыми инициализируется monoSLAM. А так же проверена его работоспособность на реальных данных.

1. Постановка задачи

Целью дипломной работы является создание библиотеки геометрической реконструкции на базе рекуррентной фильтрации для монокулярных видео:

- Создать программные средства моделирования сцены и робота
- Разработать библиотеку геометрической реконструкции
- Провести испытания на синтетических и реальных данных

2. Обзор

В данном разделе представлена информация о различных подходах к одновременной навигации и составлению карты

2.1. SLAM с применением датчика глубины

При таком подходе особые точки, которые видит робот, характеризуются тремя числами u, v, d, где u, v - координаты особой точки на экране и d - дальность, измеренная с помощью различных устройств. Данный подход позволяет добиться наилучшего качества. Однако, минусы состоят в том, что необходимо устанавливать дополнительный сенсор и учитывать его шум. Так же отпадает возможность использовать огромную коллекцию видео находящегося в открытом доступе.

2.2. SLAM с использованием бинокулярного зрения

В данном методе используется триангуляция. Зная собственные матрицы камер и расстояние между ними можно определить дальность до особой точки. Причём максимальная дальность зависит от расстояния между камерами, которое в таких системах фиксировано. Что накладывает свой отпечаток на границы применения алгоритмов, использующих данный подход. При таком подходе по прежнему не получится использовать видео из интернета.

2.3. SLAM с использованием монокулярного зрения

Ключевой особенностью этого решения является то, что все что вам нужно для построения карты - это одна камера. Для определения дальности до особых точек все так же используется метод триангуляции, но вместо 2-х камер, используется текущий кадр и тот, на котором особая точка была обнаружена. Это позволяет определять дальность даже до тех точек, которые находятся на значительном удалении от камеры, в

случае, если в процессе движения будут получены кадры с достаточным углом параллакса.

3. Математическая модель сцены и данных с камеры робота

Для тестирования правильности реализации алгоритма monoSLAM, а так же для подборки оптимальных коэффициентов и исследования границ его применимости была реализована математическая модель окружающего мира.

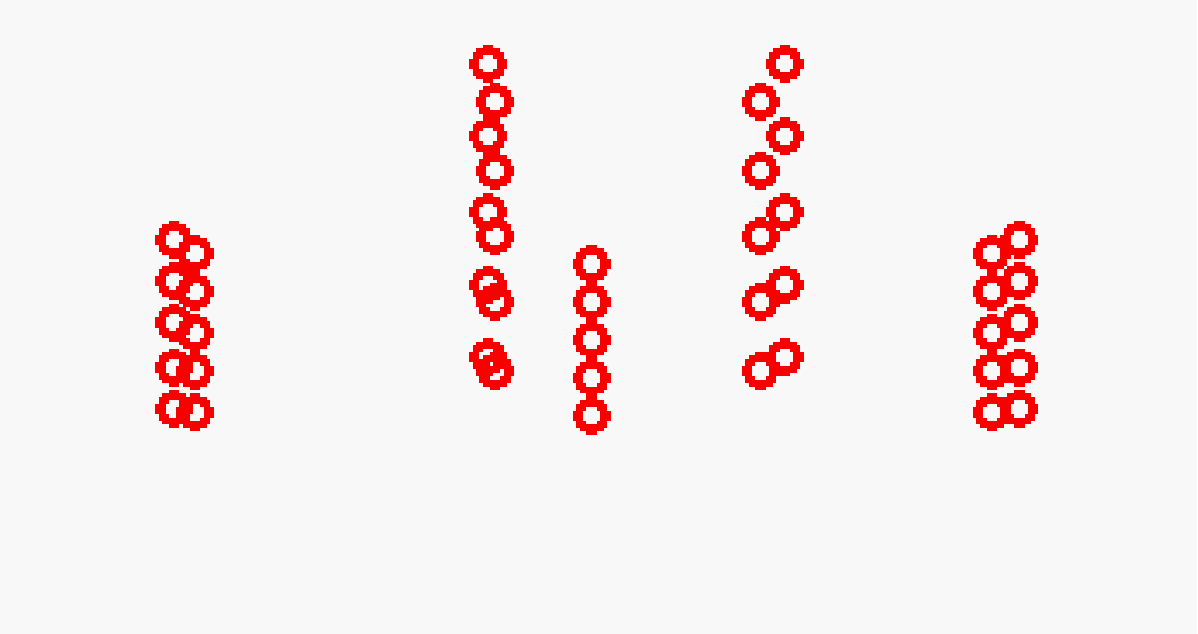

Рис. 1: Один кадр сгенерированный математической моделью. Чёрным обозначены особые точки которые видит робот

Для работы модели пользователь должен задать расположение особых точек в трёхмерном пространстве, собственную матрицу камеры, шум и траекторию движения робота. На выходе модель выдаст (u, v) - координаты точек на экране для каждого кадра, а так же их номера, чтобы робот мог отслеживать траектории их перемещения.

Разберёмся подробнее как работает математическая модель. Собственная матрица камеры описывается матрицей

$$K = \begin{pmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

где f - фокусное расстояние камеры, а (c_x, c_y) - координаты центра экрана. Положение камеры в пространстве описывается вектором

$$T = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

и матрицей поворота

$$R = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}$$

Таким образом для каждой особой точки X, которую задаёт пользователь, мы можем получить её изображение на экране камеры $x = (u, v, 1)^T$

$$x = K * [R|T] * X$$

далее, если включена соответствующая настройка к x подмешивается нормальный шум с заданной дисперсией $\mu \in \mathcal{N}(0, \sigma)$

$$x = x + \mu$$

Необходимость моделировать шум вызвана несовершенством камер и способов определения их внутренних параметров.

3.1. Задание траектории движения

Отдельно стоит остановиться на вычислении матрицы R. В трёхмерном пространстве существует насколько способов задать вращение камеры[6]. С помощью углов Эйлера, матрицы вращения, вектора и угла, кватернионов и матрицы "look-at".

Для задания ориентации робота были опробованы все эти способы с целью выбора наиболее удобного для дальнейшего использования.

Углы Эйлера. Несмотря на то, что с их помощью можно интуитивно описывать вращение с помощью угла, крена и тангажа. Однако возни-

кает проблема "шарнирного замка", а так же то, что в случае интерполяции, вращение происходит не по оптимальной траектории.

Матрица вращения. Главный их недостаток для описываемого решения заключается в том, что для задания матрицы нужно указать 9 чисел, что во-первых не удобно, а, во-вторых, контринтуитивно.

Вектор вращения и угол. Обладает теми же проблемами, что и матрица вращения, несмотря на то, что нужно задать всего лишь 4 числа.

Кватернионы Обладает теми же недостатками как и вектор вращение, более того, для задания ориентации робота с помощью кватернионов требуется провести дополнительные вычисления.

Look-at Данный способ был выбран в качестве основного по следующим причинам. Для задания ориентации камеры достаточно указать точки X_k, Y_k, Z_k, в которые будет направлена камера и интерполировать её поворот между ними. Так же данный метод интуитивно понятен и не требует использования дополнительных расчёт для задания ориентации робота по маршруту его движения.

4. Разработка библиотеки геометрической реконструкции

Для удобства проведения экспериментов, а так же дальнейшего портирования алгоритма monoSLAM[2], для использования в роботах, была разработана собственная библиотека. Она позволяет задавать в качестве источника данных как запись треков собранных роботом, так и обрабатывать их в реальном времени.

4.1. Входные данные и параметры алгоритма

Для описания треков был разработан собственный формат, который можно представить как 3d - массив. Каждая строчка состоит из 4-ки чисел

$$(N_{frame}, N_{feature}, N_{dim}, V)$$

где N_{frame} - номер кадра, $N_{feature}$ - номер особой точки, N_{dim} - номер экранной координаты (0 - u или 1 - v) и V - значение N_{dim} координаты для $N_{feature}$ особой точки на кадре N_{frame}.

Так же модель позволяет настраивать следующие параметры:

- K - собственная матрица камеры.
- X_{cam} - 3-х мерный вектор описывающий начальное положение камеры в мировой системе координат
- $X_{cam\_rotation}$ кватернион описывающий изначальный поворот камеры
- $X_{cam\_velocity}$ вектор описывающий начальное линейное ускорение камеры
- $X_{cam\_angular\_velocity}$ вектор описывающий начальную угловую скорость камеры
- $P_{velocity}$ начальная дисперсия вектора скорости

- $P_{angular}$ начальная дисперсия вектора угловой скорости
- Q матрица ковариации шума модели
- R матрица ковариации шума измерений

4.2. EKF

Основную роль в реализованном алгоритме играет EKF(eng. Extended Kalman filter)[4] - Расширенный фильтр Калмана. Он представляет из себя нелинейную версию фильтра Калмана. Основная идея заключается в том, что в случаях, когда используемая модель системы не линейна, мы можем аппроксимировать её линейной функцией испольуя разложение в ряд Тейлора.

4.3. MonoSLAM

В этом разделе приведён краткий обзор алгоритма monoSLAM. Подробное описание можно найти в оригинальной статье авторов на английском языке[2].

Алгоритм использует следующую модель реального мира(Рис. 2). Камера описывается вектором

$$X_{cam} = \begin{pmatrix} r \\ q \\ V_l \\ V_a \end{pmatrix}$$

где r - радиус вектор положения камеры в пространстве, q - кватернион, описывающий её ориентацию, а V_l и V_a векторы линейного и углового ускорения соответственно.

Каждая особая точке описывается вектором из 6 параметров

$$Y = \begin{pmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \\ \theta \\ \mu \\ p \end{pmatrix}$$

где x_{cam}, y_{cam}, z_{cam} координаты камеры в момент обнаружения особой точки, θ и μ - азимут и угол возвышения в абсолютной системе координат указывающие на особую точку, p - обратная дальность до особой точки. Как уже упоминалась ранее, такая параметризация позволяет

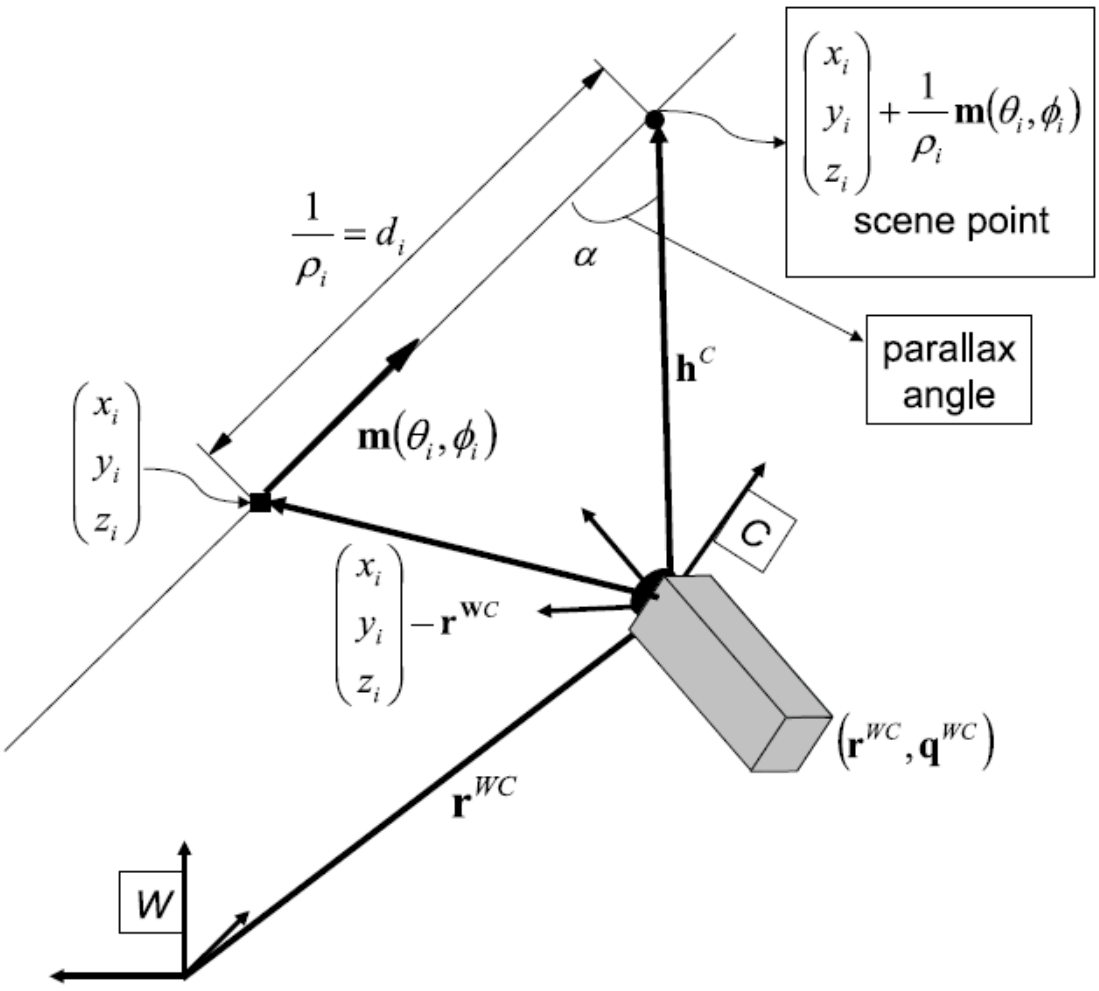

Рис. 2: Математическая модель используемая в алгоритме monoSLAM

описывать точки на бесконечности.

Все месте это образует вектор состояния системы в кадре k

$$X_k = (X_{cam}, Y_1, Y_2, ..., Y_N)$$

Ниже приведено краткое описание шагов алгоритма monoSLAM.

1. Создаётся список из особых точек видимых на текущем кадре. Формируется вектор состояния системы X и матрица ковариации P

2. Для каждой точки вычисляется её 6-и мерный вектор, основываясь на текущем положений камеры. Изначальная дальность берётся равной 1. Она будет корректироваться в процессе работы алгоритма. Так же для каждой особой точки вычисляется её матрица ковариации.

3. Вычисляется положение камеры, основываясь на известных данных о модели движения. Так же обновляется соответствующая часть матрицы ковариации P

4. На основании вычисленного на предыдущем шаге нового положения камеры предсказываются новые измерения особых точек.

5. Считывается новый кадр.

6. Производится стандартный этап коррекции фильтра Калмана. На основании новых измеренных, предсказанных значений, а так же матрицы ковариации измерений производится обновление состояния системы.

4.4. Особенности реализации

Авторы monoSLAM выложили в открытый доступ свою реализацию написанную на Matlab. Из-за того, что их реализация не соответствовала нашим требованиям, предъявляемым к алгоритмам, было принято

решение написать свою реализацию. Во-первых, Matlab является проприетарным языком, что не позволяет использовать написанные на нем алгоритмы не приобретая продукт. Во-вторых, в исходной реализации авторы использовали не только monoSLAM, но и алгоритмы для анализа изображений и поиска особых точек. Это не позволяло использовать данные из математической модели, а так же оценить эффективность самого алгоритма. Кроме того, авторы позиционировали свою программу как доказательство самой возможности реализации и не заостряли внимание на её архитектуре. В итоге полученный код достаточно тяжело портировать, в нем отсутствует модульность и расширяемость.

Для реализации алгоритма был выбран язык Python и библиотека для проведения математических расчётов Numpy[3]. Графики для анализа корректности работы алгоритмами оценки влияния начальных параметров на результат строились с помощью библиотеки Matplotlib[1].

Выбор языка Python обосновывается тем, что, во-первых, на нём не существует открытой реализации алгоритма monoSLAM и, во-вторых, он даёт большую гибкость чем Matlab в вопросах построения архитектуры.

Объем кода библиотеки геометрической реконструкции составляет примерно 500 строчек. Для алгоритма составления карты и навигации этот показатель превышает одну тысячу.

5. Эксперименты

Для визуальной проверки правильности работы модуля геометрической реконструкции был реализован механизм записи видео перемещения особых точек на экране. То есть сцена была визуализирована в том виде, в котором её видит робот в процессе своего движения. (См. Рис. 1)

Обращаю внимание на то, что точки изображены абсолютно одинаково, несмотря на то, что они находятся на разном удалении от камеры робота. Это сделано специально, т.к. робот не имеет никакой информации о физической природе точек, и следовательно не учитывает её при построении карты.

5.1. На модели сцены

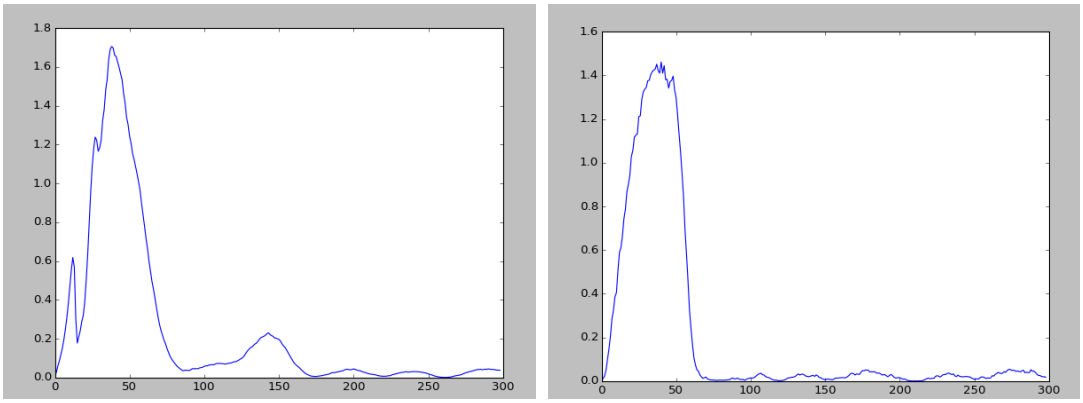

Рис. 3: Среднеквадратичное ошибка определения положения робота. Слева - не зашумлённые наблюдения. Справа - наблюдения с шумом $\sigma = 1$.

Для проверки правильности реализации алгоритма monoSLAM, было измерено средне квадратическое отклонение восстановленной роботом траектории, а так же построенной им карты местности от соответствующих идеальных значений заложенных в математической модели. Стоит заметить, что результаты работы расширенного фильтра Калмана очень сильно зависит от задания начальных параметров и, следовательно, при тестах были выбраны наиболее оптимальные значения. Из результатов (Рис. 3) видно, что даже при наличии шума, можно по-

добрать начальные параметры таким образом, чтобы восстановленный результат не уступал результату без шума.

На (Рис. 5) красным отмечены точки, соответствующие траектории движения робота заложенной в модели. Синим отмечена траектория движения робота восстановленная по результатам наблюдений за особыми точками. Видно, что несмотря на выбросы, вызванные шумом, восстановленные траектория и карта практически не отличаются от исходных.

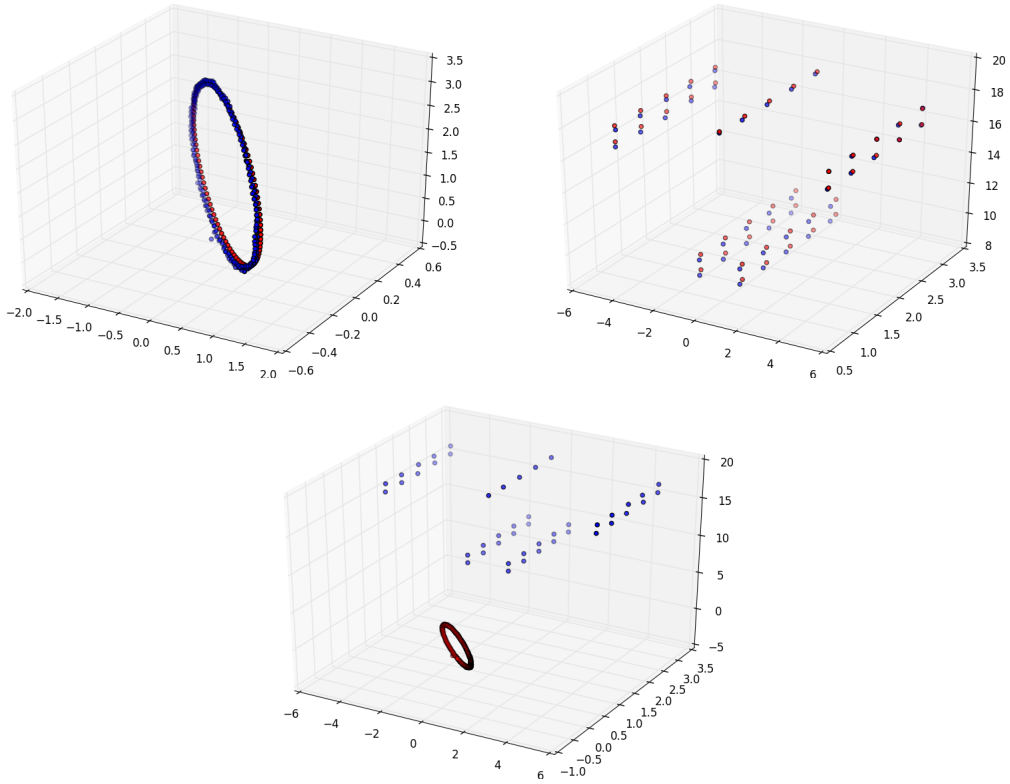

Рис. 4: Реконструкция пути робота и положения особых точек в пространстве на основе реальных данных. Слева на право: траектория движения робота, расположение особых точек на карте, общая карта включающая в себя оба графика

5.2. На реальных данных

Так же были проведены эксперименты на данных, полученных с камеры движущегося робота. Робот двигался по прямой траектории находясь в небольшом закрытом помещении(маленькой комнате). На графике(Рис. 5) видно, что по особым точкам была восстановлена форма той части комнаты, которую наблюдал робот. К сожалению, ввиду отсутствия точных приборов для измерения расстояния, было не возможно провести измерение расстояний до особых точек для оценки точности измерений.

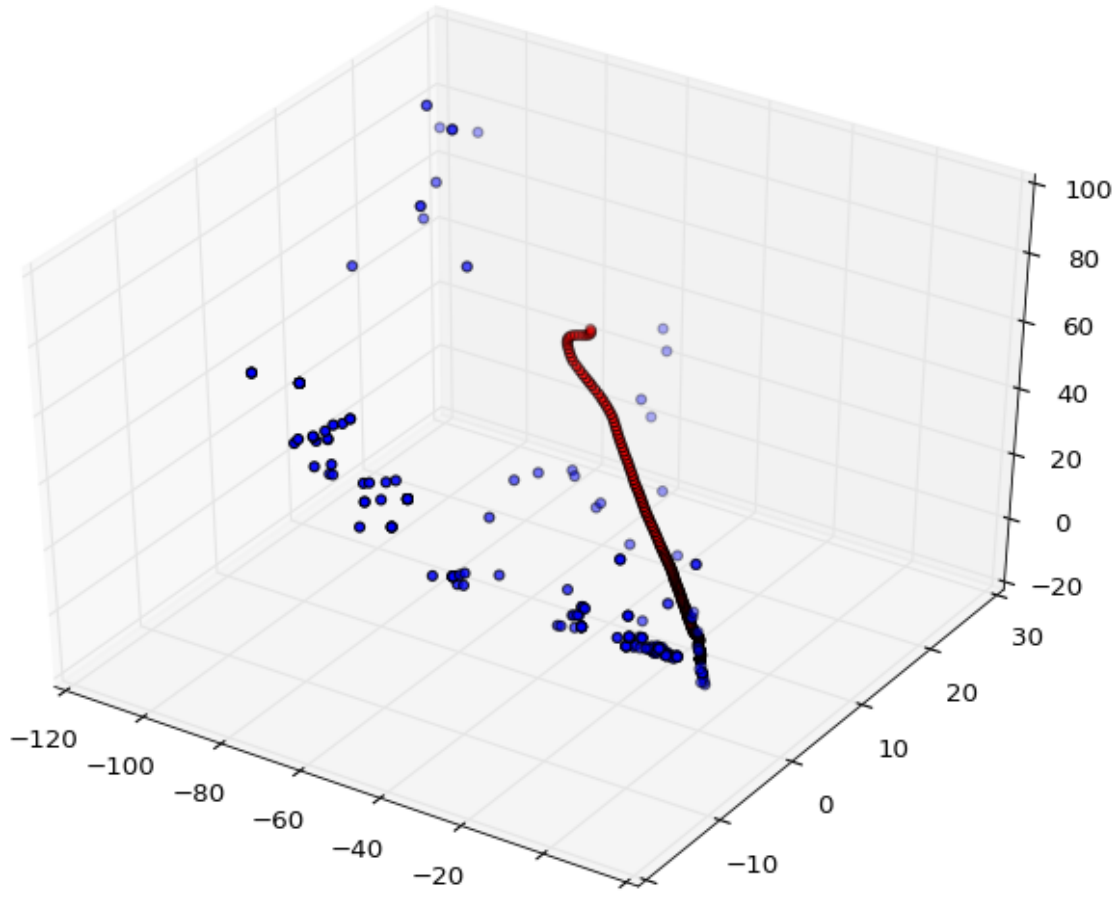

Рис. 5: Применение алгоритма на реальных данных. Красный - траектория движения робота. Синий - особые точки.

5.3. Ошибки перепроектирования

Для движения робота по модели сцены были посчитаны ошибки перепроектирования. Для каждого кадра полученного роботом, сравнивались актуальные координаты особых точек, наблюдаемых им и координаты, полученные в результате работе модели. Сравнения проводились при значениях Q и P равных 0.007.(Рис. 6)

Начальный пик, наблюдаемый на всех графиках, вызван начальными параметрами шума, заданными в фильтре Калмана. В дальнейшем фильтр сходится к оптимальному значению за фиксированное число итераций, не зависящее от уровня шума в исходных данных.

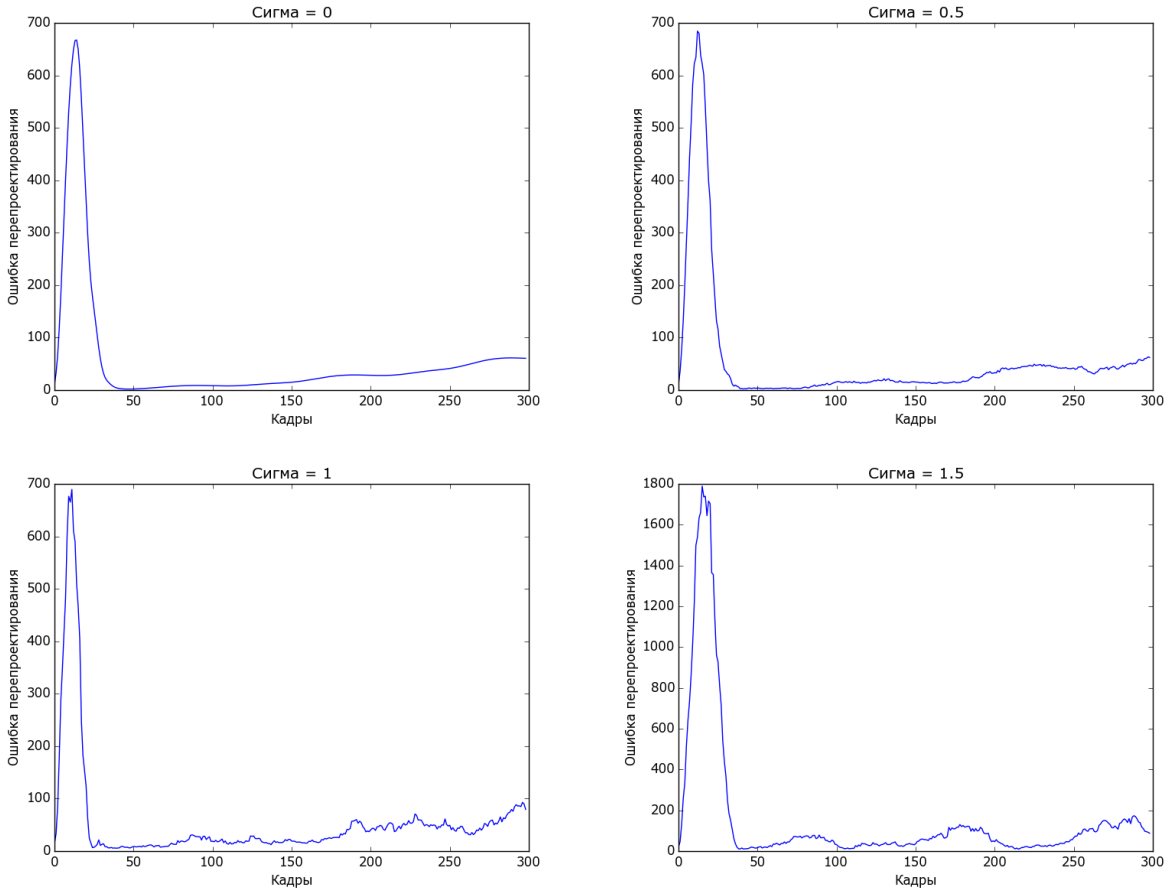

Рис. 6: Графики зависимости величины перепроектирования в зависимости от времени. По горизонтали - время в кадрах. По вертикали - величина перепроектирования.

Заключение

В результате проведённой работы были созданы программные средства моделирования сцены и траектории движения робота с учётом влияния шума. Так же была разработана библиотека геометрической реконструкции, которые позволяет применять алгоритм monoSLAM на реальных и тестовых данных, а так же упрощает внесение изменений и подбор параметров модели для различных условий.

Кроме того, были проведены испытания разработанной библиотеки на тестовых и реальных данных. Были получены результаты, которые говорят о корректной реализации алгоритма. Так же на основании полученных данных был сделан вывод о пригодности использования алгоритма monoSLAM в задаче определения местоположения, которые стоят перед роботами, разрабатываемыми на кафедре системного программирования. Результаты данной работы будут использованы в дальнейших исследованиях, направленных на выбор наиболее эффективного алгоритма для задачи определения местоположения и составления карты.

Список литературы

[1] Matplotlib. Официальный сайт Matplotlib. — 2015. — URL: http://matplotlib.org/ (online; accessed: 03.06.2015).

[2] Montiel J.M.M., Civera Javier, Davison Andrew J. Unified Inverse Depth Parametrization for Monocular SLAM.

[3] Numpy. Официальный сайт Numpy. — 2015. — URL: http://www.numpy.org/ (online; accessed: 03.06.2015).

[4] Wikipedia. Extended Kalman filter. — 2015. — URL: https://en.wikipedia.org/wiki/Extended_Kalman_filter (online; accessed: 06.06.2015).

[5] Wikipedia. Feature_(computer_vision). — 2015. — URL: http://en.wikipedia.org/wiki/Feature_(computer_vision) (online; accessed: 03.06.2015).

[6] Wikipedia. Rotation formalisms in three dimensions. — 2015. — URL: http://en.wikipedia.org/wiki/Rotation_formalisms_in_three_dimensions (online; accessed: 06.06.2015).

[7] Wikipedia. Simultaneous localization and mapping. — 2015. — URL: http://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping (online; accessed: 03.06.2015).

[8] Wikipedia. Оптический поток. — 2015. — URL: https://ru.wikipedia.org/wiki/SLAM_(метод) (дата обращения: 03.06.2015).