МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий

Кафедра «Прикладная математика и информатика»

02.03.03 Математическое обеспечение и администрирование информационных систем

Технология программирования

БАКАЛАВРСКАЯ РАБОТА

на тему Разработка сервера обработки данных спутниковых систем навигации с применением фильтра Калмана

Студент А.В. Додонов
(И.О. Фамилия) (личная подпись)

Руководитель В.С. Климов
(И.О. Фамилия) (личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н, доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____»_____20_____г.

Тольятти 2018

АННОТАЦИЯ

Тема выпускной квалификационной работы: Разработка сервера обработки данных спутниковых систем навигации с применением фильтра Калмана.

Целью ВКР является создание серверного приложения для обработки и фильтрации данных, полученных с помощью спутниковых систем навигации.

Дипломная работа состоит из следующих частей: введение, три главы, заключение, список используемой литературы и приложение.

Во введении описывается актуальность представленной темы, формируются задачи и общая цель, которую необходимо выполнить.

В первой главе описан принцип работы спутниковых систем навигации, проведён сравнительный анализ существующих разработок в сфере спутникового отслеживания, а также описывается проблема точности полученных геоданных.

Во второй главе разрабатывается математическая модель фильтра Калмана и рассчитываются итерационные формулы для применения в рамках фильтрации данных.

В третьей главе описывается разработка сервера обработки и фильтрации данных, создание базы данных для хранения информации, реализация программного кода фильтрации данных и проведён анализ результатов.

В работе представлено 18 рисунков, 1 таблица, 14 формул, список использованной литературы насчитывает 21 источник. Общий объём выпускной квалификационной работы составляет 41 страниц.

ABSTRACT

The topic of the given thesis is « Development of data processing server for satellite navigation systems using the Kalman filter ».

The thesis is devoted to improve GPS data accuracy using the Kalman filter.

The thesis is divided into logically bounded parts: an introduction, three main parts and a conclusion. There's also a list of references and an appendix with the main page's source code. All parts look toward the main goal of developing an application to use the Kalman filter on GPS data.

In the introduction we describe current problems with GPS data accuracy and state the tasks to complete.

The first part of my thesis is focused to describe how the satellite navigation systems work and why they sometimes have problems with the data accuracy. In the second part we elaborate a mathematical model of the Kalman filter and research iterative formulas to use in the server application. The third part is devoted to make a server application including the Kalman filter implementation on Java, develop a DBMS to collect and analyze the GPS data and to test the effectiveness of the system.

The graduation project includes 18 pictures, 1 table, 14 formulas, the list of references of 21 source and one appendix. The whole thesis covers 41 pages.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире повсеместно используются средства спутниковой навигации и отслеживания координат различных объектов для их сохранения и дальнейшего анализа. Основными системами спутниковой навигации являются GPS и российская система ГЛОНАСС. Они способны измерить координаты, скорость и направление движения любого объекта, оснащённого приёмником, настроенным на соответствующую передачу данных. Однако, из-за различных ошибок и погрешностей в измерении координат, как, например, отражение радиочастотных волн от соседних поверхностей и электромагнитных помех, зачастую информация о местоположении объектов может являться недостаточно точной.

Для увеличения точности определяемых координат существуют различные методы, от простой фильтрации данных до использования систем с дополнительными устройствами навигации, для которых изначально точно определены координаты местоположения. Фильтрация данных GPS-потока, разумеется, не даст такого точного результата, как применение более точного оборудования, но при этом она наименее ресурсозатратна.

В качестве алгоритма фильтрации данных в данной бакалаврской работе будет рассмотрен алгоритм фильтрации Рудольфа Калмана, Он повсеместно используется для решения различных задач в экономической сфере и инженерных исследованиях. Суть фильтра Калмана в системе геопозиционирования сводится к предположению следующей координаты некоторого объекта, а затем уточнение этой координаты в соответствии с проведённым измерением, которое содержит в себе некоторую погрешность. Данный фильтр позволяет снизить воздействие внешнего шума на координаты объекта и даёт хорошую оценку его местоположения в любой момент времени.

Цель бакалаврской работы – разработка программного метода

применения фильтра Калмана для обработки спутниковых данных.

Объектом бакалаврской работы является повышение точности геоданных с помощью фильтра Калмана.

В **задачи** бакалаврской работы входят следующие пункты:

- анализ проблем, возникающих в процессе отслеживания объектов спутниковыми системами навигации;
- создание математической модели фильтра Калмана для снижения влияния внешних шумов на координатные траектории движения;
- разработка сервера обработки и фильтрации данных соответствующим алгоритмом.

Бакалаврская работа включает в себя три содержательные главы.

В первой главе описывается принцип работы систем спутникового отслеживания и анализируются проблемы, возникающие в процессе их работы.

Во второй главе создаётся математическая модель фильтра Калмана и рассчитываются формулы для обработки данных.

В третьей главе описан процесс разработки сервера фильтрации и обработки данных для применения выбранного метода оптимизации координат.

В заключении подводятся итоги бакалаврской работы и формируются выводы по теме работы.

ГЛАВА 1 АНАЛИЗ ПРОБЛЕМ ПОЗИЦИОНИРОВАНИЯ ОБЪЕКТОВ СПУТНИКОВЫМИ СИСТЕМАМИ

1.1 Описание процесса спутникового отслеживания

Технологии отслеживания транспорта интересовали человечество испокон веков. В особенности острой актуальность вопроса отслеживания курса стала с наступлением эпохи мореплавания. В то время морские путешествия были опасной, но прибыльной авантюрой, зачастую, к сожалению, приводившими к летальным исходам из-за потери курса.

С появлением компаса человечество стало проявлять куда больший интерес к разработке систем навигации. Была создана буссоль, затем астролябия, способные помочь отследить координаты объекта в пространстве. Первые навигаторы представляли из себя достаточно тривиальное решение – простые бумажные карты, заключённые в защитный корпус и имеющие механизм прокрутки. Изначально такие карты приходилось прокручивать вручную, но вскоре был реализован автоматического прокручивания карт в зависимости от скорости движения транспорта. С развитием компьютерных технологий создавались электрические навигационные аппараты, позволяющие определять собственное местоположение при помощи радиосигналов.

Однако, настоящим прорывом стал запуск первого спутника в космос. Учёные при наблюдении за его траекторией заметили интересный эффект, описанный Кристианом Доплером ещё в 1842 году. Эффект Доплера заключается в том, что частота сигнала при приёме увеличивается во время приближения спутника и уменьшается при отдалении. Таким образом, зная свои координаты, можно выяснить текущее положение спутника и скорость его движения. Но этот эффект работает и в обратную сторону, что позволяет отслеживать координаты любого объекта, который имеет связь со спутником.

Разумеется, данная возможность в первую очередь рассматривалась со стороны военного превосходства, тем более, что активный вывод спутников на орбиту пришёлся на период «холодной войны» между СССР и США. Изначально спутники использовалась для наведения ракет на неподвижные объекты на Земле, но вскоре были разработаны программы для возможности наведения ракетных ударов в том числе и по движущимся воздушным целям. Вскоре после трагичной ситуации в 1982 году, когда корейский пассажирский авиалайнер нарушил воздушные границы военных объектов СССР и был сбит, руководство США разрешило использование GPS в гражданских целях, но с использованием намеренно введённой грубой ошибки для снижения точности координат. Впрочем, некоторые компании расшифровали алгоритм уменьшения точности частоты L1 и успешно компенсировали ошибку, поэтому в 2000 году намеренное огрубление точности спутниковых данных было отменено указом президента США.

На данный момент система спутникового отслеживания реализована следующим образом: каждый спутник генерирует псевдослучайный код на определённой радиочастоте. Каждый приёмник также генерирует код на этой же частоте по этому же алгоритму. Вычислив разницу во времени генерации этих двух кодов можно выяснить расстояние до соответствующего спутника. Очевидно, что имея информацию о расстоянии до одного спутника, нельзя сказать, где располагается приёмник, поскольку он может располагаться в любой точке сферы с известным радиусом. Но стоит добавить к системе ещё один спутник и из сферы область поиска сужается до окружности. Имея в своём распоряжении информацию о расстоянии до третьего спутника, область поиска сужается уже всего до двух точек.

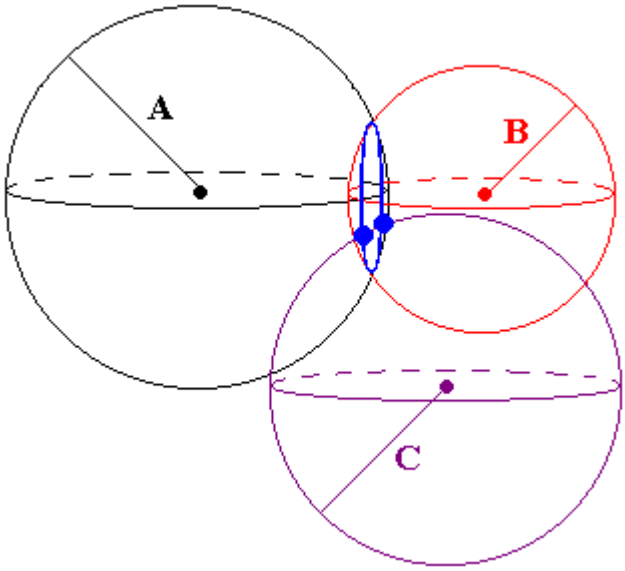

Рисунок 1.1 – Определение координат при помощи трёх спутников

В принципе, уже на этом этапе можно определить, где находится приёмник, поскольку только одна из точек находится на поверхности Земли, а вторая либо глубоко в её недрах, либо слишком высоко над её поверхностью. Однако, данная система работает исключительно в идеальных условиях, когда абсолютно точно известно расстояние до всех спутников и часы в приёмнике GPS идут в точной синхронности с часами спутника. На деле же получается, что точность часов приёмника гораздо меньше, чем хронометра в спутнике, что снижает стоимость конечного приёмника, но увеличивает погрешность вычислений. Поэтому, получив сигнал от нескольких спутников (минимум четырёх), приёмник начинает методом рекурсивных приближений корректировать свои часы до тех пор, пока не найдёт точку пересечения сигналов.

Далее координаты передаются через вышки операторов GSM-связи на серверы систем учёта телеметрических данных, откуда к ним можно получить доступ через Интернет с любого подходящего устройства. Данный процесс представлен на рисунке 1.2.

Рисунок 1.2 – Схема работы системы спутниковой навигации

1.2 Сравнительный анализ существующих систем спутниковой навигации

Наиболее распространёнными системами спутникового отслеживания являются GPS (Global Positioning System) и ГЛОНАСС (Глобальная Навигационная Спутниковая Система).

Существуют также региональные системы, не имеющие глобального применения и действующие на территории одной страны: IRNSS, индийская навигационная система, находящаяся в стадии разработки и QZSS, японская региональная навигационная система, которая имеет 4 спутника на орбите из запланированных 7. Помимо них, сейчас реализуется система спутниковой навигации, разработанная Евросоюзом под названием Galileo. Запланирован запуск 30 спутников в сумме, на данный момент на орбите находится 14 аппаратов.

Системы ГЛОНАСС и GPS по принципу действия очень похожи, но

имеют технические отличия в реализации, самое заметное из которых – орбиты расположения спутников.

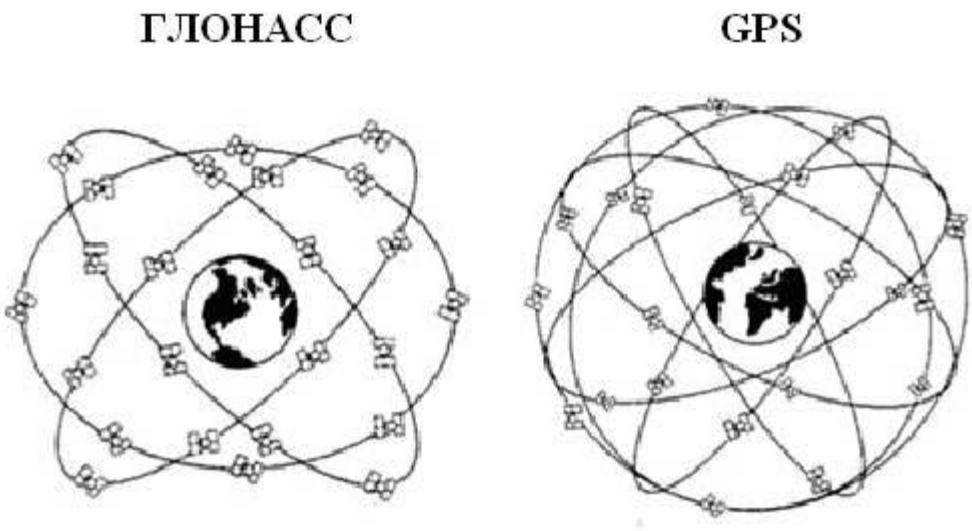

Рисунок 1.3 – Схема расположения спутников навигационных систем

Как можно заметить из рисунка 1.3, система GPS имеет большее количество орбитальных плоскостей, однако ГЛОНАСС компенсирует это увеличением количества спутников на каждой плоскости. Характеристики глобальных систем навигации указаны в таблице 1.1.

Таблица 1.1 – Сравнение систем GPS и ГЛОНАСС

Критерий	**GPS**	**ГЛОНАСС**
Количество спутников на орбите	24	24
Навигационные частоты L1, МГц	1572,42	1598,0625
Скорость кодирования, Мбит/с: C/A P	 1,023 1,23	 0,511 5,11
Период обращения спутника	11 ч. 58 мин.	11 ч. 15 мин.
Число спутников, выводимых на орбиту за один запуск	1	3

Продолжение таблицы 1.1 – Сравнение систем GPS и ГЛОНАСС

Критерий	**GPS**	**ГЛОНАСС**
Число орбитальных плоскостей	6	3
Число спутников в каждой плоскости	4	8
Наклонение орбиты в градусах	55	64,8
Высота орбиты, км	20180	19130
Система координат	WGS-84	ПЗ-90
Масса космического аппарата, кг	1055	1450
Мощность солнечных батарей, Вт	450	1250
Срок бесперебойной работы, лет	7,5	3
Эталонный часовой пояс	UTC	UTC
Способ разделения сигналов	Кодовый	Частотный
Вид орбиты	Круговая	Круговая

Из таблицы видно, что у систем навигации есть как свои достоинства, так и недостатки. Современные устройства навигации способны использовать GPS и ГЛОНАСС в связке, что повышает точность определения координат. Но даже при использовании двух систем одновременно возможны погрешности определения местоположения, которые могут быть вызваны следующими факторами:

1. Ошибки приближения времени со стороны приёмника
2. Отражения радиосигнала от зданий, металлических конструкций и других объектов
3. Вычислительные погрешности при округлении
4. Изменения расположения спутников
5. Фоновые радиопомехи

1.3 Описание проблемы точности координат маршрута

Поскольку точность получаемых данных подвергается сомнению, необходимо отследить передвижение объекта при помощи GPS и сравнить с реальным его передвижением на карте. Для выполнения данной задачи был выбран фреймворк MapBox Studio. Он позволяет создавать различные карты в соответствии с требованиями пользователя. Функции, созданные в MapBox можно импортировать в форматы CSV или GeoJSON.

CSV представляет собой простой формат хранения табличных данных, поэтому заострять внимание на нём не стоит. GeoJSON же является одним из наиболее популярных форматов, которые используются в картографических пакетах программ и онлайн-сервисов. К примеру, API Google Maps поддерживает полную поддержку координат GeoJSON. Помимо примитивных типов: точек, прямых, полигонов, GeoJSON также поддерживает мультитипы, которые объединяют в себе несколько примитивных.

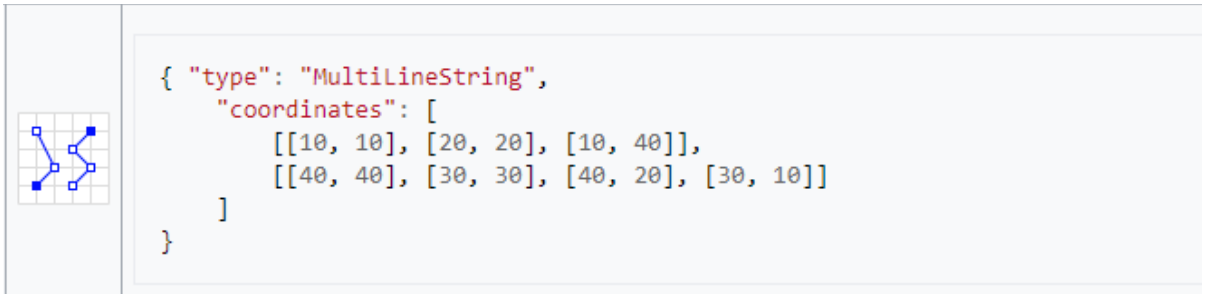

Рисунок 1.4 – Пример мультитипа GeoJSON

Для анализа точности полученных данных возьмём запись GPS-трекера на участке дороги, протяжённостью 2,5 км.

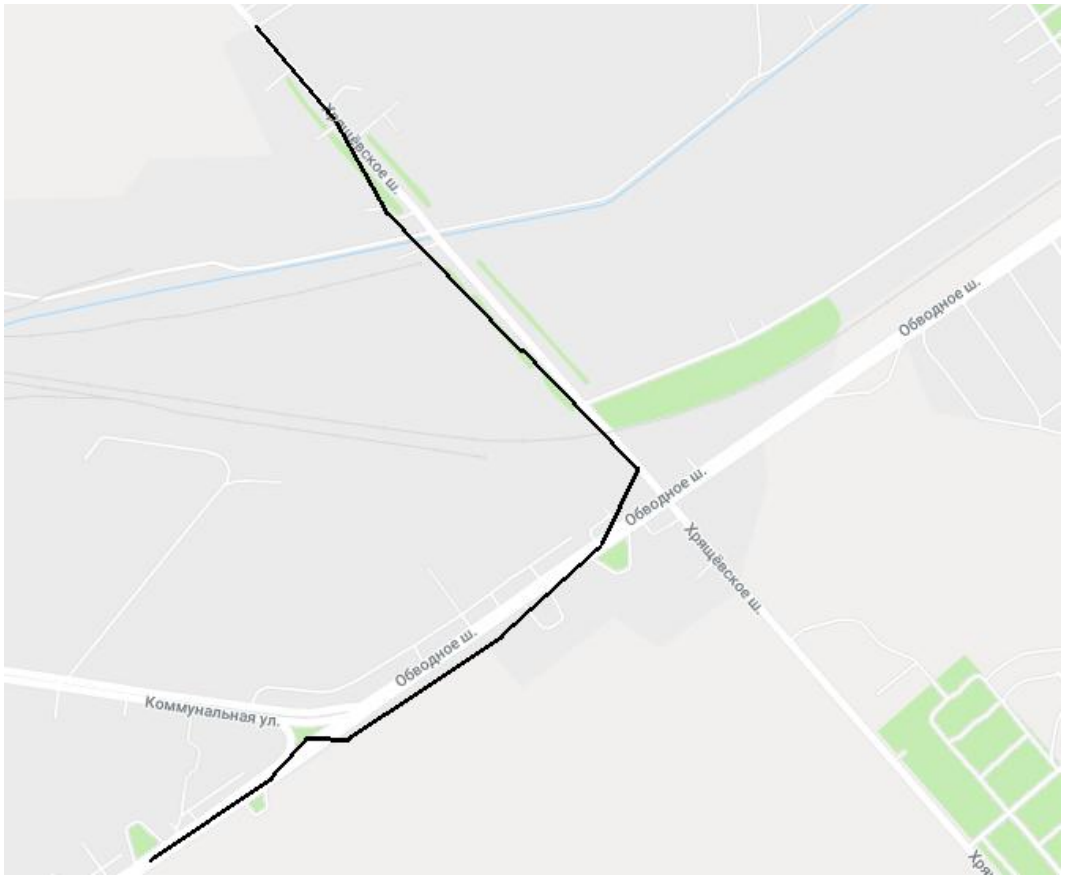

Рисунок 1.5 – Полученные навигационные данные

Как видно из рисунка 1.5, полученные данные нуждаются в корректировке. Для приближения траектории движения к реальной мы будем использовать алгоритм Калмана, широко использующийся в технических и экономических областях наук. Данный фильтр поможет сгладить данные и приблизить их к истинной траектории движения. Он предназначен для рекурсивного дооценивания вектора состояния динамической известной системы, а значит, для расчёта текущего состояния сегодня необходимо знать не только текущее измерение данных, но и предыдущее состояние самого фильтра.

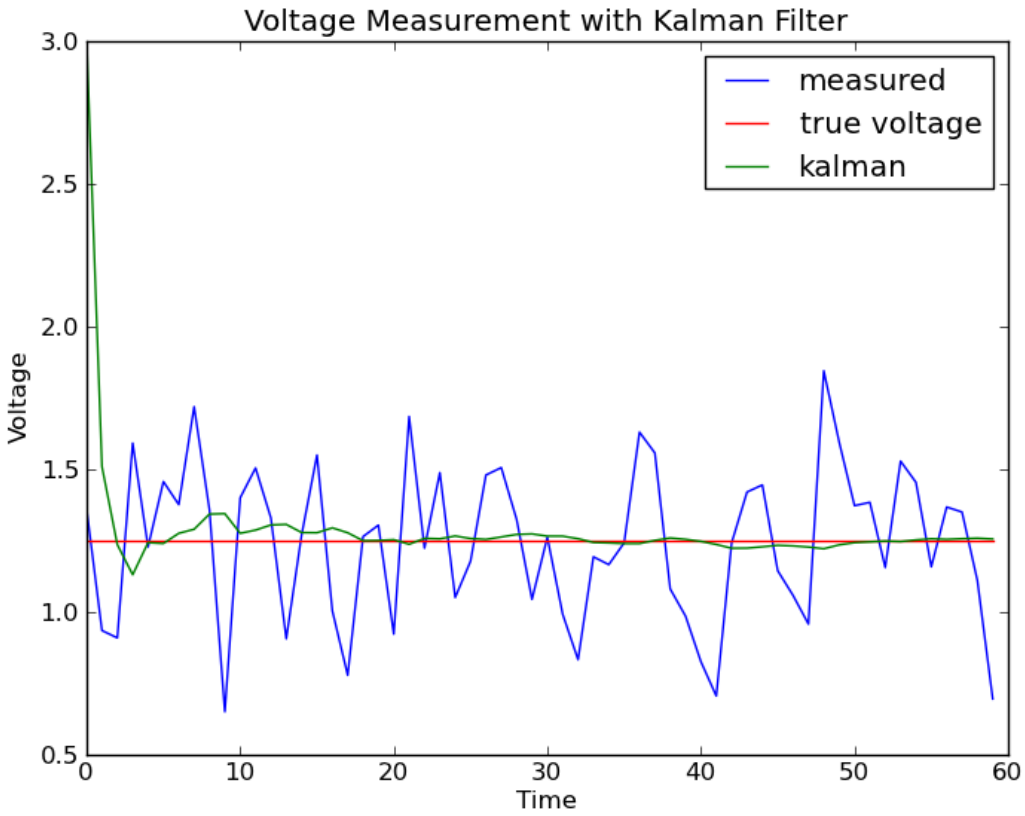

Рисунок 1.6 – Пример использования фильтра Калмана в электротехнике

На рисунке 1.6 представлен пример использования алгоритма фильтрации данных для снижения неточностей и ошибок при измерении напряжения электрического тока. Как можно заметить из графика, даже большой диапазон ошибок позволяет фильтру с достаточной степенью достоверности приблизиться к реальной величине.

Разумеется, фильтр Калмана имеет свои недостатки. Наибольшей критике подвергается тот факт, что погрешности в фильтре представлены в виде так называемого чистого белого шума, который имеет одинаковую спектральную мощность на всех своих частотах, а такой вариант шума в природе не встречается. Однако имеется возможность доработки фильтра для представления погрешностей в других, более сложных форматах.

Вывод: Исходные данные вычисляются недостаточно точно, чтобы

использовать их в изначальном виде. Для повышения точности координат стоит использовать систему корректирования входящих данных, в качестве которой был выбран фильтр Калмана, математическая модель которого будет разобрана ниже.

ГЛАВА 2 МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ АЛГОРИТМА КОРРЕКЦИИ КООРДИНАТ

2.1 Построение математической модели задачи

Для применения фильтра Калмана необходимо построить исследуемый процесс движения объекта следующим образом:

$$s_k = A_k s_{k-1} + B_k u_{k-1} + w_k \quad (1)$$

$$z_k = H_k s_k + v_k \quad (2)$$

В формуле (1) $s_k \in R^n$ – вектор состояния процесса, A – матрица $n \times n$, описывающая переход изучаемого процесса из состояния s_{k-1} в состояние s_k.

Векторная величина $u_k \in R^l$ описывает управляющее воздействие на процесс движения, в случае автомобиля, например, это сила нажатия на педаль газа. Матрица B размерностью $n \times l$ отображает вектор управляющего воздействия u в изменение состояния s. $w_k \in R^n$ является добавочной случайной величиной, характеризующей погрешности процесса, вызванные случайными внешними факторами, причём $p\ w \sim N\ 0, Q$, где Q – ковариационная матрица погрешностей процесса.

В формуле (2) отражены измерения описываемого процесса. $z_k \in R^m$ – вектор измеряемого состояния процесса, матрица H имеет размерность $m \times n$ и представляет собой состояние процесса s_k в момент измерения процесса z_k. $v_k \in R^m$ – погрешность измерений, добавочная случайная величина, и опять же, $p\ v \sim N\ 0, P$, где P – ковариационная матрица погрешностей измерений.

Поскольку мы исследуем процесс движения объекта, уравнение состояния составляется из уравнения движения тела $r_k = r_{k-1} + v_{k-1}\, dt_k + a_{k-1} \frac{dt_k^2}{2}$. Помимо этого, отсутствует какая-либо дополнительная информация

о процессе движения, поэтому будем считать, что управляющее воздействие Bu равно 0. Состояние процесса отражает вектор $s_k = \begin{matrix} x_k & y_k & v_k^x & v_k^y \end{matrix}$, где x и y – координаты объекта, а v_k^x, v_k^y – проекции скорости объекта. Тогда для изучаемого процесса уравнение (1) будет принимать следующий вид:

$$\mathrm{s_k} = \mathrm{A_k s_{k-1}} + \mathrm{G_k a_k}, \tag{3}$$

где

$$A_k = \begin{matrix} 1 & 0 & dt_k & 0 \\ 0 & 1 & 0 & dt_k \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}, \tag{4}$$

$$a_k = \begin{matrix} a_k^x \\ a_k^y \end{matrix}, \tag{5}$$

$$G_k = \begin{matrix} \frac{dt_k^2}{2} & 0 \\ 0 & \frac{dt_k^2}{2} \\ dt_k & 0 \\ 0 & dt_k \end{matrix}, \tag{6}$$

В данной модели мы рассматриваем ускорение объекта в качестве случайной погрешности процесса. Примем следующие допущения:

- Ускорения по разным осям случайны и являются независимыми величинами.
- $p\ a_k^{x,y} \sim N\ 0, \sigma_a$, то есть ускорения по обоим осям имеют нормальное распределение с нулевым математическим ожиданием и некоторым известным среднеквадратичным отклонением σ_a.

Вышеперечисленные допущения сделаны на основании следующих соображений:

1. Нет оснований полагать, что, имея значение одной компоненты вектора ускорения, можно сделать какие-либо выводы о значении другой компоненты.
2. Преимущественную часть времени объект исследования движется равномерно. Ненулевое ускорение, как правило, связано с

изменением направления движения, что происходит за короткий срок, либо с остановкой или началом движения. Стоит отметить, что разгон и торможение также происходят сравнительно быстро. Ускорение, возникающее в ситуациях обгона или опережения, также отличается от нуля в течение небольших промежутков времени.

Таким образом, в формуле (3) погрешность $G_k a_k$ выполняет роль w_k из формулы (1). Для дальнейших вычислений необходимо получить значения матрицы ошибок Q.

$$Q_k = cov\ w_k = M\ w_R w_k^T - M\ w_k\ M\ w_k^T = \\ = M\ G_k a_k a_k^T G_k^T = G_k M\ a_k a_k^T\ G_k^T, \tag{7}$$

где M – математическое ожидание. Поскольку нам известно, что компоненты вектора a_k (5) являются случайными и независимыми величинами, то $M\ a_k a_k^T = \begin{matrix} \sigma_a^2 & 0 \\ 0 & \sigma_a^2 \end{matrix}$. Следовательно, формула (7) будет принимать следующий вид:

$$Q_k = G_k G_k^T \sigma_a^2 \tag{8}$$

Вектор измерения z_k для рассматриваемой задачи изменится следующим образом:

$$z_k = \begin{matrix} \chi_k \\ \gamma_k \\ \vartheta_k^x \\ \vartheta_k^y \end{matrix}, \tag{9}$$

где χ_k и γ_k – координаты GPS, полученные приёмником; ϑ_k^x, ϑ_k^y – скорость объекта, также полученная приёмником. Матрица H из формулы (2) принимается равной единичной матрице 4×4, поскольку в рамках данной задачи считается, что измерение z_k есть линейная комбинация вектора состояния и некоторых случайных погрешностей. Матрицу погрешности измерений считаем заданной. Один из вариантов её вычисления – использование данных с приёмника GPS-координат о предполагаемой точности измерения.

2.2 Применение фильтра Калмана к построенной математической модели

Для применения фильтра Калмана к данной задаче введём следующие понятия:

- s_k – оценка состояния объекта в момент времени *k*, полученная в результате наблюдений до момента *k* включительно.
- s_k – некорректированная оценка состояния объекта в момент времени *k*.
- P_k – апостериорная ковариационная матрица погрешностей, которая задаёт оценку точности полученного вектора состояния.
- P_k – нескорректированная P_k.

Начальное положение объекта известно, поэтому матрица P_0 задаётся как нулевая.

Каждая итерация при применении фильтра Калмана состоит из двух этапов: экстраполяция и коррекция. На этапе экстраполяции вычисляются приближенные значения s_k по оценке вектора предыдущего состояния s_{k-1} и ковариационная матрица ошибок P_k.

$$s_k = A_k s_{k-1}, \tag{10}$$

$$P_k = P_{k-1} A_k^T + Q_k, \tag{11}$$

где матрица A_k известна нам из формулы (4), а матрица Q_k исчисляется по формуле (8).

Затем следует этап коррекции, на котором производится вычисление матрицы усиляющих коэффициентов K_k:

$$K_k = P_k H^T \left(H P_k H^T + R \right)^{-1}, \tag{12}$$

где R и H известны. K_k будет использоваться для корректировки оценки состояния s_k и матрицы ошибок P_k:

$$s_k = s_k + K_k \left(z_k - H s_k \right), \tag{13}$$

$$P_k = \ I - K_k H \ P_k, \tag{14}$$

где I – единичная матрица.

Необходимо отметить тот факт, что для использования вычисленных соотношений необходимо согласовать единицы измерения параметров объекта, то есть привести к единому виду. В исходных данных, к примеру, прямоугольные координаты объекта по широте и долготе приводятся в угловых единицах, а скорость – в метрических. Ускорение рассчитывать также проще исходя из метрических единиц.

Вывод: в ходе вычислений была получена математическая модель задачи движения объекта, а также итерационные формулы для применения к математической модели фильтрации данных по алгоритму Калмана.

ГЛАВА 3 РАЗРАБОТКА СЕРВЕРНОГО ПРИЛОЖЕНИЯ ОБРАБОТКИ И ХРАНЕНИЯ КООРДИНАТНЫХ ДАННЫХ

3.1 Требования к разработке серверной части системы обработки координат

Предполагается, что система будет получать для обработки и хранения следующие типы данных:

- географические координаты (в градусах широты и долготы)
- скорость движения
- направление вектора движения
- время измерения координат

Также необходима обработка идентификационных данных для возможности отслеживания нескольких транспортных средств одновременно. В качестве них будут выданы идентификаторы для каждой единицы техники, для каждого водителя и для каждой сессии.

Таким образом, система должна принимать данные от GPS-передатчиков, заносить и сохранять их в базе данных, применять фильтр Калмана, математическую модель которого мы построили в предыдущей главе, и затем возвращать данные с применённым фильтром для отображения на сайте.

К системе предъявляются следующие требования:

1. система имеет надёжную СУБД для хранения данных;
2. поток данных поступает в систему посредством обработки POST и GET-запросов;
3. система применяет фильтр Калмана для устранения шумов в исходных данных;
4. система является многопоточной, для возможности реализации

нескольких запросов в один момент времени;

5. передача отфильтрованных данных производится путём обработки GET-запросов
6. в системе должна присутствовать авторизация и аутентификация пользователей

3.2 Серверная архитектура системы обработки координат

В качестве языка реализации серверных приложений был выбран язык программирования Java. Java – это объектно-ориентированный язык программирования, разработкой которого занимается компания Oracle. Приложения на нём характеризуются кроссплатформенностью, устойчивостью к серверным нагрузкам и удобной работой с базами данных. Среди других преимуществ Java стоит отметить наличие обширной документации по работе, простоту и удобство написания кода, наличие классов для обработки HTTP-запросов, а также множество вариантов реализации многопоточности в разрабатываемых программах. Конечно, этот язык программирования не лишён недостатков – Java печально известна как язык, требующий высокопроизводительной аппаратной части, особенно в плане количества оперативной памяти. Для обработки запросов используется фреймворк Jersey, поскольку в нём имеется встроенная поддержка формата JSON, поддерживается асинхронная обработка запросов, а также простота использования. Для управления базой данных используется СУБД MySQL, благодаря простоте и удобству использования, быстрой работе, поддержки многопоточности, а также обширной документации, способствующей быстрому обучению в использовании.

Порядок работы проектируемой системы следующий:

1. В базу данных заносятся сведения о транспортных средствах и их водителях.

2. Между водителями и транспортными средствами устанавливаются соответствующие связи.
3. Система получает данные о передвижении транспорта, описанные выше.
4. Система обрабатывает полученные данные фильтром Калмана для снижения влияния шума на траекторию движения.
5. Система отправляет данные на сайт либо приложение по соответствующему запросу.

Графическое представление работы системы представлено на рисунке 3.1.

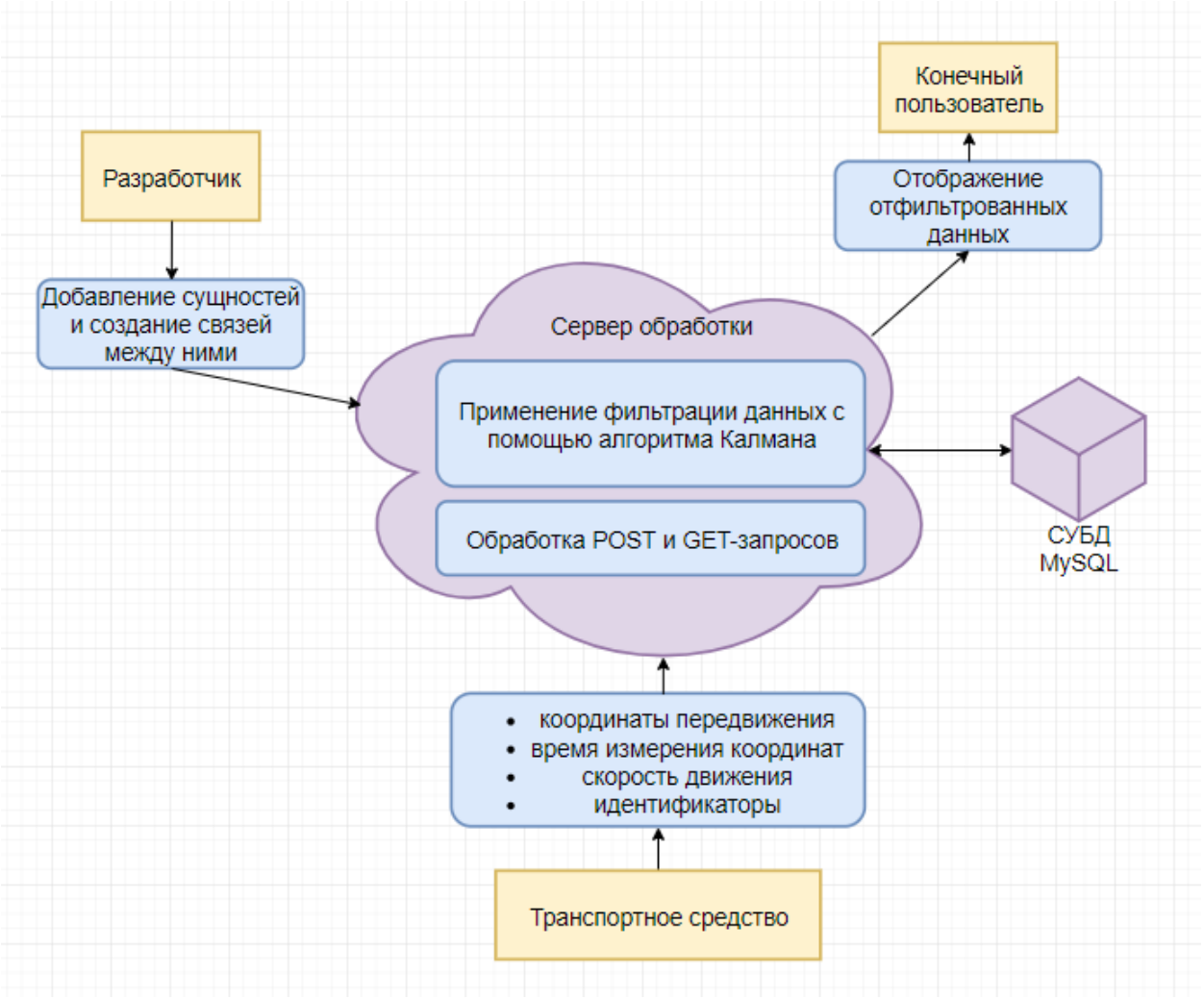

Рисунок 3.1 – Визуальная интерпретация работы системы

3.3 Создание базы данных системы обработки координат

Как было сказано ранее, в качестве системы управления базой данных

рассматривается решение MySQL, которое представляет из себя реляционную СУБД, которая распространяется под лицензией GNU General Public License.

К преимуществам данной СУБД относятся:

1. Поддержка широкого спектра операционных систем.
2. Богатый набор API и библиотек для множества языков программирования, среди которых присутствует и Java.
3. Свободная лицензия.
4. Поддержка кешей запросов.
5. Простота установки.
6. Высокая производительность.

MySQL будет использоваться для хранения данных, переданных системе, а также о самих водителях и транспортных средствах. Для этого также необходимо создать пользователей в системе с административными правами на добавление, управление и удаление записей в таблицы. Также необходимо создать отдельную таблицу для описания связей между водителем и его транспортным средством, поскольку может произойти ситуация, когда одному объекту не соответствует другой, либо соответствует несколько. К примеру, если один водитель использует разные транспортные средства, либо наоборот, ни одного.

```
CREATE TABLE user
(
  id    int           PRIMARY KEY,
  login nvarchar(40)  NOT NULL  UNIQUE,
  hash  BINARY(32)    NOT NULL
);

CREATE TABLE truck
(
  id      int           PRIMARY KEY,
  mark    int           NOT NULL,
  model   int           NOT NULL,
  color   nvarchar(20)  NOT NULL,
  number  nvarchar(20)  NOT NULL
);

CREATE TABLE driver
(
  id          int           PRIMARY KEY,
  firstName   nvarchar(32),
  secondName  nvarchar(32),
  patronymic  nvarchar(32)
);
```

Рисунок 3.2 – Таблицы базы данных

На рисунке 3.2 продемонстрированы таблицы, используемые для идентификации пользователя системы, транспортного средства и водителя. Стоит обратить внимание на бинарное поле hash, в котором впоследствии будет храниться MD5-хеш строки, получаемой при помощи обработки специальным алгоритмом строк логина и пароля. Данная процедура обеспечивает безопасность хранения паролей в системе, поскольку даже имея доступ к базе данных невозможно узнать пароль пользователя системы.

```
CREATE TABLE ownership
(
  id        int PRIMARY KEY,
  driverId  int NOT NULL,
  truckId   int NOT NULL,
  CONSTRAINT FOREIGN KEY (driverId) REFERENCES driver(id),
  CONSTRAINT FOREIGN KEY (truckId)  REFERENCES truck(id)
);
```

Рисунок 3.3 – Таблица принадлежности

В таблице, представленной на рисунке 3.3 используются внешние ключи driverId и truckId. На это есть свои причины: внешние ключи ускоряют выборку данных при использовании запроса с JOIN; такие ключи способствуют сохранению ссылочной целостности в результате добавления новых строк в базу данных; внешние ключи помогают понять взаимосвязи между таблицами простым визуальным образом.

Для хранения данных о каждой поездке были созданы ещё две таблицы.

```
CREATE TABLE trip
(
  tripId      int PRIMARY KEY AUTO_INCREMENT,
  ownershipId int NOT NULL,
  kalman      bit NOT NULL    DEFAULT 0,
  CONSTRAINT FOREIGN KEY (ownershipId) REFERENCES ownership (id)
);

CREATE TABLE track
(
  tripId      int             NOT NULL,
  lat         decimal(10, 8)  NOT NULL,
  lng         decimal(11, 8)  NOT NULL,
  orientation float           NOT NULL,
  fuel        float           NOT NULL,
  speed       int             NOT NULL,
  time        timestamp       NOT NULL,
  CONSTRAINT FOREIGN KEY (tripId) REFERENCES trip(tripId)
);
```

Рисунок 3.4 – Таблицы данных о передвижении

В этих таблицах хранятся данные, которые будут подвергнуты фильтрованию с помощью алгоритма Калмана. Особое внимание требуется обратить на логическое поле kalman, которое показывает, были ли данные обработаны или нет. Схема связей таблиц в базе данных представлена на рисунке 3.5.

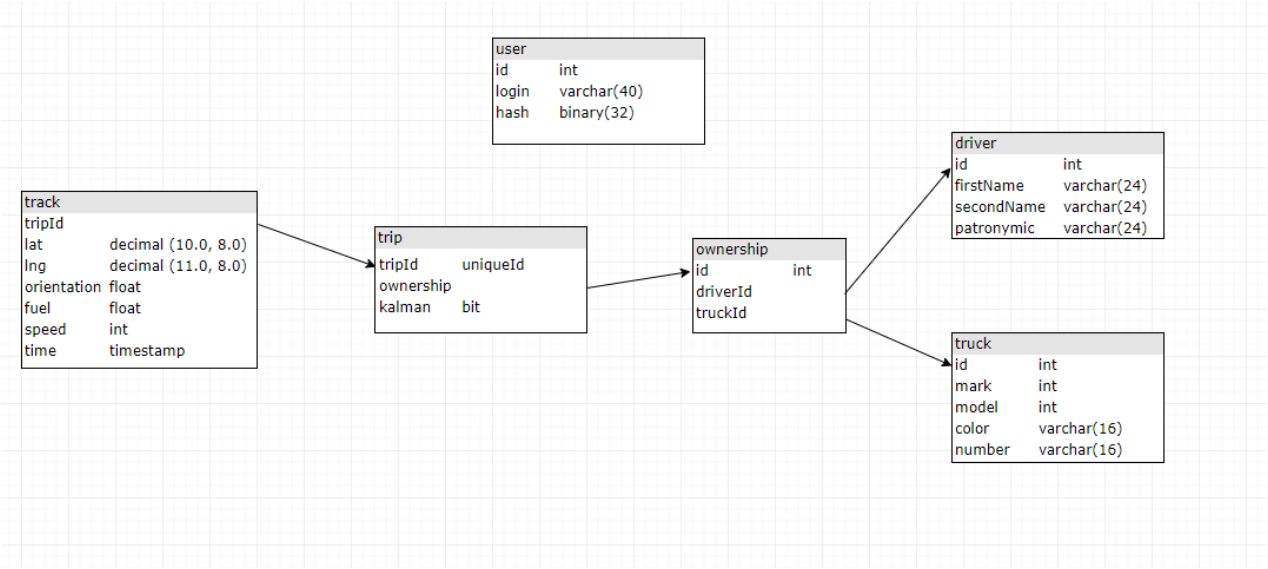

Рисунок 3.5 – Графическое представление связей таблиц

3.4 Разработка программного кода сервера обработки координат

Для начала разработки программного кода необходимо определиться с тем, каким образом клиент и сервер будут обмениваться информацией. Под термином «клиент» в данной ситуации понимается приложение или сайт, куда будет отправляться фильтрованная информация. Поскольку основным форматом картографических данных является GeoJSON, о котором было сказано ранее, то для передачи данных будет использоваться более общий формат JSON.

JSON (JavaScript Object Notation) представляет собой текстовый формат передачи данных, который базируется на JavaScript. Как и большинство других текстовых форматов, он достаточно легко может быть прочитан человеком. Благодаря лаконичности, с его помощью можно передавать большие объёмы данных, а поддержка JSON в большинстве языков программирования способствует удобству в работе с разными языками.

В разрабатываемой системе передача информации между сервером и клиентом основывается на JSON. В принципе, простота системы позволяет обойтись двумя основными шаблонами, с помощью которых можно передавать любые данные. В рамках обработки запросов сервером основным

признаком является результат обработки – успех, либо неудача. Клиенту, как правило, ничего обрабатывать не требуется. Поэтому в каждом ответе от сервера в объекте JSON будет присутствовать поле success, в котором передаётся информация, был ли выполнен запрос. Помимо этого поля, сервер будет передавать клиенту различную информацию в соответствии с клиентским запросом, а если запрос не обработан, то ошибку.

```
public class AnswerBoolean {
    private boolean success;
    private String error;

    public AnswerBoolean(boolean success) { this.success = success; }

    public AnswerBoolean(boolean success, String error)
    {
        this.success = success;
        this.error = error;
    }
}
public class AnswerObject<T> extends AnswerBoolean {
    private T object;

    public AnswerObject(boolean success, T object)
    {
        super(success);
        this.object = object;
    }

    public AnswerObject(boolean success, T object, String error)
    {
        super(success, error);
        this.object = object;
    }
}
```

Рисунок 3.6 – Классы, используемые для передачи данных

Класс AnswerBoolean отправляется в том случае, когда клиенту не нужна какая-либо информация, помимо статуса успешности обработки запроса. Допустим, при аутентификации пользователя в системе. В таком случае отправляется либо сообщение об успешной аутентификации, либо ошибка и её описание.

Класс AnswerObject используется в тех случаях, когда запрашиваемая информация не может представлять из себя логическую единицу,

свидетельствующую об успехе обработки. Предположим, клиент может запросить информацию о транспортном средстве определённого цвета. Тогда, если запрос успешно обработан, в поле object будут находиться соответствующие запросу данные. Если же запрос не был обработан, то клиент опять получит ответ в AnswerBoolean, поскольку нет возможности вернуть объект.

Для того, чтобы преобразовать объект класса в текстовые данные JSON используется библиотека GSON. Данная библиотека также имеет возможность проводить обратную операцию – из JSON строки получать объект класса. Эта библиотека была выбрана по следующим причинам:

- позволяет обрабатывать как единичные классы, так и их шаблоны, коллекции и вложенные классы;
- при конвертации JSON-строки в объект класса преобразование идёт по элементам объекта класса, что даёт возможность игнорирования дополнительных полей в JSON-строке;
- позволяет задать способ обработки null-объектов.

Преобразование строки JSON в класс необходимо в тех случаях, когда клиенту необходимо отправить на сервер более сложный объект, чем, допустим, id, который можно передать простым GET-запросом. Для обращения к серверу с более сложным объектом необходимо использование POST-запроса, а в нём как раз транслировать JSON-объект. POST-запросами обычно производится процедура аутентификации пользователя.

Кроме того, на стороне сервера необходима реализация механизма сессий. В качестве примера будет рассмотрен POST-запрос аутентификации пользователя login, реализованного в процессе создания системы

```
@POST
@Path("/login")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public Response login(User user, @Context HttpServletRequest request)
```

Рисунок 3.7 – Аннотации функции login

На рисунке 3.7 указана аннотация @Path, которая указывает URL для прохождения аутентификации. Аннотации @Produces и @Consumes определяют тип объекта на входе и на выходе из функции, в нашем случае это объект JSON. Входной параметр рассматриваемой функции – класс User.

```
public class User {
    private String login;
    private String password;

    public String getLogin() { return login; }

    public String getPassword() { return password; }

    public void setLogin(String login) { this.login = login; }

    public void setPassword(String password) { this.password = password; }
}
```

Рисунок 3.8 – Объект класса User

Описанные выше аннотации позволяют автоматически преобразовывать JSON-строку из POST-запроса в объект класса. Второй объект в данном примере – объект HttpServletRequest, который включает в себя информацию HTTP-запроса. При аутентификации пользователя в первую очередь проверяется, активна ли сессия или нет.

```
if (request.getSession( create: false) != null)
{
    return Response.status(200).entity(new AnswerBoolean( success: false,
        error: "You are already signed in.")).build();
}
```

Рисунок 3.9 – Проверка текущей аутентификации

Для проверки запрашивается флаг текущей сессии. В случае отсутствия в текущий момент, сессия не создаётся, за что отвечает аргумент false данной функции. Если же сессия существует и активна, то пользователь получает сообщение о том, что уже аутентифицирован в системе. Если же сессия не

активна, то необходимо посчитать MD5-хеш логина и пароля, сравнить с имеющимся в базе для данного логина и только в случае полного совпадения строк проводить аутентификацию. Использование хешей вместо паролей позволяет обезопасить как пользователя от компрометации данных, так и самого администратора СУБД.

```
PreparedStatement stmt = conn.prepareStatement( sql: "SELECT id FROM user WHERE login=? AND hash=?");

stmt.setString( parameterIndex: 1, user.getLogin());
stmt.setString( parameterIndex: 2, hash);

ResultSet rs = stmt.executeQuery();

AnswerBoolean ans;

// Есть в базе запись с таким же логином и хешем, "аутентификация пройдена"
if (rs.next()) {
    int id = rs.getInt( columnLabel: "id");

    HttpSession session = request.getSession( create: true);
    session.setAttribute( name: "userId", id);

    ans = new AnswerBoolean( success: true);
} else {
    ans = new AnswerBoolean( success: false,  error: "Wrong login or password.");
}
```

Рисунок 3.10 – Поиск хеша в базе данных

Программный код данного процесса изображён на рисунке 3.10. Как видно из кода, если совпадение по логину и хешу найдено, то для переменной сессии присваивается значение user id текущего пользователя, а сам пользователь получает ответ об успешно пройденной аутентификации. Использование переменных сессий обусловлено рядом причин, среди которых первое место занимает безопасность. У пользователя нет доступа к этим переменным, и он никак не может на них повлиять, что повышает информационную целостность последней. В случае, если совпадений по базе не обнаружилось, соединение с базой данных закрывается, а пользователь получает ответ о безуспешной попытке аутентификации.

Далее рассмотрим ту часть сервера, которая отвечает непосредственно за применение фильтра Калмана к данным геопозиционирования. Система построена таким образом, что клиент посылает GET-запрос с

идентификационными данными на сервер, после чего получает соответствующий id поездки. Затем с определёнными промежутками во времени клиент посылает POST-запросы, передавая данные о координатах, скорости, направлению движения и т.д. После остановки транспорта в пункте конечного назначения клиент отправляет GET-запрос на сервер, а последний проводит корректировку данных маршрута с применением описанного фильтра данных и отправляет их конечному клиенту. Полный программный код применения фильтрации данных представлен в приложении.

Рисунок 3.11 – Графическое представление исходных координат

На рисунке 3.11 изображены тестовые координаты, полученные в ходе предварительных испытаний системы при помощи портативного GPS-трекера. Это изображение наглядно демонстрирует, насколько большое влияние оказывают шумы и погрешности измерений на точность координат. Стоит отметить, что тестовые координаты были получены на достаточном удалении от зданий и других объектов, способных исказить сигнал. В районах плотной застройки траектория движения с высокой долей вероятности содержала бы так называемые «пики» – резкие отклонения от

маршрута, вызванные интерференцией сигнала. Такие пики при частом проявлении способны увеличить общую ошибку координат до значений, которые трудно сгладить даже с использованием фильтрации данных. Поэтому был выбран участок местности, где ошибка геопозиции минимальна. Затем полученные данные были обработаны фильтром Калмана, результат обработки представлен на рисунке 3.12.

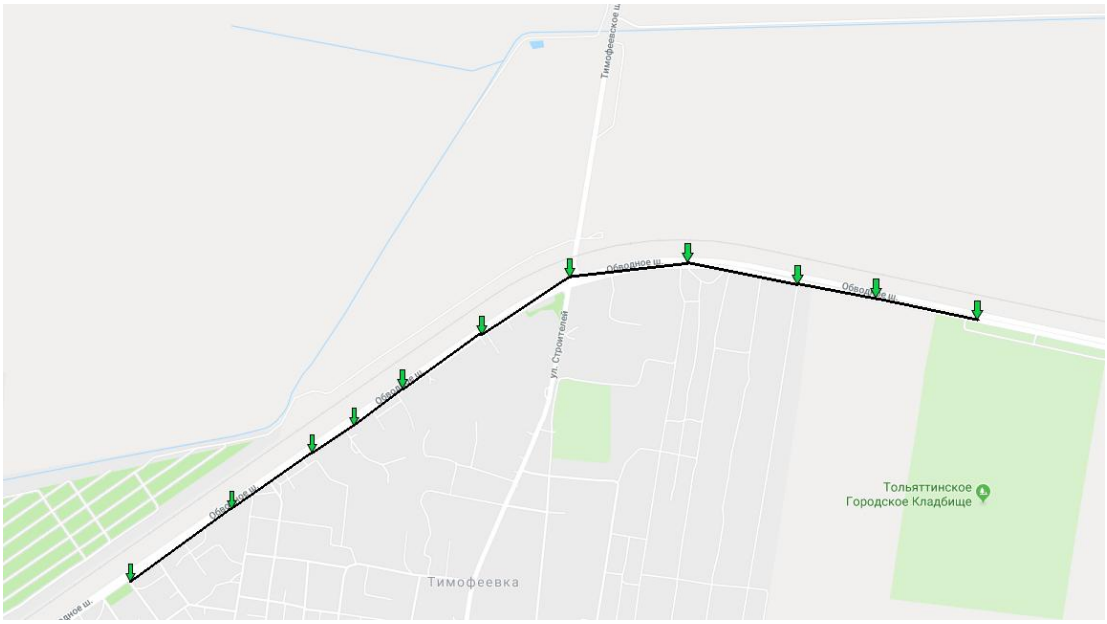

Рисунок 3.12 – Графическое представление обработанных координат

На рисунке 3.12 видно, что использование фильтра позволяет максимально приблизить маршрут к действительному и свести к минимуму влияние погрешностей измерения. При уменьшении интервала передачи данных можно добиться ещё более правдоподобного результата, но такой метод негативно отразится на времени выполнения обработки, поскольку объём обрабатываемой информации, а, следовательно, количество циклов выполнения алгоритма значительно возрастёт.

Таким образом, по итогам проделанной работы можно сделать однозначный вывод о том, что использование фильтра Калмана при обработке данных систем спутниковой навигации позволяет существенно снизить погрешности измерений и устранить ошибки измерения координат, а

также приблизить маршрут исследуемого объекта к действительному, причём точность сглаживания при применении данного фильтра увеличивается с уменьшением интервала времени между отправкой пакетов координат.

ЗАКЛЮЧЕНИЕ

Настоящая бакалаврская работа состоит из введения, трех глав и заключения.

Во введении обозначена актуальность работы, определены предмет и объект исследования, поставлены цели и указаны задачи, которые необходимо решить для достижения цели.

В первой главе описан принцип работы спутниковых систем навигации, проведён сравнительный анализ существующих разработок в сфере спутникового отслеживания, а также описывается проблема точности полученных геоданных.

Во второй главе происходит разработка математической модели фильтра Калмана, а также рассчитываются итерационные формулы для его применения.

В третьей главе описывается разработка сервера обработки и фильтрации данных, создание базы данных для хранения информации, реализация программного кода фильтрации данных и проведён анализ результатов.

Проблемой исследования в рамках настоящего бакалаврского проекта являлась недостаточная точность координатных данных, полученных со спутниковых систем навигации в их изначальном виде.

Целью исследования являлось создание сервера обработки и фильтрации спутниковых данных при помощи фильтра Калмана.

Для достижения поставленной цели были решены следующие задачи:

1. Проведён анализ проблем, возникающих в процессе отслеживания объектов спутниковыми системами навигации.
2. Создана математическая модель фильтра Калмана для снижения влияния внешних шумов на координатные траектории движения.

3. Разработан сервер обработки и фильтрации данных.

Теоретической основой исследования стали работы отечественных и зарубежных авторов, законодательные акты, ведомственные материалы, рекомендации и инструкции.

Таким образом, цель работы можно считать достигнутой.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Нормативно-правовые акты

1. ГОСТ 7.32-2001. Отчет о научно-исследовательской работе. Структура и правила оформления.
2. ГОСТ 7.1-2003. Библиографическая запись. Библиографическое описание документа.
3. ГОСТ 7.82-2001. Библиографическая запись. Библиографическое описание электронных ресурсов.
4. ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения (ИСО 5807–85) [Текст]. Введен 70 1992–01–01. – М.: Изд-во стандартов, 1992. – 14 с. – (Единая система программной документации).
5. ГОСТ 2.105-95. Общие требования к текстовым документам [Текст]. – М.: Изд-во стандартов, 1996. – 29 с. – (Единая система конструкторской документации).
6. ГОСТ 34.320-96. Информационные технологии. Система стандартов по базам данных. Концепции и терминология для концептуальной схемы и информационной базы. Введ. 2001-07-01.- М.: Изд-во стандартов, 2001. – 46с. - (Основополагающие стандарты).

Научная и методическая литература

7. Згуровский, М. З. Аналитические методы калмановской фильтрации для систем с априорной неопределенностью / М. З. Згуровский, В. Н. Подладчиков. – Киев: Наукова думка, 1995. – 298 с.
8. Богданов М.Р. Применение GPS/ГЛОНАСС. Издательство: ИД Интеллект, 2012 г. – 137 стр.
9. Яценков В.С. Основы спутниковой навигации. Системы GPS, NAVSTAR и ГЛОНАСС. Издательство: Москва, 2005 г. – 272 стр.
10. Оценивание состояния динамической системы в условиях неопределенности / В. И. Ширяев, В. И. Долбенков, Е. Д. Ильин, Е. О. Подивилова // Экстремальная робототехника: сб. докл. Междунар.

науч.-техн. конф. – СПб., 2011. – С. 234–243.

11. Кац, И. Я. Минимаксная многошаговая фильтрация в статистически неопределенных ситуациях / И. Я. Кац, А. Б. Куржанский // Автоматика и телематика. – 1978. – № 11. – С. 79–87.
12. Молчанов А.Ю. Системное программное обеспечение. Учебник для вузов. 3-е изд. Издательство: СПб.:Питер, 2010 г. – 400 стр.
13. Энтони Гонсалвес. Изучаем Java EE 7. Издательство: СПб.:Питер, 2014 г. – 640 стр.
14. Климченко, В. В. Планирование измерений параметров контролируемых технических объектов / В. В. Климченко // Труды Междунар. симп. Надежность и качество. – 2011. – Т. 1. – С. 46–48.
15. Аливер В.Ю. Применение расширенного фильтра Калмана для демодуляции хаотических колебаний: Дисс. канд. техн. наук. - М., 2005.

Литература на иностранном языке

16. Maged N Kamel Boulos, Steve Wheeler. How smartphones are changing the face of mobile and participatory healthcare: an overview, with example from eCAALYX, BioMedical Engineering OnLine, 2011
17. Cyprian M. Wronka, Matthew W. Dunnigan. Internet remote control interface for a multipurpose robotic arm, Electronic & Computer Engineering, 2009.
18. Pecora L.M., Carroll T.L. Synchronization in Chaotic Systems, Physical Review Letters, Volume 64, Number 8, 19 February 1990, pp.821-824.
19. Keqiang Zhang, Bingjie Qu. Design and Implementation of Browser based GPS/GPRS Vehicle Positioning and Tracking System, MATEC Web of Conferences, 2015.
20. Tang Y.S., Mees A.I., Chua L.O. Synchronization and Chaos, IEEE Transactions on Circuits and Systems, Vol. CAS-30, No.9, September 1983.

21. E.A. Wan and A.T. Nelson, ‘‘Neural dual extended Kalman filtering: Applications in speech enhancement and monaural blind signal separation’’, in Proceedings of Neural Networks for Signal Processing Workshop, IEEE, 1997.

ПРИЛОЖЕНИЕ

```
package robotutils.filters;

import org.apache.commons.math3.linear.MatrixUtils;
import org.apache.commons.math3.linear.RealMatrix;
import org.apache.commons.math3.linear.LUDecomposition;

public class KalmanFilter {


    protected RealMatrix _x;


    protected RealMatrix _P;

    protected RealMatrix _K;

    protected RealMatrix _F;

    protected RealMatrix _Q;


    protected RealMatrix _B;

    protected RealMatrix _H;



    protected RealMatrix _R;


    public KalmanFilter(RealMatrix x, RealMatrix P) {
        _x = x;
        _P = P;
    }

    public KalmanFilter(RealMatrix x, RealMatrix P,
            RealMatrix F, RealMatrix Q, RealMatrix B) {
        this(x, P);

        _F = F;
        _Q = Q;
        _B = B;
    }

    public KalmanFilter(RealMatrix x, RealMatrix P,
            RealMatrix F, RealMatrix Q, RealMatrix B,
            RealMatrix H, RealMatrix R) {
        this(x, P, F, Q, B);

        _H = H;
        _R = R;
    }


    public void predict(RealMatrix u) {
```

```
        predict(_F, _Q, _B, u);
    }


    public void predict(RealMatrix F, RealMatrix Q, RealMatrix B, RealMatr
ix u) {
        _x = F.multiply(_x).add(B.multiply(u));
        _P = F.multiply(_P).multiply(F.transpose()).add(Q);
    }

    public void update(RealMatrix z) {
        update(_H, _R, z);
    }


    public void update(RealMatrix H, RealMatrix R, RealMatrix z) {
        RealMatrix I = MatrixUtils.createRealMatrix(_K.getRowDimension(),
H.getColumnDimension());
        int dim = Math.min(_K.getRowDimension(), H.getColumnDimension());
        I = I.add(MatrixUtils.createRealIdentityMatrix(dim));


        RealMatrix y = z.subtract(H.multiply(_x));
        RealMatrix S = H.multiply(_P).multiply(H.transpose()).add(R);
        RealMatrix invS = new LUDecomposition(S).getSolver().getInverse();

        _K = _P.multiply(H.transpose()).multiply(invS);
        _x = _x.add(_K.multiply(y));
        _P = I.subtract(_K.multiply(H)).multiply(_P);
    }

    public void setState(RealMatrix x) {
        _x = x;
    }


    public RealMatrix getState() {
        return _x.copy();
    }
    public void setStateCov(RealMatrix P) {
        _P = P;
    }


    public RealMatrix getStateCov() {
        return _P.copy();
    }

    public RealMatrix getKalmanGain() {
        return _K.copy();
    }


    public void setProcessModel(RealMatrix F) {
        _F = F;
    }
    public RealMatrix getProcessModel() {
        return _F.copy();
    }
```

```
    public void setProcessNoise(RealMatrix Q) {
        _Q = Q;
    }


    public RealMatrix getProcessNoise() {
        return _Q.copy();
    }

    public void setControlModel(RealMatrix B) {
        _B = B;
    }

    public RealMatrix getControlModel() {
        return _B.copy();
    }
    public void setObsModel(RealMatrix H) {
        _H = H;
    }

    public RealMatrix getObsModel() {
        return _H.copy();
    }
    public void setObsNoise(RealMatrix R) {
        _R = R;
    }
    public RealMatrix getObsNoise() {
        return _R.copy();
    }

}
```