Содержание

Введение

В последнее время всё более актуальной становится проблема навигации внутри помещений, а также предоставления посетителям услуг, основанных на их местоположении и предпочтениях. Здания становятся всё более объёмными и нередко имеют довольно сложную структуру, ориентироваться в которой могут лишь те, кто постоянно посещает такие здания, а для неподготовленного человека ориентирование в таких местах превращается в пытку.

Благодаря indoor-навигации (навигации внутри помещений) появляются новые инструменты для маркетинга: проходя мимо магазина, человек может моментально узнать о проводимых в нем акциях, мероприятиях, товарах и предоставляемых услугах, благодаря всплывающему сообщению на экране своего телефона (так называемом “Geo-fencing”, причём предложенные ему предложения будут учитывать его интересы, так как можно учитывать информацию о его прошлых покупках), либо просто получить уведомление при приближении к определенному месту (второе направление indoor-навигации, называемое «Geo-aware»), а владельцы – получать статистическую информацию («тепловые карты» посетителей – своеобразный и очень мощный offline-аналог Google Analytics), основанные на перемещениях клиентов внутри торговых залов (таким образом понять, какие отделы и товары пользуются повышенным интересом, очень легко). Рынок подобной геоконтекстной рекламы уже измеряется миллиардами долларов, и с развитием систем indoor-навигации ожидается его стремительный рост.

Кроме того, решения, применяемые в indoor-навигации, помогают и в ориентировании вне зданий, на улице – там, где в условиях плотной застройки использование систем спутниковой навигации затруднено. Особенно эта проблема актуальна для Японии с высокой плотностью городской застройки.

Bluetooth-маяки, представляющие собой один из способов решения проблемы локации, также являются частью Интернета вещей, и способны изменить представление о самом Интернете вещей. Google представила концепцию "physical web которая призвана объединить два мира: реальный и виртуальную Интернет-сеть. Смартфон, сканируя ближайшие к нему маячки, получает из их сигнала встроенные URL-ссылки, доступные пользователю.

Таким образом, пользователь может мгновенно получить доступ к информации, определенной в ближайшем локационном контексте. Кроме того, это шаг навстречу унифицированному, бесшовному интерфейсу взаимодействия с цифровым миром.

Но современный этап развития технологии внутренней навигации представлен в основном технологиями для навигации роботов по маякам. Кроме того, существует множество готовых продуктов и платформ от производителей с мировым именем. В то же время сообщество программистов не располагает подходящими материалами, чтобы оценить существующие варианты алгоритмов трилатерации и выбрать лучший способ в рамках собственных приложений или свободно распространяемыми библиотеками, которые можно было бы легко подстроить в рамках решаемой задачи.

Именно это и подтолкнуло меня к решению задач, поставленных в рамках настоящей работы.

1 Постановка задачи

Цель работы:

1. Проанализировать и сравнить различные методы трилатерации, используемые при решении задачии навигации внутри помещений:
   (a) метод, основанный на поиске области пересечения сфер;
   (b) метод, основанный на поиске силового центра;
   (c) продвинутый геометрический алгоритм;
   (d) метод, основанный на фильтрации частиц;
   (e) метод, основанный на сравнении отпечатков измерений.
2. Разработать библиотеку, содержащую реализации наиболее перспективных и гибких алгоритмов из перечисленных выше, выбранных на основе предварительного анализа.

К разрабатываемому проекту выдвинуты следующие требования:

1. Поддержка различных методов трилатерации;
2. Работа алгоритмов должна быть максимально оптимизирована как по скорости вычислений, так и по объему занимаемой памяти;
3. Библиотека должна легко подключаться в проект при помощи менеджеров управления зависимостей CocoaPods или Carthage;
4. Первоначальная настройка библиотеки в проекте должна содержать минимум кода и быть интуитивно понятной.

2 Глоссарий

LBS (“Location Based Service”) – сервис, основанный на местоположении.

GPS (“Global Positioning System”) – глобальная система позиционирования.

LBA (“Location Based Advertising”) – геоконтекстная реклама.

RFID (“Radio Frequency Identification”) – радиочастотная идентификация.

NFC (“Near Field Communication”) – ближняя бесконтактная связь.

BLE (“Bluetooth Low Energy”) – беспроводная технология Bluetooth с низким энергопотреблением.

SDK (“Standard Developer Kit”) – стандартный набор разработчика.

API (“Application Programming Interface”) – интерфейс программирования приложений.

CMS (“Content Management System”) – система управления содержимым.

REST (“Representational State Transfer”) – передача репрезентативного состояния.

BVH (“Bounding Volumes Hierarchy”) – иерархия ограничивающих объемов.

DoS (“Denial of Service”) – отказ в обслуживании.

AGA (“Adaptive Geometric Algorithm”) – адаптивный геометрический алгоритм.

Спуффинг маячков (“beacon spoofing”) – тип хакерской атаки, заключающийся в выставлении поддельного маячка, настроенного с параметрами маяков некоторой существующей группы маяков.

Пиггибэкинг маячков (“beacon piggybacking”) – тип хакерской атаки, основанный на использовании параметров маячков, определенных в одном приложении, в стороннем приложении, принадлежащем третьим лицам. Сам же термин “piggybacking” с английского можно перевести как «несанкционированное проникновение вслед за зарегистрированным пользователем».

Fingerprints (с англ. буквально «отпечатки пальцев») – способ нахождения положения пользователя на основе сравнения ряда измерений расстояний до маяков с набором эталонных измерений, определенных для известных локаций.

Фильтр частиц (“particle filter”) – метод трилатерации, в котором в процессе генерации частицы (точки в двумерном пространстве) все лучше приближают истиное положение пользователя, а наименее значимые исключаются из рассмотрения.

3 Анализ задачи

В случае спутниковой навигации (GPS/Глонасс) существуют сервисы Out-Door, благодаря которым пользователь может узнать о ближайших кафе, ресторанах, гостиницах и других местах благодаря тому, что известно его текущее местоположение. А благодаря сервисам indoor-навигации возможно без проблем и оперативно найти ближайшую стойку регистрации в здании аэропорта, экспонат в музее, отдел и полку с нужным вам товаром в магазине, свободное место на парковке, и многое другое.

В качестве наиболее значимых примеров можно перечислить следующие:

- *Розничная торговля.* Установив маячки, становится возможным приветствовать клиента, сообщать ему о новых поступлениях и предложениях, когда тот находиться рядом, а заодно узнать, как он движется внутри заведения и у каких товаров проводит больше времени, - верный ключ к оптимизации для маркетологов.

- *Городской туризм.* Оказавшись рядом с объектом достопримечательности, пользователь получает информацию о месте. Этот же механизм поможет туристам получать уведомления о прибытии интересующего их транспорта к указанному месту.

- *Управление кадрами.* Используя механизм маячков, становится легко следить за тем, сколько времени тратит в офисе каждый из сотрудников, если установить пару датчиков на вход. Также возможно быстро организовывать собрания: достаточно разослать оповещение каждому, кто не находится на рабочем месте.

- *Фитнес-услуги.* Новички могут получать информацию о предназначении того или иного тренажера, а также инструкцию по его правильному использованию. Кроме того, для любого посетителя может оказаться полезной индивидуальная программа тренировок, которая бы не только замеряла время на подход и перерывы в занятиях, но и указала бы, где найти тренажер, наиболее эффективный для следующего этапа тренировок.

- *Ресторанный бизнес.* В данной сфере оригинальной идеей будет выделение уникальных посетителей заведения: это не только информирует владельца ресторана, но и позволит разработать систему персональных скидок и условий. К примеру, клиент-завсегдатай может сформировать свой заказ заранее, и, как только он войдет в заведение, заказ автоматически поступит на кухню.

- *Дом.* Bluetooth-маячки способны расширить функциональность умного дома или заменить многие сенсоры. В способы применения входят, например, умное освещение или блокирование входной двери. В данной сфере развивается, например, команда "airfy Beacon"[1].

- *Гостиницы.* Маячок на ресепшне улавливает гостей и сразу позволяет отобразить информацию о постоянных посетителях с их историей пребывания, что в итоге уменьшает время заселения. Такой системой пользуется сервис Mahana [2].

- *Концертные площадки и стадионы.* Поклонники музыкантов и спортсменов получают подробную справочную информацию и статистику. Желающие могут приобрести билеты на будущие события. С нового сезона так работает приложение Главной лиги бейсбола США, на основных стадионах которой поставили маячки [3].

- *Дейтинг.* Маячки в барах улавливают людей с поддерживаемыми приложениями. Если посетители хотят с кем-нибудь познакомиться, то получат уведомление, через которое смогут увидеть профили тех, кто находится поблизости. Функцией уже вовсю пользуются конкуренты Tinder — например, Mingleton [4].

- *Школы и университеты.* Преподавателям не нужно отмечать, кто из школьников и студентов ходит на занятия: система сама учитывает всех пришедших и, если необходимо, оповещает родителей. Кроме того, специальное приложение может предлагать учащимся вакансии, делить аудитории, высылая в разные части разный контент. Также в приложении может быть встроенный мессенджер и специальная кнопка для

застенчивых студентов, которые стесняются задать вопрос на занятии. Уже есть приложения, которые справляются с некоторыми этими задачами, например, BeHere.

- *Фестивали и конференции.* Посетителям высылают карту и расписание, справочную информацию о тех, кто выступает, и предлагают подойти к стендам. Крупнейшие площадки уже опробовали технологию: на SXSW с помощью iBeacon устраивали опросы, обсуждения среди участников и помогали разобраться, где что находится, а на кинофестивале «Трайбека» зрители могли сориентироваться, когда и где покажут нужный им фильм. Необычнее же всего технологию использовали на Каннском кинофестивале. Официальное приложение показывало, кто из гостей где находится, а также отображало ссылку на профиль каждого в LinkedIn — так было удобнее налаживать бизнес-контакты.

Благодаря большим коммерческим перспективам, направление indoor-навигации становится всё более востребованным и уже привлекло внимание таких крупных игроков на рынке, как Google, Apple, Qualcomm, Broadcom, Sony и другие, и в это, без сомнения, перспективное направление уже инвестируются сотни миллионов долларов [5].

Всемирно известная компания PayPal обозначила свой курс развития, как «инвестиции в мобильные технологии, а также возрождение розничной торговли с помощью, опять же, технологий» [6].

Разрабатываемое ими устройство – PayPal Beacon – как раз реализованный шаг в данном направлении. Оно представляет собой портативный считыватель дебетовый и кредитных карт, по функциональности не отличающийся от тех, что зачастую установлены на кассах. С помощью него, кроме непосредственно оплаты, возможно и локационное взаимодействие: местоположение пользователя является связующим звеном, помогающим определить, что он находится в конкретном заведении, и мгновенно предложить авторизацию для совершения дальнейших покупок.

3.1 Анализ примеров успешного внедрения

Международная платежная система MasterCard, Мультимедиа Арт Музей Москва, агентства Digitalizm и Insight ONE объявляют о запуске интерактивного гида «MasterCard Бесценные города – Твой МАММ» - первого в России интерактивного гида по музею, разработанного с применением технологии iBeacon. Таким образом проект решает современными инновационными средствами образовательные и маркетинговые задачи в пространстве музея [7].

Крупная сеть аптек Rite Aid установила маячки в более чем 4500 торговых точках, воспользовавшись услугами компании inMarket. inMarket знаменит услугами по изготовлению и установке Bluetooth-маячков собственного, защищенного от атак хакеров, формата [8].

В другом материале InMarket описывает кейс американского производителя продуктов питания Hillshire Brands, который хотел повысить продажи определенного сорта выпускаемых им сосисок. Разработав и внедрив приложение, работающее на основе iBeacon, компания получила увеличение узнаваемости марки сосисок на 36% и рост общего объема продаж в 10 американских магазинах, где были установлены маячки iBeacon [9].

Zatarain - первая в мире компания по производству фасованных потребительских товаров, которая внедрила технологию iBeacon. В своих магазинах компания в определенных контрольных точках установила маячки iBeacon. Оказываясь рядом с такими точками, посетители получали push-уведомления, которые мотивировали покупателей искать продукты Zatarain в магазине. Люди сканировали продукты бренда и получали за это призовые очки. Покупатели взаимодействовали с брендированнной мобильной страницей, держа в руках сам продукт от Zatarain. В итоге, люди, которые получали iBeacon-уведомления от компании, в 5 раз чаще использовали ее приложение, что соответственно привело к большому вовлечению потребителей в магазине. Оказалось, что посетители магазина, которые взаимодействовали с продуктом на контрольных точках, в 130 раз охотнее покупали его, чем остальные. Для сравнения - обычный мобильный баннер увеличивает намерение делать покупки в среднем на 3% [10].

Известный продавец обуви Timberland тестировал технологию iBeacon от Swirl в двух своих магазинах в Нью-Йорке и Бостоне. Результаты получились следующими: пользователи мобильного приложения просматривали 72% рекламных предложений от общего количества приходящих на их смартфоны, а 35% из них купили рекламируемый товар. Около 750 покупателей получили 20% скидку от Timberland [11].

"Virgin Atlantic" и аэропорт «Хитроу» тестируют сервис информирования авиапассажиров на основе маяков [12]. В Московском аэропорту «Шереметьево» данная система уже частично в ходу, инженерам потребуется около 7500 устройств, чтобы покрыть площадь в 500 тысяч квадратных метров. Следом за «Шереметьево» к использованию iBeacon-навигации должны подключиться аэропорты «Домодедово» и «Внуково» [13].

Свидетельством этого является следующий факт: один из крупнейших мировых магазинов "Macy's"планируют установить в залах своих торговых центров около 4000 таких устройств [14].

В прошлом году на Российском Интернет Форуме «РИФ+КИБ» в здании было установлено 200 маячков, и всем посетителям была наглядно продемонстрирована система навигации внутри помещения [15].

Выводы о внедрении: Компания - производитель маячков Swirl провела исследование, результаты которого показали, что пользователи мобильных приложений, в первую очередь (80%), желают получать push-уведомления о скидках. По данным того же исследования люди специально будут отключать возможность получения push-уведомлений, если в них не будет полезной информации (то есть тех же акций и скидок) или сообщения будут не релеванты их интересам или месторасположению.

Подтверждают эти выводы и исследования ювелирного магазина Alex&Ani, внедрившего в свою работу iBeacon технологии. Вот его результаты:

- 93% опрашиваемых покупателей согласилось с тем, что мобильные рекламные предложения, которые они получают в магазине увеличивают вероятность их покупки.

- 90% утверждают, что мобильные предложения заставляют их чаще посещать магазин.

- 88% сказали, что опыт пребывания в магазине становится более интуитивным и легким.

Согласно исследованию Google, 84% посетителей пользуются мобильными устройствами, находясь в магазинах, причём 50% проводят в них не менее 15 минут [16].

По данным аналитики, сегодня во время визита в магазин в 2 раза больше покупателей обращается за информацией о товарах и скидках к мобильному приложению, нежели к продавцу-консультанту. Мобильные решения сегодня полностью меняют картину маркетинговой активности потребителей [17].

3.2 Анализ существующих платформ

В рамках рассматриваемой работы были проанализированы главные существующие платформы, способные без программирования строить iBeacon-приложения, а также выделены их главные особенности.

1. Indoors (http://indoo.rs/). Платформа для indoor-навигации. Поставляется с инструментами для моделирования пространства, расстановки маяков в нем, калибровки и так далее. Демонстрационное приложение позволяет увидеть результаты моделирования.

2. LabWerk (http://labwerk.com). Основной продукт - решение для музеев mApp. Присутствует CMS для создания статей об объектах музея и настройки правил уведомления. Возможно создание опросов, привязанных к определенным регионам в музее. Поддерживается многоязычность. О посетителях собирается статистика. Приложение выводит статьи об объектах искусства при приближении к ним. Есть карта музея, на которой отображается текущее местоположение посетителя.

3. ShopJoy (http://shopjoy.se). Доступна возможность таргетировать объявления (представимых в виде push-нотификаций) по возрасту, полу и интересам пользователя, указанным при регистрации в приложении.

4. LocalSocial (https://www.mylocalsocial.com). При помощи CMS можно настроить географические зоны, промо-акции и спец-предложения для

них, зарегистрировать маяки и привязать их к определенным регионам, настроить push-уведомления при входе в регионы. Также реализована система лояльности, позволяющая зарабатывать очки лояльности и выдавать промо-коды.

3.3 Выбор типа маячков

На рынке маячков уже десятки компаний предлагают собственные продукты, обладающие различными преимуществами и недостатками. Основными критериями для сравнения маячков являются:

- **Базовая технология**, то есть непосредственно канал передачи. В основном это BLE, но возможны и другие варианты, в том числе и сочетание нескольких.
- **Совместимость**. Apple выдвинул собственный стандарт iBeacon, который не является единым и унифицированным, поэтому производитель вправе создать собственный стандарт передачи.
- **Параметры маячка**: мощность, частота обновления и другие.
- **Настройка и перенастройка**: прошивка маячков, их параметры; как производитель предполагает изменение этих параметров разработчиками.
- **Платформа и сервисы**. Сюда можно отнести такие характеристики, как SDK, единое управление, CMS и другие.
- **Безопасность**. Наличие шифрования, защиты от DoS-атак и прочие аспекты.

Среди основных производителей можно выделить следующие:

Estimote. Используют стандарт iBeacon через канал BLE. Особенностью является встроенный сенсор температуры и акселерометр.

Gimbal. Помимо iBeacon, используют собственный стандарт передачи. Обязательным требованием является регистрация приложений и маячков на

базе Gimbal SDK. Платформа для предприятий удовлетворяет спецификации REST.

StickNFind. Использует стандарт iBeacon, предлагает варианты его расширения. Опционально шифрует пакеты. Возможность получения температуры и состояния батареи. Звуковая и световая сигнализация.

3.4 Физическая реализация маячка

В плане физической реализации Beacon-маячки – это обычные Bluetooth 4.0 LE (Low Energy) устройства, таким образом, их роль может с успехом выполнять любое устройство, оснащённое BLE-чипом - например, смартфоны на базе Android, а также iPhone, iPad, обычные ноутбуки, Raspberry Pi с usb bluetooth-донглом (электронным ключом) и т.д., на которое установлено специальное приложение, реализующее функции Beacon-маячка [18].

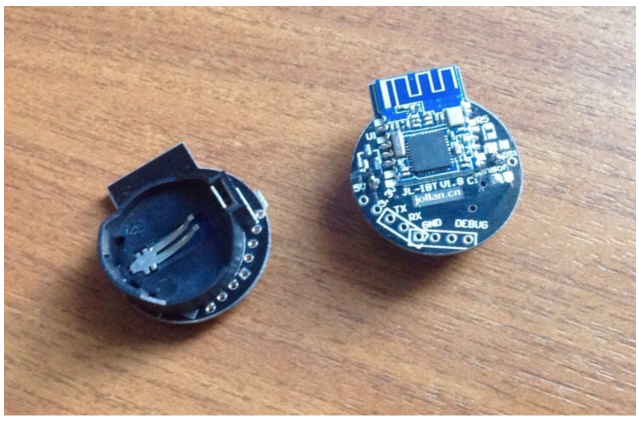


Рис. 1: Маячки, использованные в работе

Типичный маячок, показанный на рисунке выше, имеет довольно компактные размеры, и способен проработать всего лишь от одной батарейки до двух лет. Схемотехнически состоит из батарейки и Soc (System-On-Chip) Texas Instruments CC2540/2541 (также применяется Nordic nRF51822), представляющий собой 8051 микроконтроллер, в который загружается прошивка для реализации функции маячка и периферийный модуль Bluetooth LE. Дальность действия маячка в среднем 10 метров (варьируется от 15-20см до 25-40м в зависимости от модели и настроек). Периодичность выдачи данных - 200мс, но это, опять же, настраивается: можно настроить и на более частую

периодичность, и на более редкую. В рассматриваемой работе маячки были настроены на периодичность в 1с. Срок службы от одной батарейки в зависимости от модели - от чуть менее одного года до трёх лет (в среднем 2 года). Цена одного маячка составляет порядка 15-20 долларов. Маячок является простым устройством, который только выдаёт всем подряд в эфир свои данные (в advertising-режиме), используя Bluetooth-профиль GATT (при этом к нему даже не нужно выполнять подключение), тем не менее, производители, как правило, закладывают возможность подключения к маячку с целью его удалённого конфигурирования (редактирование данных, выдаваемых в эфир, периодичность выдачи данных и мощность излучения), а так же решения вопросов безопасности.

3.5 Анализ формата входных данных

С заданной периодичностью, циклически маячок выдает один и тот же набор данных [18, 19]:

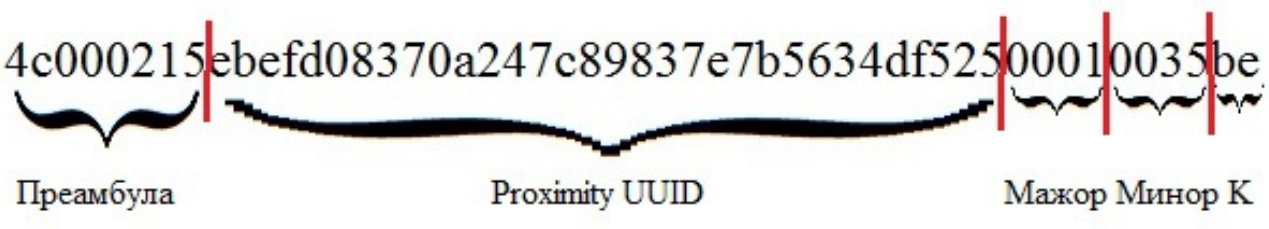


Рис. 2: Пример пакета, передаваемого маячком

Преамбула (4 байта) - префикс пакета, позволяющий установить, что мы имеем дело именно с маячком. Всегда равен `4c000215`. Преамбула состоит из 4-х полей: идентификатор компании (2 байта, в данном примере - `4c00`), тип (1 байт, в примере - `0x02`) и длина данных (1 байт, значение - `0x15`).

Proximity UUID (16 байт) – Идентификатор группы маяков. Например, если существуют несколько торговых залов, в которых требуется разместить маяки, то во всех этих залах они будут иметь один и тот же UUID, указанный при конфигурации, и это позволит отличать маяки от других, посторонних.

Мажор (2 байта) – позволяет различать небольшой набор маяков внутри одной группы. То есть внутри одной большой группы маяков, идентифи-

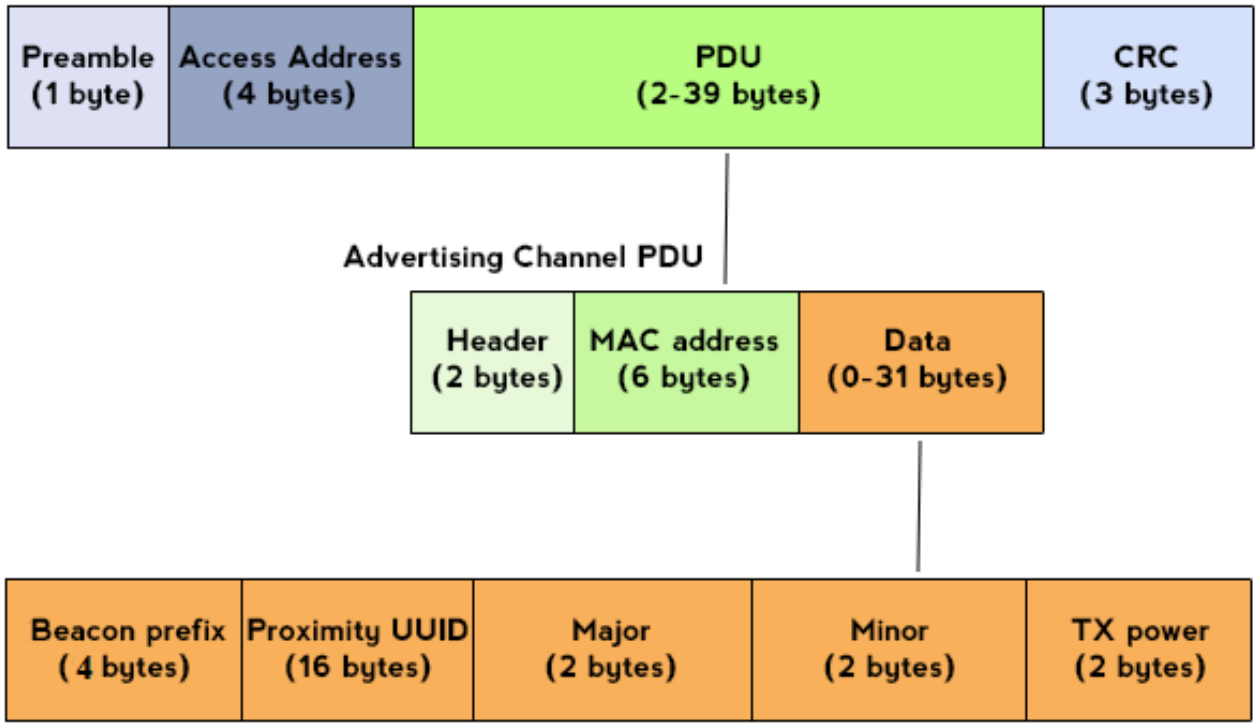


Рис. 3: Общая структура Bluetooth-пакета

цируемой UUID, может быть несколько подгрупп, каждая из которых идентифицируется по номеру мажора. Например, согласно приведенному выше примеру, каждому залу можно присвоить свой номер мажора. Если маяками требуется охватить несколько этажей здания - обычно с каждым этажом ассоциируют свой номер мажора.

Минор (2 байта) – номер, идентифицирующий сам маяк внутри мажора. Связка uuid+мажор+минор позволяет нам однозначно идентифицировать маяк и по этим данным определить координату самого маячка (обычно используется таблица соответствия маячка и его коорднат).

TX Power (параметр K на рисунке выше, 2 байта) – эталонное значение мощности маячка, представляющее собой силу сигнала на расстоянии в 1 метр от маячка. Данный параметр, как следствие, является RSSI-параметром с наложенными на него граничными условиями. TX Power измеряется и записывается в маячок единственный раз при его изготовлении. Данная константа используется при определения расстояния от пользователя до маячка. Первый бит является знаковым (1 - «−», 0 - «+»). Например, TX Power в нашем примере (см. рисунок выше, параметр «K») – `0xBE`. Это 190 в

десятичной системе счисления. Тогда эталонная сила сигнала на расстоянии 1м от маячка составляет $256 - 190 = -66\text{dBm}$.

3.6 Анализ альтернативных технологий для навигации внутри помещений

1. *RFID* – способ автоматичесткой идендификации объектов, в котором посредством радиосигналов считываются или записываются данные, хранящиеся в так называемых транспондерах, или RFID-метках. По своей функциональности, RFID, как методы сбора информации, очень близки к штрих-кодам, наиболее широко применяемым сегодня для маркировки товаров. Следовательно, RFID хорошо подойдет, например, для отслеживания посылки по большому транспортному узлу, стоит лишь снабдить весь маршрут этой посылки и ее саму RFID-метками. То есть RFID применяется там, где система знает про все. Маячки представляют собой передатчики, и мы не ориентируемся на наличие некой глобальной информации о всем состоянии, есть только информация о локальном состоянии.

2. *NFC* - технология беспроводной высокочастотной связи малого радиуса действия. Она предоставляет возможность обмена данными между устройствами, находящимися на расстоянии около 10 см. Вместе с существующей инфраструктурой бесконтактных карт, технология нацелена на использование в платежных системах и общественном транспорте. Надо заметить, что в своей сфере она обладает некоторыми важными преимуществами: NFC-метки потребляют мало энергии и легко встраиваются.

   Следует сделать замечание, что сам принцип работы технологий RFID и NFC не совсем подходит для решения круга задач, перечисленных ранее. На практике или их сложно встроить, ведь их требуется много (RFID), или радиус действия слишком мал, как в случае NFC.

3. *Навигация по Wi-Fi.* Используется уже существующая инфраструктура сетей связи – точки беспроводных сетей Wi-Fi, и это наименее затрат-

ный вариант. Методика определение координат следующая: устройство пользователя сканирует доступные Wi-Fi-точки доступа, затем информацию о них отправляет на сервер, где эти данные по базе данных сопоставляются с координатами этих точек доступа, по которым и вычисляются координаты пользователя. К сожалению, координаты Wi-Fi точек точно не известны, плюс могут меняться (перенесли Wi-Fi точку в другое место или заменили её на другую – координаты уже оказываются неверными).

Точность при таком подходе оставляет желать лучшего (погрешность - до 25 метров! При использовании специально созданной Wi-Fi инфраструктуры точность достигает 3-5 метров, но это уже требует ощутимых затрат на создание и обслуживание подобной системы), да и идентифицировать клиентов по Wi-Fi, привязывая их расположение к карте помещений, проблематично: начиная с iOS 8, mac-адреса Apple-устройств (iPhone, iPad) постоянно меняются для предотвращения «рекламной» слежки.

4. *Геомагнитное позиционирование.* Основано на ориентировании по магнитному полю Земли и базируется на геомагнитных аномалиях как критериях для геомагнитного позиционирования (аномалии возникают вследствии неоднородности геомагнитного поля). Заключается в фиксации геомагнитных аномалий и нанесении их на карту территории, на которой предполагается ориентироваться. В дальнейшем навигация производится по составленной карте устройством, в которое встроен магнитометр. Практический пример реализации – система IndoorAtlas команды учёных из финского университета Оулу.

   Недостаток – высокая сложность реализации, невысокая точность. В помещениях очень много динамически меняющихся магнитных аномалий (проводка, поле в которой меняется в зависимости от подключённой нагрузки и сильно меняет конфигурацию магнитного поля вокруг себя; посетители со своими радиоэлектронными устройствами; стеллажи; тележки), сильно усложняющих навигацию, основанную на указанном способе ориентировании в пространстве.

5. *Системы спутниковой навигации (GPS/Глонасс и другие) + инерциальные навигационные системы (ИНС).* Применимо, когда периодически появляется сигнал систем спутниковой навигации. Его точности по мнению современных исследователей достаточна, чтобы вычислить положение пользователя вплоть до десятка сантиметров [20]! Однако когда пользователь въезжает в тоннель, ещё доступны актуальные координаты и направление движения с GPS/Глонасс-спутников, далее в самом тоннеле сигнал теряется, и поэтому используется уже инерциальная навигационная система (ИНС, основанная на базе акселерометра, гироскопа, магнитометра), которая использует в качестве начальных условий последние актуальные данные с GPS/Глонасс до потери связи со спутником и поддерживает их актуальность на основе получаемых с датчиков данных о текущей скорости, ускорении и направлении движения до возобновления связи со спутниками.
   Стоит принимать во внимание, что в ИНС ошибки постоянно накапливаются, и со временем данные, полученные с ИНС, становятся все более и более отличными от действительности.

6. *Ориентирование по базовым станциям операторов сотовой связи (GSM).* В зоне видимости сотового телефона/GSM-модема постоянно находятся как минимум одна базовая станция GSM, а обычно их несколько. Координаты расположения этих базовых станций известны (благодаря многочисленным навигационным сервисам (например «Яндекс.Навигатор»), приложение получает информацию о видимых мобильным телефоном базовых станций и текущем положении по GSM/«Глонасс», и отправляет эти сведения в «Яндекс», где на основе этих данных строится база соответствий «Базовая станция-координаты», к которой имеется свободный доступ через предоставляемое API). Далее в модем отправляется команда "$AT + CREG = 2$", в результате чего можно получить сообщения +CREG: с информацией о текущей подключенной базовой станции: LAC и CELLID (соответственно код зоны и идентификатор базовой станции). Отправив эти данные на один из специальных сервисов (предоставляемый «Яндекс», "Google"и другими компа-

ниями), возможно определить координаты этой базовой станции. Многие модемы позволяют получить список видимых базовых станций (БС) с указанием их LAC и CELLID - остаётся только через базы данных с координатами БС получить их координаты и методом триангуляции определить примерное местоположение пользователя.
Минусы: невысокая точность (БС может быть удалена на расстоянии в 35км от пользователя, некоторые БС являются мобильными и постоянно меняют свою дислокацию).

7. *Использование Bluetooth-маячков iBeacon* даёт достаточную точность при приемлемом уровне финансовых затрат; перспективная технология, которая активно развивается, поэтому именно iBeacon был выбран в качестве предмета данной работы.

8. *Навигация, основанная на синергетическом эффекте* решает задачу определения текущего местоположения, используя все (или большинство) из перечисленных выше способов. Эффективность достигается за счёт того, что используется сразу несколько векторов определения координат, что способствует компенсации ошибок и повышению точности определения координат. На реализацию подобной системы в прошлом году фондом развития центра разработки и коммерциализации новых технологий «Сколково» был выделен грант в 1 млн долларов [18].

3.7 Анализ безопасности

Не стоит переоценивать данную технологию и забывать о таком важном аспекте, как безопасность. На текущий момент развития трафик в процессе передачи никаким образом не шифруется, поэтому злоумышленник потенциально может провести атаку.

Выделяются 2 главных типов атак: спуффинг и пиггибэкинг [21].

- *Beacon Spoofing.* Атака основана на включении маячка или устройства, работающего как маячок, с параметрами, идентичными параметрам некоторой группы маячков. Это может быть использовано для того,

чтобы симулировать некоторое событие или оповещение в месте, отличном от предполагаемого в контексте приложения.
Пример угрозы: предположим, есть некоторый магазин, использующий маячки для приветствия новых посетителей. Злоумышленник копирует настройки маячков, и впоследствие это позволяет ему разослать приветственное уведомление пользователям, которые находятся в абсолютно другой локации. Это может спровоцировать недоумение и, в конечном итоге, неудовлетворенность программным продуктом.

- *Beacon Piggybacking, или Hijacking.* Атака основана на применении уже существующих маячков, предназначенных для некоторого приложения, в своем, стороннем приложении. Это может быть использовано для получении аналитики, основанной на оригинальный маячках, а также в рассылке сообщений и симуляции событий, не относящихся к предполагаемым.
Пример угрозы: пусть кофейня «А» конкурирует с кофейней «Б». «А» начинает использовать приложение с использованием маячков. В ответ кофейня «Б» разрабатывает свое приложение с информацией об идентификаторах чужих маячков. В результате каждый раз, как пользователь с установленным приложением от кофейни «Б» заходит в кофейню «А», на его устройство приходит оповещение о скидках на кофе в «Б».

Неприятный инцидент произошел на выставке потребительской электронике (Consumer Electronics Show – CES) в 2014 году. Для всех желающих была организована «охота за сокровищами», в рамках которой через приложение пользователи должны были определить место, где якобы зарыт сундук с кладом. Однако еще до начала мероприятия неизвестная группа хакеров взяла apk-файл приложения, проанализировала его структуру через декомпилятор, и смогла извлечь параметры всех используемых маячков. Это давало всю необходимую информацию, и победить таким способом можно было даже не выходя из дома [22].

Выделяют 4 способа защиты:

1. **Геолокационная проверка**. После получения оповещения от одного из новых (в рамках текущей сессии) маячков, устройство использует

геолокационный сервис, чтобы убедиться, что маячок физически действительно находится поблизости.

2. **Идентификация на основе начального значения (seed)**. Используются маячки с периодически меняющимся UUID. Алгоритм смены, в свою очередь, основан на некотором цифровом значении, хранимом отдельно. Через единый SDK происходит обновление, идентификация и синхронизация всего процесса. Маячок, не прошедший проверку – потенциальный вредитель – будет выкинут из рассмотрения.

3. **Облачное подтверждение**. Способ базируется на основе предыдущего, но роль связующего звена на себя берет облачный сервис, а не локальный SDK. Этот механизм уже используется в маячках компании Estimote, и известен под названием "UUID Rotation". Секретный ключ, определяющий смену идентификаторов, хранится на платформе Estimote Cloud.

4. **Управление на уровне аппаратных средств**. Начальные параметры и их возможное обновление берут на себя аппаратные средства – контроллеры. При этом оповещение об обновлении приходит от облачного сервиса отдельно на контроллеры и отдельно на устройство пользователя. После этого контроллеры обновляют UUID маяков. В дальнейшем проверка маячков будет происходить на устройстве без использования сервиса.

Преимущества и недостатки перечисленных методов представлены в таблицах 1 и 2, соответственно.

Способ защиты	**Геолокационная проверка**	**Идентификация на основе генерации случайных чисел**	**Облачное подтверждение**	**Управление на уровне аппаратных средств**
Преимущества	1. Наиболее дешевый способ 2. Простой в конфигурации и поддержке 3. Может быть легко включен или выключен в любой момент 4. Позволяет использовать традиционный iBeacon-формат для лучшей совместимости	1. Позволяет защититься от перечисленных типов атак 2. Нет привязки к геолокации или Интернет-соединению 3. Не требует дополнительного оборудования	1. Позволяет в большей степени защититься от перечисленнных способов атак 2. Не требуется подключение дополнительных устройств 3. Усложняет процедуру проведения атаки для злоумышленника	1. Надежный уровень защиты 2. Предоставляет устройство для обновления и обслуживания маячков 3. Изменения легко применить к существующей системе 4. Позволяет использовать традиционный iBeacon-формат для лучшей совместимости

Способ защиты	**Геолокационная проверка**	**Идентификация на основе генерации случайных чисел**	**Облачное подтверждение**	**Управление на уровне аппаратных средств**
Недостатки	1. Не защищает против piggybacking-атак 2. Геолокационная относительность не дает такой же точности, как остальные методы 3. Определение локации может являться причиной задержки в начале сессии	1. Сложен в модификации в случае неполадок 2. UUID маячков могут быть легко определены злоумышленником 3. Более сложное развертывание 4. Формат использования не поддерживается в приложениях Apple	1. Сложен в модификации в случае неполадок 2. Сложен в развертывании 3. Задержки в использовании могут отразиться на удобстве использовании приложения 4. Предполагает наличие Интернет-соединения 5. Формат использования не поддерживается в приложениях Apple	1. Наиболее дорогой тип размещения 2. Требует наличия дополнительного набора устройств 3. Не работает для маячков, расположенных на удалении от остальных 4. Предполагает наличие Интернет-соединения (периодически)

Примечательно, что Apple iOS SDK на уровне реализации не позволяет приложению сканировать эфир на обнаружение BLE-пакетов. Обязательным условием является явная конфигурация UUID, major и minor-идентификаторов [23].

3.8 Выбор языка и технологий программирования

Objective-C - компилируемый объектно-ориентированный язык программирования, используемый корпорацией Apple, построенный на основе языка Си и парадигм Smalltalk. В частности, объектная модель построена в стиле Smalltalk — то есть объектам посылаются сообщения.

Язык Objective-C является надмножеством языка C, поэтому C-код полностью понятен компилятору Objective-C.

Компилятор Objective-C входит в GCC и доступен на большинстве основных платформ. Язык используется в первую очередь для Mac OS X (Cocoa) и GNUstep — реализаций объектно-ориентированного интерфейса OpenStep. Также язык используется для iOS (Cocoa Touch).

Swift - мультипарадигменный объектно-ориентированный язык программирования, созданный компанией Apple для разработчиков iOS и OS X. Swift работает с библиотеками Cocoa и Cocoa Touch и совместим с основной кодовой базой Apple, написанной на Objective-C. Swift задумывался как более безопасный язык в сравнении с Objective-C. Язык поддерживается в среде программирования Xcode; программы на нем компилируются при помощи Apple LLVM и используют рантайм Objective-C, что делает возможным использование обоих языков (а также чистого C и C++) в рамках одной программы.

Для реализации библиотеки был выбран язык Objective-C, так как язык хорошо поддерживается стандартной IDE “XCode”, отлично документирован и обладает широким сообществом программистов. Кроме того, в Objective-C лучшая по сравнению со Swift поддержка вставок кода на C, а именно на нем написаны алгоритмы трилатерации, рассматриваемые в данной работе.

3.9 Анализ предоставляемого API

Для indoor-навигации Apple предоставляет библиотеку CoreLocation. Она содержит несколько основных сущностей, использованных в работе:

- `CLBeacon`. Класс, инкапсулирующий как идентификаторы маяка (uuid, major, minor), так и информацию, используемую при ранжировании (rssi, accuracy, proximity).

- `CLBeaconRegion`. Класс, содержащий идендификаторы группы маяков (uuid, [major, [minor]]). С его помощью можно удобно разделять группы маячков, находящихся, например, на разных этажах, и работать с каждой группой по отдельности.

- `CLLocationManager`. Центральный класс API, который может отслеживать маячки в указанном регионе (`CLBeaconRegion`), рассылая событиями как факты входа или выхода в зону действия маячков (*monitoring*), так и прием пакетов для определения расстояния до источника (*ranging*).

Таким образом, Apple представила не только формат на BLE-маячки iBeacon, ставший уже широко распространенным, но и удобные средста для работы с ними.

3.10 Анализ лиейного фильтра Калмана

В рамках исследуемой работы возникает небходимость фильтровать некоторые данные. Прежде всего, к ним относятся значения RSSI, получаемые от маячков.

Фильтр Калмана был выбран на основании нескольких положений [24]:

1. Фильтр имеет рекуррентную форму, благодаря чему он удобен для программирования. Кроме того, так как новые оценки формируются на основе старых, нет необходимости хранить весь массив наблюдений, что экономит память и время работы.

2. Алгоритм фильтрации одновременно представляет собой непосредственное описание способа реализации фильтра.

3. Его легко распространить на нестационарные сигналы; это относится и к случаю, когда наблюдения начинаются в произвольный момент времени.

4. Легко распространяется на многомерный случай.

К тому же, скользящее среднее при большем размере окна (обозачим размер за N) будет отражать не актуальные данные, а лишь усреднение за последние N измерений. Калман же при большем доверии новому значению больше приближает показание к последнему измеренному, при этом не исключая корректировки выбросов.

Фильтр Калмана использует динамическую модель системы (например, физический закон движения), известные управляющие воздействия и множество последовательных измерений для формирования оптимальной оценки состояния. Алгоритм состоит из двух повторяющихся фаз: *предсказание* и *корректировка*.

На первом рассчитывается предсказание состояния в следующий момент времени (с учетом неточности их измерения). На втором новая информация с датчика корректирует предсказанное значение (также с учетом неточности и зашумленности этой информации). Важно отметить, что задача фильтрации – получить наиболее близкое значение к реальной величине.

Рассмотрим алгоритм фильтра в общем случае для следующих имеющихся данных:

$$\begin{cases} x_{k+1} = x_k + u_k + \xi_k & (1) \\ z_k = x_k + \eta_k & (2) \end{cases}$$

где x_k – измеряемая величина;
u_k – член, отвечающий за контроль системы извне (закон, по которому изменяется величина x);
ξ_k – ошибка модели, то есть разница между реальным значением и рассчитанным;
η_k – ошибка, образовавшаяся в результате измерения сенсором;
z_k – полученные данные с сенсора;
x_{k+1} – новое значение величины на очередном шаге итерации.

Задача состоит в том, что, зная неверные показания сенсора z_k, найти хорошее приближение для истинного значения x_k. Это хорошее приближение мы будем обозначать как x_k^{opt}.

Будем рассуждать по индукции. Представим, что мы уже нашли отфильтрованное значение с сенсора x_k^{opt}. Можно предположить, что на шаге $k+1$ система эволюционирует согласно закону 1, и сенсор покажет значение, близкое к $x_k^{opt} + u_k$. С другой стороны, зная соотношение 2, мы будем знать неточное значение сенсора z_{k+1}.

Идея Калмана состоит в том, что чтобы получить наилучшее приближение к истинной координате x_{k+1}, мы должны выбрать золотую середину между показанием неточного сенсора и $x_k^{opt} + u_k$ — нашим предсказанием того, что мы ожидали от него увидеть. Показанию сенсора мы дадим вес K а на предсказанное значение останется вес $(1 - K)$:

$$x_{k+1}^{opt} = K z_{k+1} + (1 - K)(x_k^{opt} + u_k)$$

Коэффициент K называют коэффициентом Калмана.

Согласно выводам самого Калмана, коэффициент имеет свойство стабилизироваться после небольшого числа итераций [25].

Алгоритм обладает достаточной гибкостью: он позволяет задать априорную информацию о характере системы, связи переменных, и на основании этого строить более точную оценку. Кроме того, даже в случае отсутствия априорной информации фильтр дает отличные результаты [26, 27].

3.11 Анализ оптимального расположения маячков в помещении

Мы имеем начальную систему уравнений, основанную на методе трилатерации:

$$\begin{cases} (x_0 - x_1)^2 + (y_0 - y_1)^2 = r_1^2 \\ (x_0 - x_2)^2 + (y_0 - y_2)^2 = r_2^2 \\ (x_0 - x_3)^2 + (y_0 - y_3)^2 = r_3^2 \end{cases}$$

Однако данная система не может дать однозначного решения в виде точки, если, например, общая область пересечения кругов не является точкой.

Назовем в таком случае область такого общего пересечения областью ошибки локализации. Введем также обозначение: пусть величина ошибки варьируется в интервале $(-\epsilon, \epsilon)$. Используя это, определим:

$$C_{p_i} = \{(x; y) \in R^2 | (x-x_i)^2+(y-y_i)^2 \leq (r_i+\epsilon_i)^2, (x-x_i)^2+(y-y_i)^2 \geq (r_i-\epsilon_i)^2\}$$

В приведенном выше определении можно заметить, что $\epsilon_i = 0, \cap_i C_{p_i}$ образует единственную точку. В случае, когда $\epsilon_i > 0, \cap_i C_{p_i}$ образует область с площадью, отличной от нуля.

В исследовании [28] математически доказано, что данная площадь будет минимальна (учитывая, что величина ϵ мала), если маячки установлены симметрично. То есть в случае трилатерации оптимальным вариантом будет расположение их в узлах равностороннего треугольника, а дальнейшая установка новых устройств для покрытия большей площади будет выполнена, как на рисунке 4.

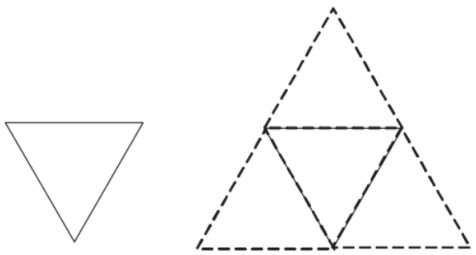


Рис. 4: Оптимальное расположение маячков

Расположение в узлах квадратов дает несколько худший результат (ошибка в среднем больше на 5,3%), а случайное расположение маячков увеличивает ошибку на 34,9% [29].

3.12 Анализ задачи выбора оптимального набора маячков

Существует целое семейство алгоритмов, называемое “OASIS” – Optimum Anchor Selection algorIthmS, что в переводе и значит «выбор оптимального набора маячков». Для работы с GPS существует функционал, называемый $GDOP$ – Geometric Dilution Of Precision, то есть «геометрическое ослабление

точности». Данная величина отражает, насколько на текущий момент взятое созвездие спутников вляет на результирующую точность.

Вычислить GDOP возможно по следующей формуле:

$$GDOP(M, \phi) = \sqrt{\frac{M}{\sum_i \sum_{j,j>i} |\sin(\phi_{ij})|^2}} \tag{3}$$

где M – количество спутников, рассматриваемых в рамках нахождения текущей локации, а ϕ – угол между каждой из пар спутников. При этом чтобы вычислить 3 для отдельно взятой точки, необходимо знать ее положение. В приложениях, использующих GPS, начальное грубое вычисление исходной позиции может быть допущено ввиду больших дистанций между пользователем и спутниками. В рамках задачи определения локации внутри помещений такой подход неприемлем и не может быть использован без каких-либо изменений.

Рассмотрим альтернативные методы OASIS:

1. **Совместная кластеризация**. Подход метода заключается в выборе k лучших по силе сигнала (RSSI) маячка для дальнейшего вычисления координат.

2. **Простая выпуклая оболочка**. В данном методе рассматриваются дистанции, определенные при анализе величины сигнала от каждого из маячков $n_1, n_2, ..., n_N$. Фунционал выпуклой оболочки, обозначаемый C, может быть вычислен по формуле:

$$C \equiv \{\sum_{j=1}^{N} \lambda_j n_j : \lambda_j \geq 0\}, \sum_{j=1}^{N} \lambda_j = 1$$

3. **Сила сигнала**. Используемая в данной работе техника,заключающаяся в выборе трех маячков с наиболее сильным сигналом вне зависимости от их расположения.

4. **Выбор, основанный на определении площади**. Согласно исследованию, проведенному М.Мирононовой и Х.Халвисом [30], большая площадь, сформированная тремя спутниками, образует лучшее (то есть

меньшее) значение $GDOP$. Применяя это к поставленной задаче, мы перебираем все возможные треугольники, образованные тройками маячков, выбирая наибольший по площади из них.

5. **Периметр**. Метод достаточно схож по своей сути с приведенным выше, но при рассмотрении треугольника учитывается его периметр. Периметр также показывает линейную независимость координат образовавших его маячков. С другой стороны, использование периметра бесполезно в рамках некоторых сценариев (например, нахождение трех маячков на одной прямой).

6. **Плотность**. Алгоритм, рассматривая все комбинации маячков, вычисляет все возможные координаты пользователя. После этого область с наибольшей плотностью возможных координат пользователя выбирается в качестве результата, а конечной точкой обычно берется центроид промежуточных результатов.

7. **Мультилатерация**. Если рассматривать сразу N маячков, то возможно составить систему уравнений для вычисления положения пользователя, учитывая показания каждого из них.
   Положим, что координата узла i есть b_i. Необходимо вычислить положение пользователя p. Показания маячков равны d_i, соответственно. В идеальной ситуации мы имеем:

$$||b_i - p||_2^2 = d_i^2$$

   Перепишем левую часть в виде

$$||b||_2^2 + ||p||_2^2 - 2b_i^T p = d_i^2 \tag{4}$$

   Учитывая, что всего участвуют в расмотрении M маячков, снова возможно переписать выражение в следующем виде для вычисления позиции пользователя:

$$p = 0.5B^+(a + \gamma) \tag{5}$$

   при этом мы воспользовались заменой $\gamma = ||p||_2^2$ и $[a]_i = ||b||_2^2 - d_i^2$, а также вектор-столбец $B = [p_1, p_2, ..., p_M]^T$. Нотация B^+ означает псевдообращение Мура-Пенроуза. Чтобы решить 5, необходимо вычислить

γ:

$$\gamma = p^T p = [0.5B^+(a+\gamma)]^T[0.5B^+(a+\gamma)]$$

В нашем случае мы выбираем γ, которая бы давала наименьшую среднюю ошибку между каждым из значений $p1, p2, ..., p_M$ и искомой позицией (см. 4).

В работе [31], показано, что в рамках проводимого ряда экспериментов, наименьшая средняя ошибка достигается при использовании методов, основанных на выборе по силе сигнала, совместной кластеризации и мультилатерации.

Важно отметить, что те методы, которые не использовали RSSI-показатель сигнала явно (как например основанные на площади или периметре), давали худшие результаты.

Наконец, при увеличении вовлеченных в процесс вычисления маячков, разные подходы адаптировались лучше или хуже. Так, мультилатерация, несмотря на свою точность, значительно замедлялась в работе из-за решения сверхдетерминированной системы уравнений.

Руководствуясь вышеприведенными рассуждениями, я выбрал метод, основанный на использовании трех маячков с наилучшими сигнал-шумовыми показателями.

4 Реализация

Реализованная библиотека находится в открытом доступе и расположена в репозитории по веб-адресу:

`https://github.com/EnlightenedCSF/IBeaconLocation`

4.1 Реализация алгоритма, основанного на сравнении локационных отпечатков

Локационные «отпечатки пальцев» – набор признаков, присвоенных некоторой заведомо известной точке пространства [32]. В рамках задачи будем считать, что этим набором признаком является вектор из эталонных показателей RSSI, принятых от каждого из маячков.

Смысл алгоритма тогда будет состоять в следующем: пользователь в некоторый момент времени определяет ряд значений RSSI. Этот набор сравнивается с остальными с целью нахождения ближайшего «отпечатка», значения которого отличается минимально в рамках принятой метрики. И так как положение каждого из «отпечатка» заранее известно, становится возможным определить положение пользователя.

Данную задачу можно рассматривать как задачу классификации. Так, каждый отпечаток FP_i в точке пространства $i \quad (i = \overline{1, N})$ может быть представлен как $FP_i = \{RSSI_1^{FP_i}, RSSI_2^{FP_i}, ..., RSSI_M^{FP_i}\}$, и соответствующая ему локация есть $L_i^{FP} = (X_i, Y_i)$.

Если представить положение пользователя как T, то ему сопоставим отпечаток $S = \{RSSI_1^T, RSSI_2^T, ..., RSSI_M^T\}$. После этого возможно найти дистанцию между S и FP_i:

$$D_{FP_i}^T = \sqrt{\sum_{j=1}^{N}(RSSI_j^{FP_i} - RSSI_j^T)^2} \tag{6}$$

Будем считать M количеством маячков, участвующих в рассмотрении. Теперь на основе расстояния можем вычислить вероятность того, что объект

находится около отпечатка FP_i с измерениями S:

$$P(FP_i|S) = \frac{1}{D^T_{FP_i}} \tag{7}$$

Тогда правило определения локации можно сформулировать так: *искомая точка – i – если $P(FP_i|S) > P(FP_j|S)$, где $i, j = \overline{1, N}$. Ей соответствует локация L^{FP}_i.*

В реальности искомая позиция пользователя не является дискретной, и поэтому возникает необходимость интерполяции. Эта процедура осуществляется с помощью следующего преобразования, представленного ниже:

$$LT(x, y) = \sum_{i=1}^{N} P(FP_i|S) L^{FP}_i(x, y)$$

где (x, y) - координаты FP.

В работе [32] предложено дополнение для данного подхода, называемое Budgeted Dynamic Exclusion (BDE, "бюджетное динамическое исключение"). Суть модификации сводится к следующему: сравниваются показания j и отпечаток i. Если некоторая пара значений RSSI, соответствующих одному и тому же маяку, меньше установленной константы, именуемой бюджетом, то эти два значения выкидываются из вектора с остальными значениями RSSI. Это приводит к тому, что оцениваемое формулой 6 Евклидово расстояние уменьшается, и, соответственно, степень доверия, вычисляемая с помощью формулы 7, увеличивается.

Главным препятствием в работе такой системы является дифракция, отражение и рассеяние сигнала, принимаемого от маяков [33]. Кроме того, возникают сложности, связанные с выходом из строя маячков. Ведь в таком случае необходимо либо переснимать показания сигнал-шума от оставшихся маячков заново, либо предусмотреть дополнительную логику в приложении и работе алгоритма в целом. Оба способа достаточно затратны по времени.

Хотя данный метод нахождения пользовательской локации был использован, например, на конференции GeekPicnic, проведенного в Москве в 2015г. [34], в итоговую версию библиотеки, разработанной в рамках данной работы, алгоритм не включен.

4.2 Реализация алгоритма локализации по Монте-Карло

Метод также известен как «локализация с помощью фильтра частиц».

В рамках данного метода алгоритм генерирует сотни частиц, представляющих возможные будущие положения пользователя. Вычисляются расстояния до маяков и учитываются значения их сигналов. На основании этих актуальных показаний и несоответствия предсказаний этим данным, часть частиц фильтруется, то есть выходит из рассмотрения, а оставшаяся часть участвует в дальнейших вычислениях. Это происходит в результате обновления весов, присвоенных каждой из частиц.

Проблему локализации можно сформулировать как проблему определения апостериорной вероятности $p(x_k|z_{1:k})$. Начальное распределение $p(x_0)$ считается равномерным по всем возможным локациям $[x \;\; y]^T$.

Рассматривая локализацию по Монте-Карло, требуемое распределение $p(x_k|z_{1:k})$ может быть представлено в виде множества взвешенных значений:

$$S_k = \{x^i, \omega^i\}, i = 0, 1, 2, ..., N_p$$

где ω^i - вес частицы - представляет собой фактор важности, для которого выполняется равенство $\sum_{i=1}^{N_p} \omega_i = 1$.

Апостериорная вероятность может быть аппроксимирована с помощью следующего выражения:

$$p(x_k|z_k) \approx \frac{1}{N_p} \sum \delta(x_k - x_k^i)$$

где δ - дельта-функция Дирака. Для достаточно большого N_p аппроксимация отклоняется от истинного значения искомого распределения незначительно.

Алгоритм функционирует в несколько стадий:

1. **Предсказание**. На этом этапе апостериорное распределение $p(x_k|x_{k-1}, u_{k-1})$ в момент времени k предсказывается на основании состояния $p(x_{k-1}|z_{1:k-1})$ и управляющего вектора u_{k-1}. Множество частиц S_{k-1} соответствует состоянию x_{k-1}. Вектор u_{k-1} должен быть применен к каждой частице из S_{k-1}. Это дает новый набор $S'_k = x'^i, \quad \omega'^i,$ $i = 0, 1, 2, ..., N_p$. Заметим, что $\omega'^i = \omega^i$.

2. **Обновление**. На этом шаге принимается во внимание модель измерений: каждая частица из S'_k изменяет вес на основании степени схожести с $p(z_k|x^i_k), \quad i = 0, 1, 2, ..., N_p$. После этого образуется новый набор частиц S_k.

3. **Вырождение**. После нескольких итераций большинство частиц имеют незначительный вес, и лишь немногие из них вносят значительный вклад в вычисления. Это стало причиной разработок многих алгоритмов ресэмплинга, позволяющих выделить только необходимую часть частиц. Чтобы избежать накладных расходов на ресэмплинг во время каждой из итераций, зачастую вычисляется Effective Sample Size, ESS. На основании этого функционала, ресэмплинг применяется лишь в случае, когда величина ESS опускается ниже определенного порога. Ресемплинг зачастую, помима отсева незначащих частиц, дуплицирует те, значения весов которых значительны.

$$cv_t^2 = \frac{var(\omega_t^i)}{E^2(\omega_t^i)} = \frac{1}{N_p}\sum_{i=1}^{N_p}(N_p\omega^i - 1)^2$$
$$ESS_t = \frac{N_p}{(1 + cv_t^2)}$$

Однако, в рамках решения задачи локализации с помощью данного метода возникает так называемая *проблема потери частиц* (*"particle deprivation"*). Суть этой проблемы, подробно описанной в [35], сводится к тому, что алгоритм во время проведения этапа ресэмплинга может отбросить частицы, являющиеся значимыми для локализации. Это особенно значимо для вариантов реализации с небольшим ($M \leq 50$) числом частиц.

В качестве возможного, хотя и не однозначного решения возможно генерировать дополнительно некоторое количество частиц каждую итерацию работы алгоритма.

Тем не менее, из-за неустойчивости в формировании результата данным алгоритмом, было решено в рамках реализации библиотеки не включать данный метод.

4.3 Реализация алгоритма трилатерации, основанного на определении силового центра

Другим вариантом реализации метода трилатерации может быть алгоритм, предложенный бельгийскими исследователями В.Пьерло, М.Урбин-Шоффреем и М.Ван Другенброек, которые свели проблему к решению системы линейных уравнений [36].

Чтобы объяснить суть метода, прежде всего, необходимо ввести понятие о так называемом *силовом центре* ("power center" или "radial center") трех окружностей – уникальной точке, обладающей равной мощностью ("power") по отношению к этим окружностям. Мощность P точки при этом может быть найдена по формуле:

$$P_{c,p} = (x - x_c)^2 + (y - y_c)^2 - R^2 \tag{8}$$

где x, y – координаты точки, для которой вычисляется мощность, x_c, y_c – координаты центра окружности радиуса R. Согласно формуле 8, если точка лежит на окружности, мощность равна нулю; меньше нуля в случае, когда точка находится внутри окружности и больше нуля, когда находится вне окружности.

Силовой линией ("power line") двух окружностей назовем такое геометрическое место точек, обладающих равной мощностью по отношению к этим окружностям.

Она перпендикулярна прямой, соединяющей центры данных окружностей и проходит через область их взаимного пересечения (в случае наличия такового). Если рассматриваются три окружности, такие, что никакие две из них не расположены в одной точке, и центры которых не коллинеарны, то для них можно определить силовой центр – точкой пересечения силовых линий. На приведенном ниже рисунке силовой центр обозначен точкой.

Нахождение координаты точки пересечения двух прямых – задача, имеющая линейную сложность. В данном случае мы можем, учитывая формулу 8,

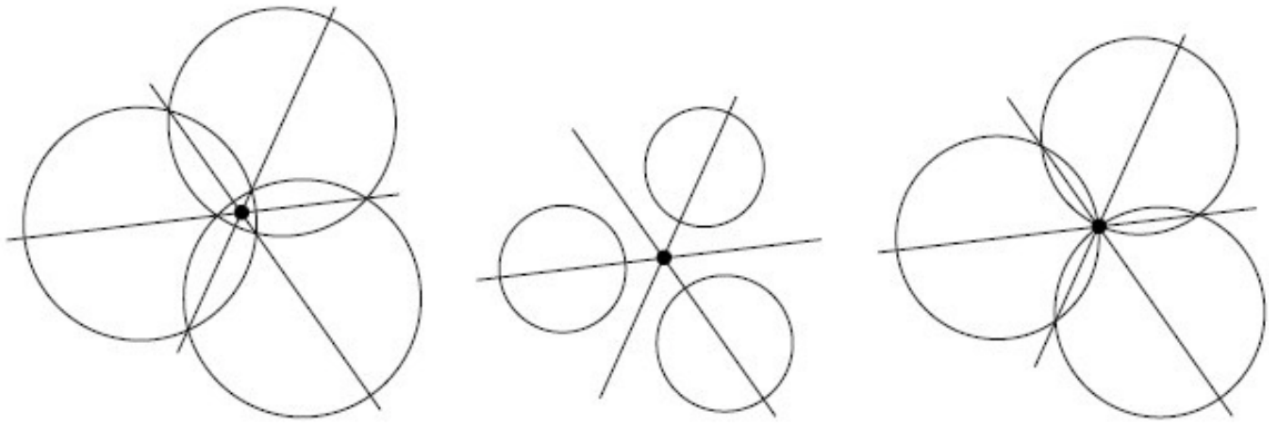


Рис. 5: Определение силового центра для окружностей различных радиусов

просто приравнять мощности.

$$
\begin{aligned}
&(x - x_1)^2 + (y - y_1)^2 - R_1^2 = (x - x_2)^2 + (y - y_2)^2 - R_2^2 \\
&\Rightarrow x(x_1 - x_2) + y(y_1 - y_2) = \frac{x_1^2 + y_1^2 - R_1^2}{2} - \frac{x_2^2 + y_2^2 - R_2^2}{2} \\
&\Rightarrow x(x_1 - x_2) + y(y_1 - y_2) = k_1 - k_2
\end{aligned}
$$

Введен дополнительный коэффициент k, равный мощности источника относительно некоторой окружности, деленный на два:

$$k_n = \frac{x_n^2 + y_n^2 - R_n^2}{2}$$

Наконец, чтобы определить координаты точки пересечения силовых линий, достаточно решить систему линейных уравнений:

$$
\begin{cases}
x(x_1 - x_2) + y(y_1 - y_2) = k_1 - k_2 \\
x(x_2 - x_3) + y(y_2 - y_3) = k_2 - k_3 \\
x(x_3 - x_1) + y(y_3 - y_1) = k_3 - k_1
\end{cases} \tag{9}
$$

Можно заметить, что одно из уравнений в системе может быть получено путем сложения двух других, что является, в свою очередь, доказательством пересечения трех силовых линий лишь в одной точке. Ее точные координаты можно определить по следующим формулам:

$$x_r = \frac{\begin{vmatrix} k_1 - k_2 & y_1 - y_2 \\ k_2 - k_3 & y_2 - y_3 \end{vmatrix}}{D}, \quad y_r = \frac{\begin{vmatrix} x_1 - x_2 & k_1 - k_2 \\ x_2 - x_3 & k_2 - k_3 \end{vmatrix}}{D}$$

$$D = \begin{vmatrix} x_1 - x_2 & y_1 - y_2 \\ x_2 - x_3 & y_2 - y_3 \end{vmatrix} = \begin{vmatrix} x1 & y1 & 1 \\ x2 & y2 & 1 \\ x3 & y3 & 1 \end{vmatrix}$$

Определитель матрицы $D \neq 0$, если центры окружностей не коллинеарны, тогда же, соответственно, существует решение приведенной системы уравнений 9. В противном случае найти координаты пересечения прямых в явном виде нельзя.

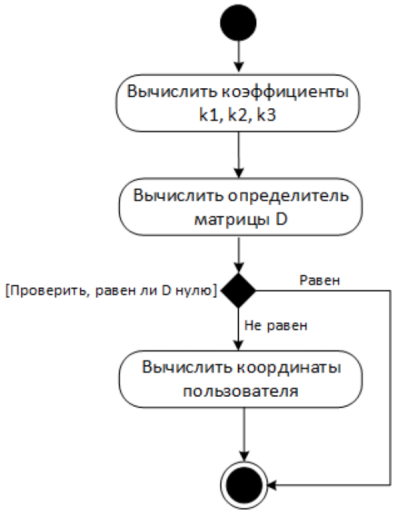


Рис. 6: Activity-диаграмма рассматриваемого алгоритма

Исходный код, реализующий данный метод, приведен в приложении 1.

4.4 Реализация алгоритма трилатерации, основанном на пересечении сфер

В геометрии трёхмерная проблема трилатерации представляет собой нахождение координат точки пересечения трёх сфер, которые определяются путём решения системы уравнений. Чтобы упростить вычисления, полагаем, что центры всех трех сфер лежат в плоскости $z = 0$, один из них совпадает с началом координат, второй — лежит на оси x. Наложенные ограничения не уменьшают общности: к такому виду может быть приведена любая система соответствующих уравнений путем перехода к другой системе координат. Чтобы найти решение в исходной системе координат, к решению, найденному

в этой (приведенной) системе координат, применяются преобразования, обратные к тем, которые позволили исходное множество из трех точек привести в соответствие с ограничениями.

С первого взляда может показаться, что задача определения координаты устройства по нескольким маячкам схожа с задачей, которая решается спутниками GPS, и, соответственно, мы тоже должны решать задачу о пересечении сфер. К тому же, маяки могут быть закреплены в помещении на разных высотах. Но на практике этим можно пренебречь, отбросив z-координату. Предпосылкой для этого является и то, что хотя маячки могут находиться на различной высоте, на одной и той же высоте находится пользователь. Таким образом, задача сводится к работе с окружностями.

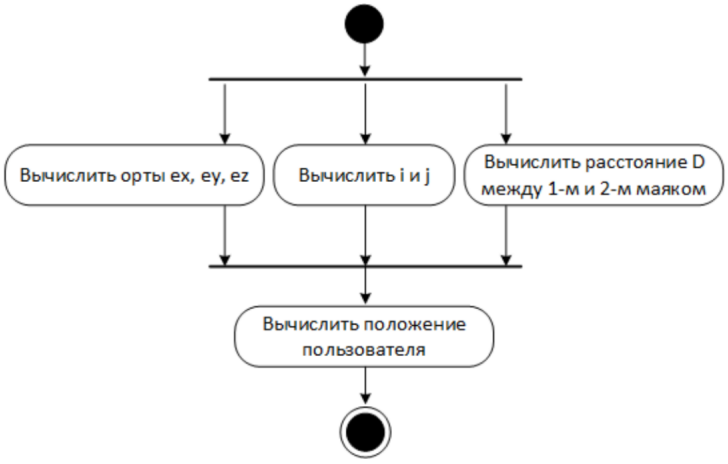


Рис. 7: Activity-диаграмма алгоритма пересечения сфер

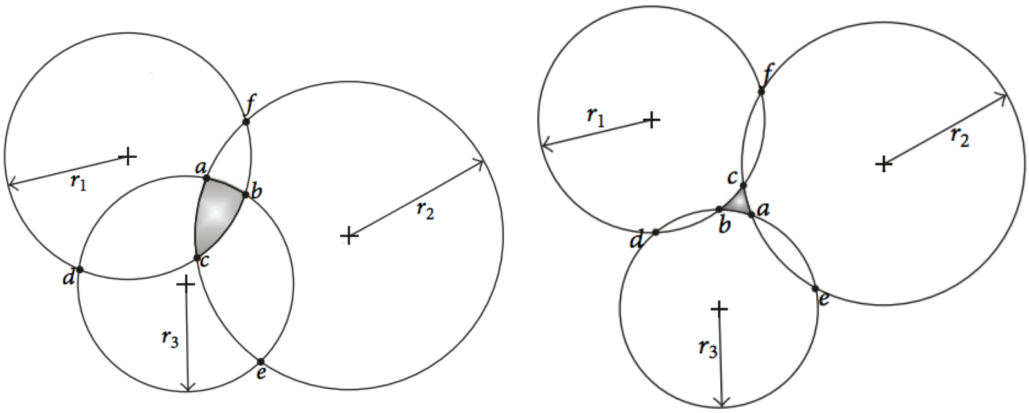
С листингом данного метода можно ознакомиться в приложении 2.

4.5 Адаптивный геометрический алгоритм

Прежде всего, следует объяснить принцип работы геометрического алгоритма.

Геометрический алгоритм опирается на взаимоположение окружностей и их радиусы, и результирующие точки пересечения имеют ключевое значение. На рисунке 8 показаны наиболее часто возникающие сценарии (точки, соответствующие пересечениям окружностей, обозначены как a, b, c, и так далее).

В случае, изображенном на рисунке 8а, нас интересуют только три точки из шести, так как они формируют область, принадлежащую всем трем



(a) (b)

Рис. 8: Наиболее часто встречающиеся сценарии при работе геометрического алгоритма

окружностям.

На рисунке 8b более сложный сценарий, ведь общей области пересечения всех трех окружностей нет. Следовательно, мы должны воспользоваться другой логикой для нахождения искомой области. Конкретнее, для каждй пары точек, образовавшейся в результате пересечения пары окружностей, нас удовлетворяет та точка, которая находится ближе к третьей окружности (например, из точек a и e будет выбрана a, так как она находится ближе к окружности с радиусом r_1).

Наконец, искомым положением пользователя является центроид, вычислимый по формуле 10.

$$x = \frac{1}{K}\sum_{l=1}^{K} x_l, \quad y = \frac{1}{K}\sum_{l=1}^{K} y_l, \quad l = 1, 2, ..., K \tag{10}$$

K в представленной выше формуле означает количество подходящих точек пересечений. Для случая с тремя окружностями $K = 3$.

В работе [37] представлена идея *адаптивного геометрического алгоритма*. В сущности, геометрический алгоритм, описанный выше, является частным случаем, а точнее - первой итерацией адаптивного геометрического алгоритма.

Благодаря данной первой итерации получен набор точек пересечения. Суть алгоритма сводится к пропорциональному увеличению или уменьше-

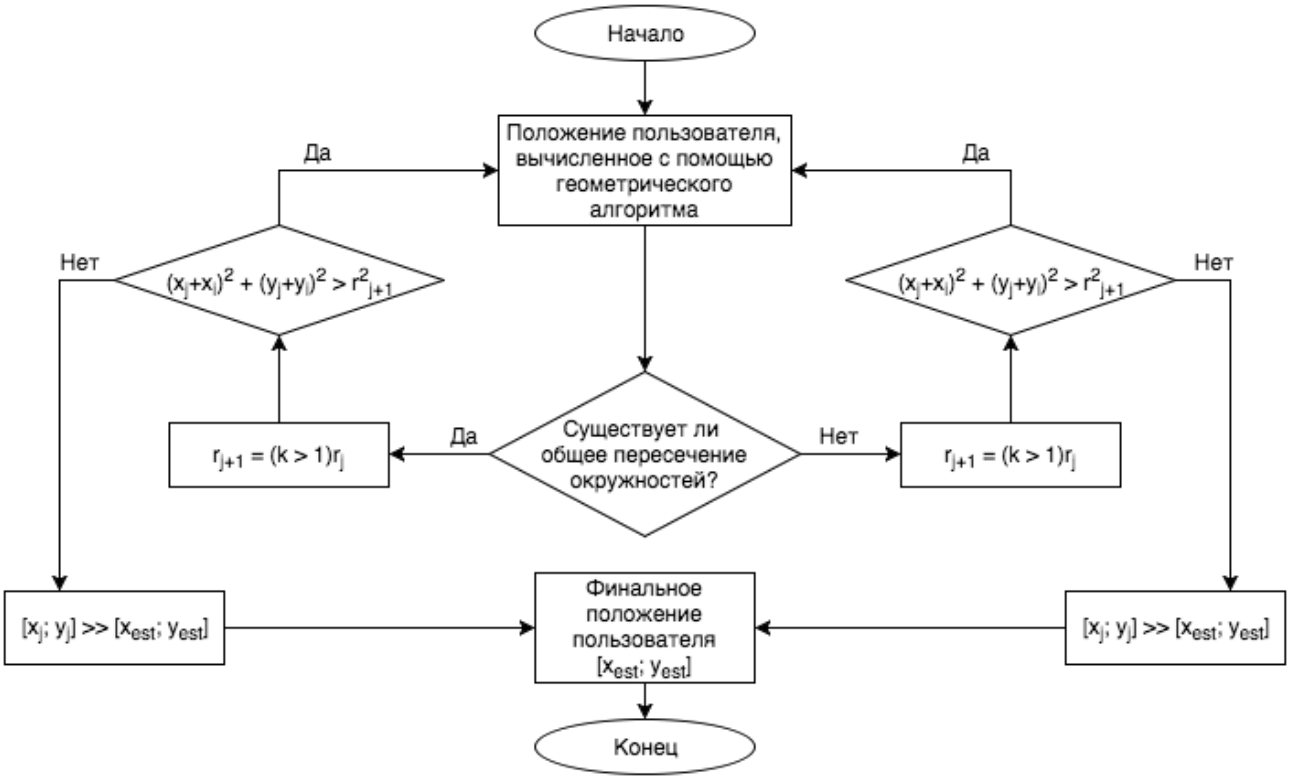


Рис. 9: Диаграмма потока данных адаптивного геометрического алгоритма

нию всех трех радиусов окружностей так, чтобы общая область пересечений (см. случай на рисунке 8a) была наименьшей. Конечный этап - нахождение положения пользователя - осуществляется по формуле 10. Алгоритм работы проиллюстрирован на рисунке 9.

В настоящей работе предложена модификация алгоритма для увеличения скорости его работы. Суть изменения сводится к следующему: для случая, показанного на рисунке 8a, вместо пропорционального уменьшения радиусов, вычисляются координаты центроида точек пересечения. Для случая, показанного на рисунке 8b, алгоритм функционирует, как описано выше.

Полный листинг конечной версии алгоритма представлен в приложении 3.

4.6 Основные классы библиотеки

4.6.1 BeaconLocation

Класс `BeaconLocation` – центральный класс библиотеки. Предполагается, что пользователь начинает работу с инстанцирования этого класса. Во

время создания `BeaconLocation` создает необходимые объекты `CoreLocation` (`CLBeaconRegion`, `CLLocationManager`), а так же подписывается на обновления данных, получаемых от маячков.

Кроме того, здесь же происходит создание классов `Processor`, который ответственен за вычисление новых координат пользователя, и `Floor`, который является хранилищем всех идендификаторов маячков, их координат и очередных показаний.

4.6.2 Floor

Класс `Floor` инкапсулирует маячки, расположенные на этаже. Также предоставляет набор методов, с помощью которых пользователь может добавить новые маячки, получить информацию об уже имеющихся и удалить указанные. Предполагается, что задающие их координаты измеряются в метрах, так как обновления значений расстояний до маячков тоже формируются в метрах самой библиотекой `CoreLocation`.

4.6.3 Processor

Класс `Processor` - ядро библиотеки. По умолчанию он подписывается на события об изменении вычисленных расстояний до маячков, и вычисляет на их основе новое положение пользователя. Пользователю предлагаются 3 готовых алгоритма, а также делегат, с помощью которого он может реализовать и использовать собственный метод. Кроме того, есть возможность использовать комбинацию из имеющихся алгоритмов для достижения лучшего результата.

Степень доверия результату каждого из алгоритмов может быть указана в параметрах метода

```
-(void)setAlgorithmsAndTrusts:(NSDictionary<NSNumber*, NSNumber*>*)algorithms
```

Переданные в паре с каждым из алгоритмов числа нормируются к единице. Пусть первый алгоритм вычислил положение пользователя как точку с координатами $(1; 1)$. Второй используемый алгоритм - как точку $(5; 5)$. По умолчанию, все алгоритмы эквивалентно важны, и общим результатом будет среднее арифметическое: точка $(3; 3)$. Если второй алгоритм вдвое более важен, чем первый, то результат будет вычислен как $0,66 * (5; 5) + 0,33 * (1; 1)$,

то есть $(3,66; 3,66)$.

4.6.4 Attractor и Spot

Для того, чтобы сделать приложение более отзывчивым во время приближения пользователя к некоторому объекту, в библиотеке предусмотрены классы `Attractor` и `Spot`.

`Spot` - класс, представляющий некоторую «горячую точку» в локационном контексте приложения, другими словами, место, в котором мы ожидаем увидеть пользователя чаще, чем в другом (витрина в магазине, экспонат в музее).

Работа с ними осуществляется через класс `Attractor`. После того, как точки добавлены, а положение пользователя вычислено, `Attractor` приближает положение пользователя к ближайшей «горячей точке», основываясь на определенным пользователем коэффициенте («сила притяжения» - `attractorPower`). Учтена и «мертвая зона», внутри которой `Attractor` перестает приближать положение пользователя к указанной отметке.

Все это позволяет создавать более дружественный интерфейс разрабатываемого приложения.

4.7 Структура библиотеки

На рисунке 10 показана структура классов разработанной библиотеки.

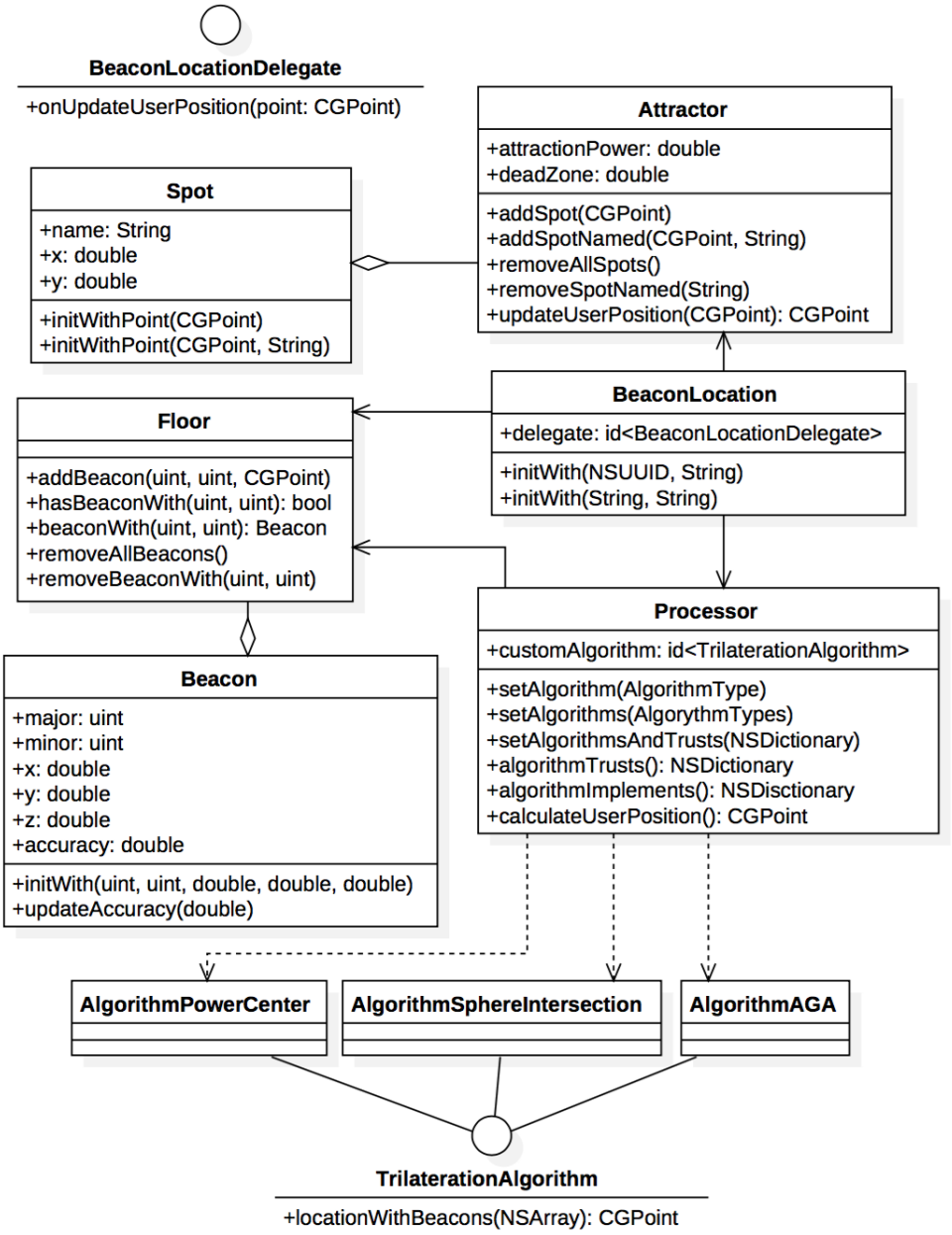


Рис. 10: Диаграмма классов разработанной библиотеки

Заключение

Список литературы

[1] *Siewart Steffen.* Airfy beacons - make your smart home even smarter. — URL: https://www.indiegogo.com/projects/airfy-beacon-make-your-smart-home-even-smarter/. — 2016. — (дата обращения: 10.05.16г.).

[2] Crunchtime! — URL: http://www.crunchtime.com/. — 2016. — (дата обращения: 10.05.16г.).

[3] Mlb.com ballpark. — URL: https://itunes.apple.com/app/id513135722?mt=8. — 2016. — (дата обращения: 10.05.16г.).

[4] Mingleton. — URL: https://itunes.apple.com/ru/app/mingleton/id807307276. — 2015. — (дата обращения: 03.05.16г.).

[5] Знакомьтесь: биконы! — Сессия-интенсив про электронные маячки, для бизнесменов и разработчиков. — URL: http://2014.secr.ru/lang/ru/program/submitted-presentations/. — 2014. — (дата обращения: 17.08.15г.).

[6] Российские программисты не спали 24 часа - интервью с Джоном Ланном из paypal. — URL: http://goo.gl/YHqQre. — 2015. — (дата обращения: 06.03.16г.).

[7] Первый проект внедрения технологии ibeacon в пространствах музея появился в России. — URL: http://www.retail-loyalty.org/news/. — 2015. — (дата обращения: 06.04.15г.).

[8] Rite aid deploys proximity beacons in more than 4,500 us stores. — URL: http://www.zdnet.com/article/rite-aid-deploys-proximity-beacons-in-more-than-4500-us-stores/. — 2016. — (дата обращения: 18.01.16г.).

[9] Hillshire increases sales w/ ibeacon, 20x increase in purchase intent, 500x increase over average mobile ads. — URL: http://9to5mac.com/2014/07/22/. — 2014. — (дата обращения: 22.07.15г.).

[10] Inmarket & zatarain's launch world first cpg ibeacon campaign. — URL: http://www.ibeacon.com/inmarket-zatarains-launch-world-first-cpg-ibeacon-campaign/. — 2014. — (дата обращения: 04.04.15г.).

[11] Timberland, kenneth cole track shoppers who opt-in for deals. — URL: http://adage.com/article/datadriven-marketing/timberland-tests-store-tracking-app-swirl-offers/243811/. — 2013. — (дата обращения: 08.05.16г.).

[12] Virgin atlantic lights the way with apple's ibeacon technology at heathrow. — URL: https://blog.virgin-atlantic.com/t5/Our-Future/Virgin-Atlantic-lights-the-way-with-Apple-s-iBeacon-technology/ba-p/26359#.VzMaUxV96Rt. — 2014. — (дата обращения: 04.07.15г.).

[13] Московские аэропорты могут начать использовать ibeacon. — URL: http://appleinsider.ru/sluxi/moskovskie-aeroporty-mogut-nachat-ispolzovat-ibeacon.html. — 2014. — (дата обращения: 05.08.15г.).

[14] Macy's takes ibeacon technology nationwide, installing more than 4,000 devices. — URL: http://marketingland.com/macys-100162. — 2014. — (дата обращения: 09.09.15г.).

[15] Одно успешное внедрение ibeacon: 200 маячков для РИФ+КИБ 2014. — URL: https://habrahabr.ru/post/225713/. — 2014. — (дата обращения: 03.07.15г.).

[16] How mobile is transforming the shopping experience in stores. — URL: https://www.thinkwithgoogle.com/research-studies/mobile-in-store.html. — 2015. — (дата обращения: 12.11.15г.).

[17] «Успешных кейсов в России - ноль» Эксперт о технологии ibeacon, которая умеет продавать в оффлайне с помощью mobile. — URL: http://www.therunet.com/experts/4749-uspeshnyh-keysov-v-rossii-nol. — 2015. — (дата обращения: 08.09.15г.).

[18] Навигация в помещениях с ibeacon и ИНС. — URL: https://habrahabr.ru/post/245325/. — 2014. — (дата обращения: 10.12.15г.).

[19] Indoor навигация с ibeacon в ios7. — URL: https://habrahabr.ru/company/touchinstinct/blog/195104/. — 2013. — (дата обращения: 04.03.15г.).

[20] Gps tracking down to the centimeter. — URL: http://phys.org/news/2016-02-gps-tracking-centimeter.html. — 2016. — (дата обращения: 21.01.16г.).

[21] Secure beacons: Overview & options. — URL: http://www.slideshare.net/localzco/beacon-security-overview. — 2015. — (дата обращения: 20.04.15г.).

[22] Hacking the ces scavenger hunt. — URL: http://makezine.com/2014/01/03/hacking-the-ces-scavenger-hunt/. — 2014. — (дата обращения: 04.04.15г.).

[23] Corebluetooth doesn't let you see beacons. — URL: http://developer.radiusnetworks.com/2013/10/21/corebluetooth-doesnt-let-you-see-ibeacons.html. — 2013. — (дата обращения: 22.03.15г.).

[24] *Эйкхофф П.* Основы идентификации систем управления. — М.: Мир, 1975. — Vol. 680.

[25] *Дусеев ВР, Рудь МН, Мальчуков АН.* ИССЛЕДОВАНИЕ ФИЛЬТРА КАЛМАНА В ПРИЛОЖЕНИИ К ФИЛЬТРАЦИИ ДАННЫХ ГЛУБИНЫ С СЕНСОРА kinect.

[26] *Давид Худавердян.* Фильтр Калмана. — URL: https://habrahabr.ru/post/166693/. — 2013. — (дата обращения: 15.02.15г.).

[27] Фильтр Калмана — Введение. — URL: https://habrahabr.ru/post/140274/. — 2012. — (дата обращения: 15.02.15г.).

[28] *Han Guangjie, Choi Deokjai, Lim Wontaek.* Reference node placement and selection algorithm based on trilateration for indoor sensor networks // *Wireless Communications and Mobile Computing.* — 2009. — Vol. 9, no. 8. — Pp. 1017–1027.

[29] *Bulusu Nirupama, Heidemann John, Estrin Deborah.* Adaptive beacon placement // Distributed Computing Systems, 2001. 21st International Conference on. / IEEE. — 2001. — Pp. 489–498.

[30] *Mironovova Martina, Havlis H.* Calculation of gdop coefficient // *CTU in Prague: Faculty of Electrical Engineering.* — 2011.

[31] Comparison of anchor selection algorithms for improvement of position estimation during the wi-fi localization process in disaster scenario / Oleksandr Artemenko, Thierry Simon, Andreas Mitschele-Thiel et al. // Local Computer Networks (LCN), 2012 IEEE 37th Conference on / IEEE. — 2012. — Pp. 44–49.

[32] *Elbes Mohammed, Al-Fuqaha Ala, Anan Muhammad.* A precise indoor localization approach based on particle filter and dynamic exclusion techniques // *Network Protocols and Algorithms.* — 2013. — Vol. 5, no. 2. — Pp. 50–71.

[33] Survey of wireless indoor positioning techniques and systems / Hui Liu, Houshang Darabi, Pat Banerjee, Jing Liu // *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on.* — 2007. — Vol. 37, no. 6. — Pp. 1067–1080.

[34] *Бабаев Александр.* ibeacon. Мифы и реальность. — URL: `https://habrahabr.ru/post/278689/`. — 2016. — (дата обращения: 04.03.16г.).

[35] Robust monte carlo localization for mobile robots / Sebastian Thrun, Dieter Fox, Wolfram Burgard, Frank Dellaert // *Artificial intelligence.* — 2001. — Vol. 128, no. 1. — Pp. 99–141.

[36] *Pierlot Vincent, Urbin-Choffray Maxime, Van Droogenbroeck Marc.* A new three object triangulation algorithm based on the power center of three

circles // Research and Education in Robotics-EUROBOT 2011. — Springer, 2011. — Pp. 248–262.

[37] *Brida Peter, Machaj Juraj.* A novel enhanced positioning trilateration algorithm implemented for medical implant in-body localization // *International Journal of Antennas and Propagation.* — 2013. — Vol. 2013.

Приложения

Приложение 1. Реализация алгоритма, основанного на поиске силового центра

```
double * calculateUserPositionPowerCenter(double *xs, double *ys, double *
      accs) {

    double *result = (double *)malloc(sizeof(double) * DIMENSIONS);

    // k[n] = ( B[n].x^2 + B[n].y^2 - dist[n]^2 )/2
    double *ks = (double *)malloc(sizeof(double) * MIN_BEACONS);
    for (size_t i = 0; i < MIN_BEACONS; ++i) {
        ks[i] = (xs[i]*xs[i] + ys[i]*ys[i] - accs[i]*accs[i])/2.0;
    }

    // D = | x1-x2  y1-y2 | = (x1-x2)*(y2-y3) - (y1-y2)*(x2-x3)
    //     | x2-x3  y2-y3 |
    double d = (xs[0] - xs[1])*(ys[1] - ys[2]) - (ys[0] - ys[1])*(xs[1] - xs
          [2]);

    if (fabs(d) < 1e-7) {
        result[0] = -1;
        return result;
    }

    // X = | k1-k2  y1-y2 | = (k1-k2)*(y2-y3) - (y1-y2)*(k2-k3)
    //     | k2-k3  y2-y3 |
    double x = (ks[0] - ks[1])*(ys[1] - ys[2]) - (ys[0] - ys[1])*(ks[1] - ks
          [2]);

    // Y = | x1-x2  k1-k2 | = (x1-x2)*(k2-k3) - (k1-k2)*(x2-x3)
    //     | x2-x3  k2-k3 |
    double y = (xs[0] - xs[1])*(ks[1] - ks[2]) - (ks[0] - ks[1])*(xs[1] - xs
          [2]);

    result[0] = x/d;
    result[1] = y/d;

    return result;
}
```

Приложение 2. Реализация алгоритма, основанного на пересечении сфер

```
double * calculateUserPositionSphereIntersection(double *xs, double *ys,
      double *accs) {
    double *result = (double *)malloc(sizeof(double) * DIMENSIONS);

    double temp = (xs[1] - xs[0])*(xs[1] - xs[0]) + (ys[1] - ys[0])*(ys[1] -
          ys[0]);
    double exx = (xs[1] - xs[0]) / sqrt(temp);
    double exy = (ys[1] - ys[0]) / sqrt(temp);

    double p3p1x = xs[2] - xs[0];
    double p3p1y = ys[2] - xs[0];

    double ival = exx * p3p1x + exy*p3p1y;

    double p3p1i = (xs[2] - xs[0] - exx)*(xs[2] - xs[0] - exx) + (ys[2] - ys
          [0] - exy)*(ys[2] - ys[0] - exy);

    double eyx = (xs[2] - xs[0] - exx) / sqrt(p3p1i);
    double eyy = (ys[2] - ys[0] - exy) / sqrt(p3p1i);

    double d = sqrt(temp);

    double jval = (eyx * p3p1x) + (eyy * p3p1y);

    double xval = (accs[0]*accs[0] - accs[1]*accs[1] + d*d) / (2*d);
    double yval = (accs[0]*accs[0] - accs[2]*accs[2] + ival*ival + jval*jval
          )/(2*jval) - (ival/jval)*xval;

    result[0] = xs[0] + exx*xval + eyx*yval;
    result[1] = ys[0] + exy*xval + eyy*yval;

    return result;
}
```

Приложение 3. Реализация адаптивного геометрического алгоритма

```
#define MAX_ITERATIONS  500
#define Points          struct point *
#define enoughPoints    7
#define step            0.1f
```

```
/**
 *  Struct represents a beacon or a point
 */
struct point {
    double x;
    double y;
    double r;
};

/**
 *  Checks whether the point belongs to all three circles
 *
 *  @param x      x
 *  @param y      y
 *  @param points Beacons
 *
 *  @return 1 if true, 0 otherwise
 */
int isPointBelongToAllCircles(double x, double y, Points points) {
    int belongs = 1;
    for (size_t i = 0; i < MIN_BEACONS; ++i) {
        double dist = getDistance(x, y, points[i].x, points[i].y);
        if (dist > (points[i].r + 1e-2)) {
            belongs = 0;
            break;
        }
    }
    return belongs;
}

/**
 *  Returns the centroid coordinate of given points
 *
 *  @param points Array of points
 *  @param size   Its size
 *
 *  @return Centroid coordinates
 */
Points getCenter(Points points, size_t size) {
    Points center = (Points )malloc(sizeof(struct point));
    center->x = 0;
    center->y = 0;

    for (size_t i = 0; i < size; ++i) {
        center->x += points[i].x;
```

```
        center->y += points[i].y;
    }
    center->x /= size;
    center->y /= size;

    return center;
}

/**
 *  Calculates all circle-circle intersections and return an array of
        resulting point
 *
 *  @param pointA Circle A
 *  @param pointB Circle B
 *  @param cnt    Reference to the resulting array's size
 *
 *  @return Array of points
 */
Points getCircleCircleIntersection(Points pointA, Points pointB, int *cnt) {
    *cnt = 0;
    double r1 = pointA->r;
    double r2 = pointB->r;
    double p1x = pointA->x;
    double p1y = pointA->y;
    double p2x = pointB->x;
    double p2y = pointB->y;
    double d = getDistance(p1x, p1y, p2x, p2y);

    // if too far away, or self contained - can't be done
    if ((d >= (r1 + r2)) || (d <= fabs(r1 - r2))) {
        return 0;
    }

    double a = (r1*r1 - r2*r2 + d*d)/(2*d);
    double h = sqrt(r1*r1 - a*a);
    double x0 = p1x + a*(p2x - p1x)/d;
    double y0 = p1y + a*(p2y - p1y)/d;
    double rx = -(p2y - p1y)*(h/d);
    double ry = -(p2x - p1x)*(h/d);

    *cnt = 2;
    Points result = (Points )malloc(*cnt * sizeof(struct point));
    result[0].x = x0 + rx;
    result[0].y = y0 - ry;
    result[1].x = x0 - rx;
    result[1].y = y0 + ry;
```

```
    return result;
}

/**
 *  Returns an array of circle intersections
 *
 *  @param points Beacons
 *  @param size   Size of the array
 *  @param cnt    Reference to the size of the resulting array
 *
 *  @return Array of points
 */
Points getIntersectionPoints(Points points, size_t size, int *cnt) {
    size_t e = 20;
    Points result = (Points )malloc(e * sizeof(struct point));

    size_t indexToAdd = 0;
    for (size_t i = 0; i < size; ++i) {
        for (size_t j = 1; j < size; ++j) {
            int cnt = 0;
            Points intersects = getCircleCircleIntersection(&points[i], &
                  points[j], &cnt);
            if (cnt > 0) {
                for (size_t k = 0; k < cnt; ++k) {
                    result[indexToAdd++] = intersects[k];
                }
            }
        }
    }

    return result;
}

/**
 *  Selects only common points from the circles intersection
 *
 *  @param points Intersection points
 *  @param size   Size of that array *  @param beacons The given beacons and
       their accuracies
 *  @param cnt    Reference to the resulting array's count
 *
 *  @return Array of points
 */
Points getCommonPoints(Points points, size_t size, Points beacons, int *cnt)
      {
    Points result = (Points)malloc(sizeof(struct point) * enoughPoints);
```

```
    int k = 0;
    for (size_t i = 0; i < size; ++i) {
        if (!isPointBelongToAllCircles(points[i].x, points[i].y, beacons)) {
            continue;
        }

        result[k].x = points[i].x;
        result[k].y = points[i].y;
        result[k].r = points[i].r;
        ++k;
    }
    *cnt = k;

    return result;
}

/**
 *  Mutates three arrays of coordinates into one array of points
 *
 *  @param xs   xs
 *  @param ys   ys
 *  @param accs accs
 *
 *  @return Array of points
 */
Points createPoints(double *xs, double *ys, double *accs) {
    Points points = (Points )malloc(MIN_BEACONS * sizeof(struct point));
    for (size_t i = 0; i < MIN_BEACONS; ++i) {
        points[i].x = xs[i];
        points[i].y = ys[i];
        points[i].r = accs[i];
    }
    return points;
}

/**
 *  Enlarges the beacons' accuracies proportionally
 *
 *  @param beacons Array of beacons
 */
void enlargeAccuracies(Points beacons) {
    for (size_t i = 0; i < MIN_BEACONS; ++i) {
        beacons[i].r *= (1 + step);
    }
}
```

```
//*************************************************************************

double * calculateUserPositionEpta(double *xs, double *ys, double *accs) {
    double *result = (double *)malloc(sizeof(double) * DIMENSIONS);

    Points beacons = createPoints(xs, ys, accs);

    int iterations = 0;
    while (1) {
        if (++iterations > MAX_ITERATIONS) {
            return result;
        }

        int intersectionCount = 0;
        Points intersections = getIntersectionPoints(beacons, MIN_BEACONS, &
              intersectionCount);

        if (intersectionCount == 0) {
            return result;
        }

        int commonCount = 0;
        Points common = getCommonPoints(intersections, intersectionCount,
              beacons, &commonCount);

        if (commonCount == 2 || commonCount == 3) {
            Points center = getCenter(common, commonCount);
            result[0] = center[0].x;
            result[1] = center[0].y;
            return result;
        }

        enlargeAccuracies(beacons);
    }

    return result;
}
```