Федеральное агентство по образованию
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет информатики
Кафедра теоретических основ информатики

УДК 681.03 513

ДОПУСТИТЬ К ЗАЩИТЕ В ГАК
Зав. кафедрой, проф., д.т.н.
_____ Ю.Л. Костюк
«___» _____ 2006 г.

Пешехонов Сергей Васильевич

ВЫДЕЛЕНИЕ ФРАГМЕНТОВ ПОВЕРХНОСТЕЙ ТРЁХМЕРНЫХ ТЕЛ ПО ДАННЫМ ЛАЗЕРНОГО ЗОНДИРОВАНИЯ С ИСПОЛЬЗОВАНИЕМ ПОВЕРХНОСТНОЙ ТРИАНГУЛЯЦИИ

Дипломная работа

Научный руководитель,
проф., д.т.н. Ю.Л. Костюк

Исполнитель,
студ. гр. 1411 С.В. Пешехонов

Электронная версия дипломной работы помещена
в электронную библиотеку. Файл
Администратор

Томск – 2006

Реферат

Дипломная работа с. 45, рис.12, табл. 2, источников 15, прил. 1

ВЫЧИСЛИТЕЛЬНАЯ ГЕОМЕТРИЯ, ЛАЗЕРНОЕ ЗОНДИРОВАНИЕ, ПОВЕРХНОСТНАЯ ТРИАНГУЛЯЦИЯ, МЕТОД НАИМЕНЬШИХ КВАДРАТОВ, РАСПОЗНАВАНИЕ ПРОСТРАНСТВЕННЫХ ФИГУР, VISUAL STUDIO, MFC, STL, OPENGL, C++

Объект исследования – алгоритмы выделения трехмерных объектов по поверхностной триангуляции.

Цель работы – разработка и реализация алгоритма для распознавания простейших трехмерных фигур по поверхностной триангуляции.

Область применения – обработка данных, полученных в результате наземного или воздушного лазерного зондирования.

Метод исследования – теоретическое исследование и практическая реализация.

Результаты работы – разработан алгоритм выделения кусочно-плоскостных объектов, обладающий линейной трудоёмкостью. А также общий алгоритм выделения поверхностей сфер, цилиндров, конусов и торов, на основе таких объектов. Создано программное приложение, основанное на алгоритме выделения поверхностей трёхмерных тел.

Содержание

Введение

В последнее время всё чаще можно встретить компьютеры в самых разнообразных областях науки и техники. Многие процессы, которые ранее требовали большого разнообразного человеческого труда, автоматизируются. Вместе с тем, богатый опыт и информация, накопленные за многие годы требуют нового представления, удобного для компьютера. Так происходит, например, в картографии, где по старым снимкам строят цифровые модели рельефа, пригодные для будущего использования. Но «оцифровка» такого рода информации зачастую представляется довольно сложной задачей. Поскольку требует труда человека по сверке карт, по их сканированию. К тому же, если карты были получены, например, в разное время, то они содержат различные данные. Первоначально, такие различия должен обработать опять же человек. И, наконец, эти данные должны быть обработаны компьютером, что тоже является довольно трудоёмким процессом, особенно, если понадобится помощь эксперта для разрешения каких-либо неподдающихся распознаванию фрагментов.

Принципиально иным подходом является метод лазерного зондирования. Этот метод получает данные об объектах сразу в трехмерных координатах. После чего по такому множеству точек строится трехмерная модель. Принципиальное отличие такого подхода в том, что результат может быть получен на порядок быстрее, чем при фотограмметрии, к тому же, этот результат требует меньшего труда человека и является более точным [1]. Лазерное сканирование является молодой отраслью - производство сканирующих лазерных систем в мире можно назвать штучным. Алгоритмы, применяемые в программном обеспечении для обработки данных лазерного сканирования, ещё не достаточно хорошо разработаны. Однако интерес к системам лазерного зондирования возрастает.

Существует множество алгоритмов для выделения объектов по облаку точек. К таким, например, относятся алгоритмы, которые строят статистическую модель для выделения стандартных объектов, а также алгоритмы, которые используют поверхностную триангуляцию (подробное изложение алгоритмов выделения приводится в обзоре литературы). Зачастую, в таких алгоритмах достаточно большое количество работы по поиску тех или иных объектов или по определению, к какому типу объектов относится тот или иной объект возлагается на человека. В предлагаемом алгоритме выделения простейших фигур делается попытка свести к минимуму подобные обращения к «эксперту», если говорить в терминах «экспертных систем».

Целью данной работы является разработка и реализация алгоритма, позволяющего по поверхностной триангуляции выделять такие фигуры как плоскость, сфера, цилиндр, конус и тор.

1. Обзор литературы

1.1. Лазерное зондирование

Лазерное зондирование используется сравнительно недавно. На данный момент лазерные сканирующие системы делятся на системы наземного базирования [2], которые устанавливаются непосредственно на площадке, трехмерную модель которой требуется построить, а также системы воздушного сканирования [1]. Первые применяются при сканировании заводов, зданий, цехов и прочих сооружений, где необходима точность съёмки и не требуется масштабной съемки. Вторые – для съемки больших площадей, например, ЛЭП, мест, пригодных для строительства железных дорог, автострад и т.п.

Наземные сканирующие системы.

Система, как правило, состоит из портативного, работающего в автоматическом режиме, пульсового лазера и полевого персонального компьютера со специализированным программным обеспечением (рисунок 1).

Рисунок 1 - Наземная сканирующая система

Для сканирования пользователь направляет лазер на область, для которой нужно построить модель и запускает процесс сканирования, после чего может непосредственно на месте получить предварительные результаты сканирования и при необходимости повторить сканирование с другой точки. Наземные лазерные сканеры обладают углом обзора 360-180°, таким образом, как правило, особо точной настройки для снятия определённого объекта не требуется.

Процесс сканирования происходит следующим образом: пульсирующий лазер отражается от зеркал, которые определяют направление луча и луч движется в пространство, где отражается от поверхности (земли, растений, зданий). В зависимости от времени, которое потребовалось лучу, чтобы вернуться назад вычисляется расстояние до данной точки пространства, а по направлению далее вычисляется трёхмерная координата. Некоторые лазерные сканеры позволяют также ловить несколько отражённых сигналов в одном направлении, так например, можно уловить сигнал, отражённый от листьев и сигнал, который отразился от земли, пройдя через эти листья. Как правило, для построения моделей технических сооружений используют последний отражённый сигнал.

Одновременно с трехмерной съемкой рельефа (или какого-либо другого объекта) можно также производить фотографическую съемку. Которая также будет привязана к общей системе координат. Используя результаты фотографической съемки, можно будет получить цвета всех сканируемых точек. Также, можно использовать фотографии, полученные при сканировании при разрешении конфликтных ситуаций, когда распознающей системе потребуется консультация эксперта.

Для того, чтобы результаты сканирования, полученные с различных точек съёмки можно было совместить в лазерных сканирующих системах используют систему GPS

(Global Positioning System), которая позволяет получить координаты местоположения объекта (лазерного сканера) в любом месте земли с высокой точностью.

В таблице 1 представлены основные технические характеристики наземных лазерных сканирующих систем, а также их примерные значения на текущий момент (данные взяты из [2]).

Таблица 1 - Основные технические характеристики наземных лазерных сканирующих систем.

Точность	От 5 мм до 5 см.
Дальность действия, м	от 25 до 2500
угол поля зрения	от 40x40 до 360x180 градусов
Время сканирования	от 15 с до 15 мин
количество точек	от нескольких тыс. до нескольких млн.
Стоимость комплекта наземного базирования, $	от 150 тыс.

Воздушные сканирующие системы

Воздушные сканирующие системы отличаются от наземных некоторыми особенностями.

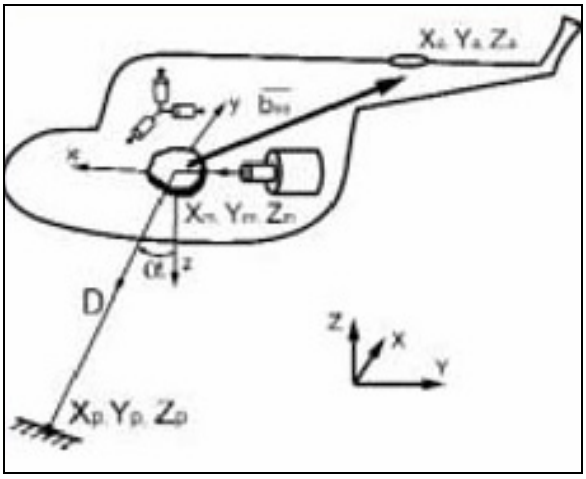

Рисунок 2 - Геометрическая схема лазерно-локационного определения пространственных координат зондируемой поверхности.

Схема действия воздушных сканирующих систем изображена на рисунке 2. Лазерный сканер расположен на борту вертолёта или самолёта (как правило, для того, чтобы установить лазерный сканер на борт летательного аппарата требуются конструктивные изменения, что, трудно осуществимо, если вертолёт берётся на прокат). Также, на борту располагается система датчиков GPS, которая указывает местоположение сканера в пространстве и направление полёта. Через определённые промежутки времени система получает свои координаты, строя потом некоторую кривую полёта во времени, во время полёта лазерный сканер сканирует поверхность и записывает время получения той или иной координаты, её направление относительно сканера и расстояние до неё. После завершения полёта эти данные могут быть обработаны и получен массив точек, который уже может быть использован в дальнейшем.

В воздушных лазерных сканерах используется специальная оптическая система, которая позволяет, в зависимости от типа системы получать точки, например, «по спирали» либо «зигзагом». Когда точки получаются «по спирали» их количество на краях сканируемой области выше, чем посередине, однако этот метод позволяет сканировать одни и те же точки, до подлёта и после, что способствует более точному распознаванию рельефа. Когда точки получаются методом «зигзаг» они получаются равномерно распределёнными. Каждый метод хорош для определённых нужд.

Так же, как и в наземных сканирующих системах, возможна одновременная фотосъемка и лазерное сканирование. Основные характеристики присущие наземным лазерным установкам присущи и воздушным системам сканирования.

1.2. Алгоритмы обнаружения объектов

Существует множество алгоритмов, которые используются для построения моделей по данным лазерного зондирования. Исследования в данной области столь разнообразны, что существуют отдельные исследования посвящённые, например применению лазерных сканеров для определения видов лесов. Когда проводится классификация лесов по породам деревьев. Данная классификация осуществляется с помощью вычисления высоты и диаметра стволов. Лазерные сканирующие системы используются также на производстве, где требуется получить цифровую модель цеха. В данном случае имеет смысл говорить о выделении правильных фигур, таких как цилиндры, плоскости и т.д. Ещё одной областью применения сканирующих лазерных систем является построение цифровых моделей рельефа местности. Где отдельно выделяется поверхность земли и сооружения, расположенные на ней. Областей применения лазерных сканирующих систем достаточно много, чтобы говорить о каком-то общем алгоритме для обработки исходных данных и получении некоторой определённой модели. Для каждой из таких областей существует достаточно большое количество разнообразных алгоритмов обладающих теми или иными недостатками и преимуществами. Идеи, положенные в основу алгоритмов, могут быть, однако, похожими. Вплоть до того, что некоторые функциональные части алгоритмов, применяемых в различных областях, могут совпадать. Такие части требуют особого внимания.

Для обнаружения объектов используют различные алгоритмы.

Алгоритмы, которые выделяют подмножества точек облака точек, которые удовлетворяют определённым статистическим гипотезам. Таким образом, можно выделить отдельно стоящие объекты, например, дома, столбы ЛЭП и т.п.

Алгоритмы, которые относят множества точек к определённому виду объектов (например «дом»), если их границы имеют сходную топологию. В таких алгоритмах, как правило, вначале строятся на множестве точек некоторые простые пространственные объекты – например, сетки точек, которые находятся примерно на одном уровне и соседние точки, в которых, расположены на небольшом расстоянии. После чего выделение происходит уже с использованием отношений между такими сетками. Вводится ряд отношений: «лежать выше», «лежать ниже», «лежать рядом», «лежать ниже всех», с помощью которых уже и выделяются подобные объекты. Такое выделение является довольно грубым, поскольку использует самые общие свойства выделяемых объектов (например, в случае выделения объекта «дом» это могут быть свойства: объект, стоящий на земле, обладающий некоторой минимальной высотой, с треугольной или плоской крышей, которая находится над четырьмя стенами).

Как правило, и как видно из предыдущего алгоритма, для обнаружения объектов исходное множество точек используется не напрямую. В различных алгоритмах для обнаружения объектов строят: сетки точек (meshes), отношения между точками (например, принадлежать одной компоненте связности, где две вершины считаются

смежными, если между ними расстояние не превышает некоторый порог), пространственную триангуляцию и, наконец, поверхностную триангуляцию. В случае триангуляции, это может быть либо триангуляция Делоне, либо триангуляция, имеющая другие ограничения, либо триангуляция без ограничений.

Далее рассматриваются алгоритмы, которые необходимы для распознавания объектов на местности. Такими объектами могут выступать строения, дороги, растительность и т.п. В зависимости от реализации алгоритма и его детализации рассматривается большее или меньшее количество классов объектов. Распознавание объектов на местности проходит в несколько этапов:

1) вначале исходное множество точек *фильтруется*, то есть точки, которые были получены случайно, отбрасываются, а остальные точки делятся на два класса: поверхность земли и остальные объекты.

2) после чего происходит процесс классификации, когда объекты, полученные на предыдущем шаге, распределяются по классам (строения, растительность и т.д.);

3) и, наконец, по выделенным объектам происходит определение их параметров, когда непосредственно строится цифровая модель.

Алгоритмы фильтрации.

Данные алгоритмы фильтруют входные данные и проводят классификацию точек: точки, принадлежащие земной поверхности и точки, принадлежащие каким-либо объектам. Данные алгоритмы используются, в основном, как предварительный этап обработки облака точек, полученного в результате воздушного сканирования местности. В работе [3] приводится сравнительный анализ достаточно большого числа алгоритмов фильтрации. В работе [4] описан алгоритм фильтрации, основанный на сегментации. Ниже приводятся характеристики различных фильтрационных алгоритмов.

Структура данных.

Выходными данными лазерного сканера является неупорядоченное множество трёхмерных точек. Иногда, такое множество может содержать метки времени получения точек, а также координаты точки, из которой данная точка была видна, это является важным, когда алгоритм использует информацию о том, что луч, направленный от сканера к поверхности не пересёк ни одного объекта на своём пути. Также, в некоторых алгоритмах делается преобразование облака точек в сетку-изображение, чтобы использовать все преимущества методов обработки изображений.

Проверка соседства точек и количество точек, фильтруемых за раз.

Фильтры всегда работают в локальных областях. В операциях классификации (например, отделения объектов от поверхности Земли) рассматриваются сразу несколько точек. Такая классификация может быть проведена тремя различными способами.

Точка-Точка – В таких алгоритмах на каждом шаге сравниваются 2 точки. Функция–определитель (которая определяет, как классифицировать точку: принадлежащей Земной поверхности или поверхности некоторого объекта) базируется на координатах этих точек. Если значение такой функции больше некоторого порога, то одна из точек относится к поверхности объекта. Только одна точка классифицируется на каждом шаге.

Точка-Точки – В таких алгоритмах соседние точки (к точке, которая в данный момент рассматривается) используются для определения значения функции-определителя. Основываясь на этом значении, точка относится к одному или другому классу. Одна точка классифицируется на каждом шаге.

Точки-Точки – В таких алгоритмах используются несколько точек, чтобы найти значение функции-определителя. Основываясь на значении функции, эти точки относятся к одному или другому классу. Более одной точки рассматривается на каждом шаге.

Определение разрывов.

Практически все алгоритмы классификации базируются на некоторых определениях меры разрыва. Такой мерой может выступать: разность высот, наклон, кратчайшее расстояние до поверхности, заданной поверхностной триангуляцией или кратчайшее расстояние до параметризованной поверхности

Идея фильтра.

Алгоритмы, основанные на вычислении угла: в таких алгоритмах функция-определитель вычисляется как угол между точками либо как разница их высот. Если угол превышает некоторое значение, то точка, располагающаяся выше, считается принадлежащей некоторому объекту.

Минимальный блок. Функция-определитель - это горизонтальная плоскость с буферной зоной над ней, если точка попадает в эту зону – то она считается принадлежащей земной поверхности, иначе – некоторому объекту.

Поверхность. Функция-определитель – это некоторая параметризованная поверхность с буферной зоной над ней. Если точка попадает в эту зону – то она считается принадлежащей земной поверхности, иначе – некоторому объекту.

Группировка. Сегментация. Некоторое множество точек, которые группируются, считаются принадлежащими некоторому объекту, если их объединение находится над ближайшей областью этого объекта. Важно отметить, что для того, чтобы данный алгоритм работал необходимо выделить некоторые очертания объектов, которые не включали бы границ объектов.

Однопроходные. Итеративные.

Одни алгоритмы фильтруют входные данные в один проход, другие - в несколько проходов. Преимущество однопроходных алгоритмов заключается в скорости выполнения. Многопроходные алгоритмы с каждым шагом увеличивают «знание» о локальных областях, которые рассматривают на каждом шаге, поэтому с каждой итерацией могут принимать более точные решения.

Замена. Браковка.

При отбраковке, точка просто выбрасывается из исходного множества. Отбраковка обычно используется в алгоритмах, которые оперируют на нерегулярных множествах точек. При замене, фильтруемая точка возвращается обратно, во множество исходных точек, с изменённой высотой (обычно интерполируемой по соседним точкам). Замена обычно используется в алгоритмах, которые пользуются растеризованными облаками точек.

Использование нескольких откликов от поверхности при сканировании.

Некоторые сканеры возвращают несколько отражённых сигналов. Такая возможность полезна при сканировании лесных массивов, когда первым откликом является отражение от поверхности растительности, тогда как последним откликом является отражение от земной поверхности. В дополнение к тому, что вычисляются несколько отражённых сигналов, вычисляется также их сила. Различные земные поверхности могут по-разному отражать или поглощать лазерный луч, вот почему использование такой информации может быть полезным при классификации точек.

После того, как исходные данные обработаны алгоритмом фильтрации, точки исходного множества оказываются разбитыми на некоторые классы. В общем случае на два класса – класс точек принадлежащих поверхности земли и класс точек, принадлежащих иным объектам. После такого выделения, объекты, а точнее, некоторые области, содержащие эти объекты, передаются на вход следующего этапа – классификации объектов.

Классификация объектов.

Как показано в работе [5], следующих параметров достаточно, чтобы провести достаточно чёткую классификацию объектов по таким классам, как: строения, растительность (деревья, кусты и т.д.) и местность. Эти параметры можно представить четырьмя пунктами:

1) градиенты на краях сегментов;

2) различие между первым и последним отражениями луча;

3) поверхность;

4) текстура высот.

Значения градиента на краях сегментов позволяют классифицировать объект как строение и растительность, или как поверхность земли (местность). По разбросу значений первого и последнего лучей, могут быть разграничены объекты-строения, и объекты, представляющие собой растительность. Поверхность строений более гладкая, нежели растительности, в большинстве случаев. И, наконец, текстура высот, в основном используется, для выделения крыш зданий, имеющих характерную текстуру.

После проведения классификации объектов, происходит переход к этапу выделения объектов.

Моделирование строений.

Моделирование строений выполняется из тех предположений, что сооружение представляет собой некоторый многогранник. Вот почему, важным является выделение плоскостей.

Плоскости выделяются, используя специальный алгоритм растущих регионов. Изначально регион является достаточно небольшим по размерам. И содержит точки, которые находятся в непосредственной близости друг от друга, а также обладают примерно сходными значениями высот. По таким точкам строится уравнение плоскости, проходящей через них. Потом, постепенно добавляются точки, которые находятся в некоторой окрестности от исходного множества, и если какая-либо точка, после её добавления, не сильно влияет на аппроксимируемую плоскость, то она добавляется к данной плоскости, иначе – выбрасывается и её влияние на строящуюся плоскость прекращается.

Как только трёхмерный объект был разбит на плоскости, анализируются топологические связи между такими плоскостями. После этого, выделяются разнообразные параметры (углы зданий, высоты, тип крыши) и строятся более точные модели.

Наряду со специфичными для каждой отдельной области алгоритмами выделения объектов, существуют и достаточно общие методы. Далее, рассматривается алгоритм, основанный на преобразовании Хафа. (В работе [6] приведено описание данного алгоритма для эффективного выделения цилиндров).

Преобразование Хафа.

Алгоритм выделения какого-либо объекта, используя преобразование Хафа, заключается в следующем. Пусть данный объект может быть задан несколькими (N) параметрами. Тогда возможные значения данных параметров дискретизуются и получается N-мерная таблица. Каждая ячейка данной таблицы ассоциирована с некоторым набором дискретизированных параметров. Для каждой вершины исходного множества вершин проверяется гипотеза принадлежности её объекту с заданными параметрами (просматриваются все варианты – все ячейки таблицы). И если данная вершина принадлежит объекту с данными параметрами, значение, в соответствующей ячейке таблицы увеличивается. После того, как были рассмотрены все точки, происходит поиск ячеек, содержащих максимальное количество точек – эти ячейки представляют собой некоторые гипотезы о возможных параметрах искомых фигур.

Преимуществами такого алгоритма являются простота и наглядность. К тому же данный алгоритм используется во многих задачах распознавания: в изображениях с шумами данный алгоритм хорошо распознаёт линии и кривые, такие как окружность, по данным лазерного зондирования алгоритм распознаёт плоскости, которые используются для распознавания строений, а также используется в других задачах.

Однако к недостаткам данного алгоритма следует отнести высокую ёмкостную и временную сложность, которые можно выразить, как $O(s^p)$ и $O(s^{p-1}n)$ соответственно, где n – количество точек исходного множества, p – число параметров и s – число значений каждого параметра, на которое производится дискретизация.

Пример. Пусть требуется по множеству точек на плоскости выделить все окружности. Окружность задаётся тремя параметрами: координатой по оси абсцисс x, координатой по оси ординат y, и радиусом r, то есть N=3. Пусть заранее известно, что все эти координаты лежат в пределах (-100;100). На рисунке 3 представлена некоторая окружность и множество точек, по которому она была выделена. Каждый из трёх параметров дискретизируется на 201 значение: -100, -99, …, 0, 1, …, 100. Поэтому s=201. Далее, для всех точек и для всех вариантов выбора параметров проверяется расстояние от точки, до предполагаемой окружности. Если оно меньше некоторого порога, то данная клетка таблицы увеличивается. В результате находятся те клетки таблицы, которые содержат достаточное количество точек, и выделяются соответствующие окружности.

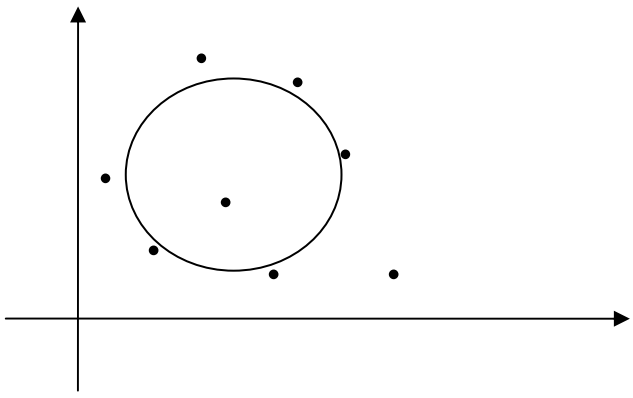

Рисунок 3 – Множество точек и окружность их аппроксимирующая.

Ещё одной характеристикой данного алгоритма является возможность выделять объекты по несвязным данным, то есть, для того, чтобы выделить окружность одного цвета на изображении совсем не обязательно, чтобы все точки окружности имели этот цвет, главное, чтобы их было достаточно много. С одной стороны это положительное

качество, поскольку позволяет выделять объекты даже на зашумлённых данных, однако, с другой стороны это не позволяет выделять части объектов, например некоторый сектор окружности.

1.3. Алгоритмы построения поверхностной триангуляции.

В данной работе, предлагается алгоритм, который использует в качестве входных данных поверхностную триангуляцию. Существует достаточно большое количество алгоритмов построения поверхностной триангуляции, однако, цель данной работы не включает в себя их изучение. Далее приводятся два алгоритма построения поверхностной триангуляции по множеству исходных точек.

Также следует отметить, что алгоритм, предложенный для выделения фигур в данной работе, в принципе, может быть модифицирован под другую структуру данных. Основным требованием к которой, является то, что точки, которые лежат на поверхности некоторой фигуры достаточно близко, расположены в близких узлах этой структуры. Таким образом, например, может быть использована сеточная модель. Однако триангуляция была выбрана потому, что подразумевает простоту реализации. Также потому, что существует достаточно большое количество алгоритмов для её построения.

Другим важным замечанием является то, что, вообще говоря, трудоёмкость построения триангуляции играет заметно меньшую роль, нежели трудоёмкость самого предлагаемого алгоритма. То есть, при использовании алгоритма, построить триангуляцию потребуется всего лишь один раз, в то время как использовать алгоритм выделения фигур, возможно, придется интерактивно, изменяя те или иные параметры поиска.

Алгоритм выпукло-вогнутой оболочки.

Данный алгоритм применяется, в основном, для построения поверхностной триангуляции сплошной поверхности земли. Однако он может быть также использован, для построения триангуляции иных поверхностей. Более полное изложение данного алгоритма, а также его применения для более позднего этапа классификации и распознавания объектов может быть найдено в работе [5].

Алгоритм заключается в следующем:

1) на основе наиболее низко располагающихся точек строится выпуклая оболочка (используя триангуляционную структуру);

2) для каждого треугольника выпуклой оболочки проводится поиск точек, которые располагаются непосредственно над ним, и таких, которые располагаются в непосредственной близости от него;

3) после чего, среди точек выбранных на шаге 2 выбирается одна, которая наиболее подходит по какому-либо критерию и треугольник, над которым она располагается, разбивается на три, а выбранная точка включается в поверхность.

4) шаги 2 и 3 повторяются до тех пор, пока существуют точки удовлетворяющие условиям.

В результате строится достаточно хорошая поверхностная триангуляция, которая содержит выпуклые и вогнутые части.

Алгоритм, использующий объемную триангуляцию и линии обзора.

В работе [7] приводится следующий метод построения поверхностной триангуляции. В качестве исходных данных, в нём используется массив точек, а также линии обзора этих точек, то есть линии, начинающиеся в том месте, откуда пускался

лазерный луч, и заканчивающиеся в точке, где он отразился от поверхности. Далее приводится описание этого алгоритма для размерности 2.5. То есть когда съёмка поверхности происходит с достаточно удалённой от местности точки, например с самолёта. В статье [7] также описывается реализация данного алгоритма для размерности 3, которая содержит незначительные изменения.

Алгоритм включает в себя несколько шагов.

1) Вводятся точки и линии обзора.

2) Конструируется соответствующая объёмная триангуляция (тетраэдрация) Делоне (по аналогии с триангуляцией Делоне на плоскости существует объёмная триангуляция, когда элементами такой структуры становятся не треугольники, а тетраэдры, и ограничения, которые накладывались на треугольники, накладываются на тетраэдры – чтобы никакая точка не попадала в сферу, описанную вокруг тетраэдра). Следует отметить, что при построении тетраэдрации также учитываются рёбра ограничений, которые после построения тетраэдрации включаются в неё. Для этого используются точки Steinera, которые добавляются в тетраэдрацию и являются, по сути, точками пересечения исходных линий обзора с поверхностями тетраэдров.

3) На этом шаге выделяются сильные рёбра. Если грань некоторого тетраэдра содержит одну точку Steinera и две точки, полученные из исходного множества, то алгоритм преобразует точку Steinera в точку, полученную из исходного множества. Ребра такой грани помечаются как сильные. Также ребра граней, которые состоят из трёх точек исходного множества, помечаются как сильные. Такое множество сильных рёбер определяет тело объекта как выпуклую трёхмерную оболочку точек исходного множества.

4) На этом шаге выделяются ребра принадлежащие поверхности. Для этого применяется алгоритм удаления невидимых рёбер, чтобы выделить ребра поверхностной триангуляции. В этом алгоритме все сильные ребра проецируются на некоторую двумерную поверхность, и проводится поиск точек их пересечения. В каждой такой точке производится вычисление координат Z, каждого из ребер попавшего в пересечение. Ребро, которое имеет наименьшее значение наиболее удалено от наблюдателя и, поэтому, не располагается на поверхности. Такое ребро удаляется из списка сильных ребер. Все оставшиеся ребра считаются принадлежащими поверхностной триангуляции.

Проблема возникает в том, что некоторые ребра, которые были удалены, должны быть включены в набор ребер поверхности для того, чтобы получить полную и правильную триангуляцию. Удаленные сильные ребра, которые не имеют пересечений (в проекции) с другими удаленными ребрами, поэтому, добавляются в список ребер поверхностной триангуляции.

5) Наконец, вычисляется поверхность триангуляции. Ребра, полученные на предыдущем шаге, составляют непересекающиеся треугольники, которые определяют поверхность. Внутренняя структура ссылок в ребрах, позволяет построить грани поверхностной триангуляции.

Данный алгоритм, несомненно, более информативный, и получает более хорошую триангуляцию, поскольку использует в качестве своей основы, во-первых, тетраэдрацию, а во-вторых, линии обзора. Однако, не смотря на все свои преимущества, алгоритм является достаточно трудоёмким из-за необходимости учёта всех ограничений, а ограничений в данном случае столько же, сколько и точек. Поэтому, на практике, почти всегда пользуются алгоритмами, которые, во-первых, не используют информацию о линиях обзора, а во-вторых, проводят свои вычисления либо в 2.5 пространстве или в проекции на двухмерную плоскость.

2. Алгоритм выделения фигур. Общие замечания.

В данной работе предлагается алгоритм выделения пространственных фигур по поверхностной триангуляции. Данный алгоритм состоит из двух этапов: на первом этапе [8] происходит выделение плоскостных объектов, на втором, по этим плоскостным объектам делается попытка построить более сложные фигуры, такие как сфера, конус, цилиндр или тор.

В самом общем виде первый этап может быть представлен следующим образом. На поверхностной триангуляции выбирается некоторый треугольник, после чего выбирается некоторое множество соседних с ним треугольников. Используя вершины, составляющие такое множество треугольников, строится приближение некоторой плоскости (используя метод наименьших квадратов). После чего все треугольники, которые связаны с исходным множеством, проверяются на принадлежность этой плоскости, и если таких треугольников оказывается достаточно, кусочно-плоскостной объект выделяется, а треугольники, принадлежащие ему, выбрасываются из дальнейшего рассмотрения. Таким образом, в результате первого этапа имеется набор кусочно-плоскостных объектов и треугольников, их составляющих.

Пусть выделен набор кусочно-плоскостных объектов, каждый из которых задаётся набором треугольников принадлежащих данному объекту и набором значений параметров уравнения плоскости, наиболее близкой (в смысле суммы квадратов расстояний) данному набору точек. Тогда, используя связи между такими объектами можно выделять более сложные объекты, такие как сфера, цилиндр, конус и тор. Такое выделение и составляет задачу второго этапа.

В предлагаемом методе выделение сложных объектов предлагается проводить следующим образом: сначала, построить граф связей между выделенными плоскостями, после чего, используя построенный граф выделить сами объекты. Понятно, что после выделения объектов - плоскостей, каждый такой объект может оказаться частью более чем одного сложного объекта (например, плоскость окажется с некоторым приближением частью шара, частью цилиндра и т.д.). В таком случае, разумным представляется выделение такого объекта, который будет иметь максимальные размеры (количество треугольников, его составляющих, число вошедших плоскостных объектов или расстояние между максимально удалёнными точками).

В третьей и четвёртой частях приводится более детальное описание каждого из этапов работы такого алгоритма.

3. Первый этап алгоритма: выделение кусочно-плоскостных объектов.

В качестве входа к первому этапу служит набор вершин в трёхмерном пространстве – суть точек, находящихся на поверхностях разнообразных тел, которые участвовали в процессе сканирования. Также на входе этого этапа используется поверхностная триангуляция, составленная на множестве данных точек и исходных ограничений. Триангуляция представляется множеством треугольников, содержащих ссылки на их вершины, а также ссылки на соседние (через общие ребра) треугольники.

3.1. Общий алгоритм.

Алгоритм выделения плоскостных объектов представляется следующим образом:

Цикл по всем треугольникам (те, которые помечены и относятся к какому-либо выделенному кусочно-плоскостному объекту, пропускаются);

Шаг 1: на этом шаге, начиная от текущего треугольника, в ширину выбираются треугольники. Треугольники выбираются таким образом, чтобы брать ещё не вошедшие ни в один кусочно-плоскостной объект и связные с текущим множеством выбранных треугольников. Как только количество треугольников достигнет некоторого (задающегося параметром в программе) уровня, которого достаточно для построения аппроксимирующей множество этих треугольников плоскости, выбор прекращается. Если такого количества треугольников достичь не удалось – то происходит переход на следующую итерацию цикла. На рисунке 4 представлено поэтапное выделение области треугольников (необходимо представлять себе эти рисунки как проекции некоторой области поверхностной триангуляции). Номерами показан порядок выделения треугольников. Слева выделен один изначальный треугольник, посередине уже четыре треугольника: начальный и три смежных с ним, справа показаны все 8мь (если задавался параметр равный восьми) выбранных треугольников, по вершинам которых далее строится модель плоскости.

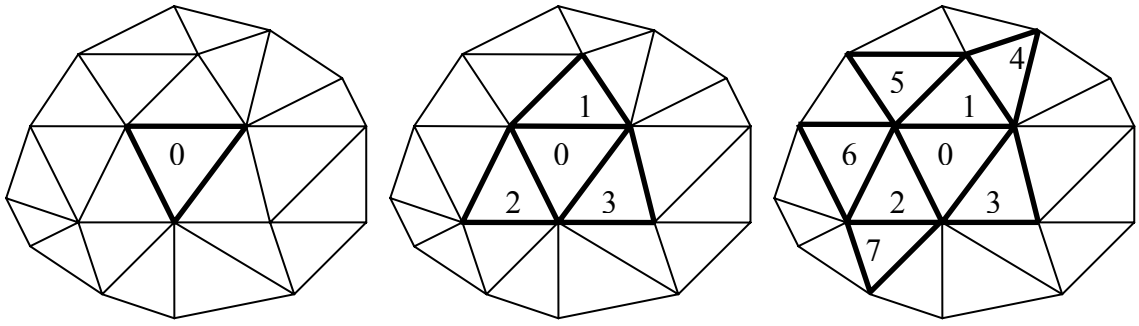

Рисунок 4 – выделение множества треугольников, необходимых для построения аппроксимирующей плоскости.

Шаг 2: по выбранному на первом шаге множеству точек строится аппроксимирующая плоскость, которая задаётся уравнением $Ax + By + Cz + D = 0$. Аппроксимация проводится в смысле метода наименьших квадратов. То есть сумма квадратов расстояний от точек до полученной плоскости минимальна среди всех возможных плоскостей.

Шаг 3: после того, как была построена плоскость, происходит выделение самого плоскостного объекта. Аналогично тому, как это делалось на первом шаге, начиная от первого треугольника, в ширину выбираются треугольники. Выбор треугольника происходит в том случае, если все три его вершины лежат от плоскости, полученной на шаге 2, не более чем на некотором расстоянии δ (которое также задаётся параметром в программе). Если треугольник выбран, то рассматриваются его соседи, на предмет возможности их выделения, если треугольник не попал во множество выбранных, то на нём поиск прекращается. Таким образом, достигается связность всей области выделения. На рисунке 5 выделено некоторое множество треугольников, составляющих кусочно-плоскостной объект. Необходимо заметить, что область, которая выделяется, может быть какой угодно, единственное ограничение, которое на неё накладывается – связность. Ещё одной особенностью является тот факт, что треугольники, по которым строится аппроксимирующая плоскость, вообще говоря, не обязательно должны входить в область выделения. Такие треугольники могут находиться далеко от плоскости, которая была построена с их участием.

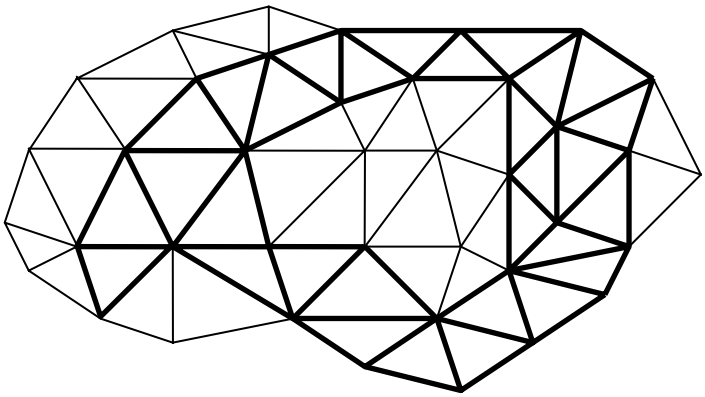

Рисунок 5 – выделенный кусочно-плоскостной объект.

Шаг 4: если треугольников, которые были выбраны на шаге 3 достаточно для выделения кусочно-плоскостного объекта (третий параметр в алгоритме), то он выделяется, а треугольники, его составляющие помечаются принадлежащими ему.

Конец цикла.

3.2. Замечания к алгоритму.

Для второго шага далее приводится алгоритм, который находит аппроксимирующую плоскость.

На третьем шаге, когда выделяется область треугольников подозрительная на кусочно-плоскостной объект, возможны некоторые вариации. Так, например, возможно такое изменение алгоритма, что если бы треугольник был включён во множество принадлежащих плоскости треугольников, то рассматривались бы не только треугольники непосредственно смежные с ним, но и треугольники на расстоянии 3. Такой подход улучшит качество распознавания, поскольку некоторые случайные погрешности в измерениях плоскостей, или случайные обрывы будут пропущены при выделении объекта. Однако, такой подход, несомненно, более трудоёмкий. На рисунке 6 показаны два варианта выделения плоскостных объектов. Слева показан способ, в котором, если некоторый треугольник был выделен, то просматриваются смежные с ним треугольники и на этом просмотр завершается. На рисунке 3, справа, показан другой способ выделения. Если некоторый треугольник был выбран (выделен пунктиром), то рассматриваются треугольники на расстояниях 1 и 2 (выделены жирным), и на расстоянии 3 (выделены

точками) от него. Что позволяет, иногда, пропускать треугольники которые не находятся близко к текущей плоскости и продолжать выделять кусочно-плоскостной объект далее.

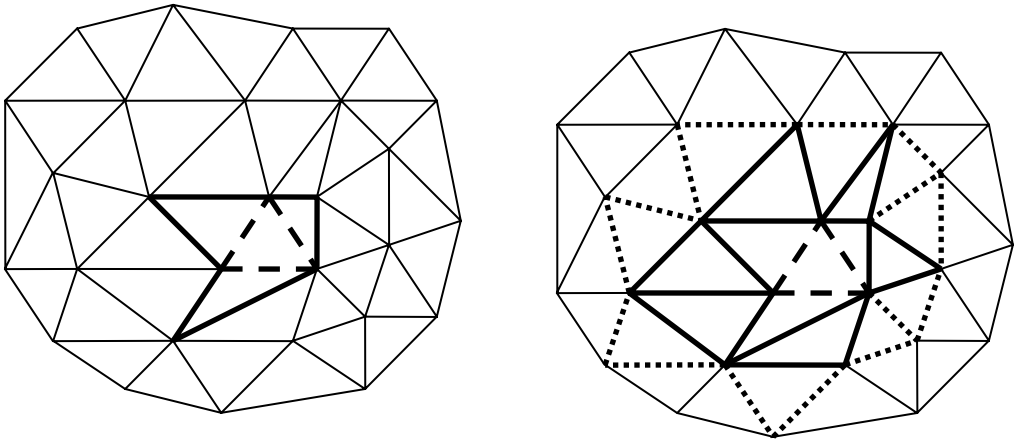

Рисунок 6 – два варианта выделения плоскостных объектов.

3.3. Выделение плоскости, по множеству точек используя метод наименьших квадратов.

Описание данного алгоритма можно найти также в [9]. Как и множество других алгоритмов, использующих метод наименьших квадратов для выделения плоских и пространственных фигур и кривых. Здесь приводится лишь алгоритм выделения плоскостей, почти дословно.

Во-первых, необходимо задать плоскость. Для этого достаточно указать точку X - принадлежащую плоскости и $a = A/|A|$ - вектор направляющих косинусов нормали к поверхности. Где A - вектор направления нормали.

Во-вторых, определить функцию расстояния между точкой и плоскостью, которая задаётся формулой (1).

$$d_i = d(X_i) = d(X_i, X, a) = a(X_i - X) \tag{1}$$

В-третьих, определить функцию, которую требуется минимизировать. Данная функция (2) представляет собой сумму квадратов расстояний между точками и плоскостью.

$$J(X, a) = \sum \left[a(X_i - X)\right]^2 \tag{2}$$

Центральная точка множества точек (среднее арифметическое координат всех точек) должна лежать на плоскости, которая строится. Это можно увидеть, поскольку $\nabla J = 0$ для плоскости – решения, и, следовательно $\sum a(X_i - X) = 0$. Умножение на $1/N$ даёт $\frac{a}{N}\sum(x_i - x) + \frac{b}{N}\sum(y_i - y) + \frac{c}{N}\sum(z_i - z) = 0$, раскрытие сумм даёт $a(\bar{x} - x) + b(\bar{y} - y) + c(\bar{z} - z) = 0$, это значит что $d(\overline{X}, X, A) = 0$, то есть точка $\overline{X}$ лежит на

плоскости – решении. Далее, зная точку, принадлежащую плоскости-решению, происходит параллельный перенос всех исходных точек таким образом, чтобы эта точка передвинулась в начало координат. Точкой X берётся начало координат и фактически задача сводится к поиску вектора нормали к плоскости.

Направление искомой плоскости, a, может быть найдено при помощи решения задачи минимизации с ограничениями. Требуется минимизировать функцию J с ограничением $|a| = 1$. Для этого определяется функция, $G = |a|^2 - 1$, так, что проблема минимизации, сводится к минимизации J при ограничении $G = 0$. Метод множителей Лагранжа определяет минимум в точке, в которой $\nabla J = \lambda \nabla G$, где λ, некоторое вещественное число. (Так как, a, b и c считаются независимыми переменными, а ограничение хранится в G. То, $\nabla = (\partial/\partial a, \partial/\partial b, \partial/\partial c)$.) Но $\nabla G = 2a$, и $\nabla J = 2(M^T M)a$ (где i-ая строка матрицы M представляет собой координаты i-ой точки), в результате получается уравнение $(M^T M)a = \lambda a$, решение которого состоит в поиске собственных векторов.

Такая задача 3x3 может быть просто решена, используя многие хорошо известные методы (например, метод Якоби). Однако нужно заметить, что собственные векторы $M^T M$ также являются сингулярными векторами (из сингулярного разложения) M. Это позволяет достичь численной стабильности применяя сингулярное разложение (SVD – Singular Value Decomposition) к M без вычисления $M^T M$.

Наконец, нужно определить, как выбирать корректный собственный вектор (или сингулярный вектор) из трёх, полученных при помощи SVD. Уравнения нормали могут быть записаны в виде (3).

$$\begin{aligned} \sum x_i (aX_i) &= \lambda a \\ \sum y_i (aX_i) &= \lambda b \\ \sum z_i (aX_i) &= \lambda c \end{aligned} \tag{3}$$

Умножая эти три уравнения на a, b и c, соответственно, затем, суммируя равенства, получается формула (4)

$$\sum (aX_i)^2 = \lambda |a|^2 = \lambda \tag{4}$$

Но сумма левой части – это функция J, то есть сумма квадратов расстояний между точками и плоскостью. Поэтому значение λ - это значение оптимизируемой функции, поэтому, корректным собственным вектором является вектор, обладающий наименьшим собственным значением. Если используется сингулярное разложение, выбирается вектор, соответствующий минимальному сингулярному значению, поскольку сингулярные значения есть квадратные корни собственных значений.

4. Второй этап алгоритма: выделение пространственных фигур по набору кусочно-плоскостных объектов.

К завершению первого этапа (выделения плоскостных объектов) имеются следующие данные:

1) изначальный массив вершин;

2) массив треугольников составляющих поверхностную триангуляцию, каждый треугольник, которого, содержит пометку – индекс плоскости, в которую он входит или пометку о том, что данный треугольник не принадлежит ни одной из выделенных плоскостей;

3) массив кусочно-плоскостных объектов, в котором каждая плоскость содержит индекс первого треугольника, принадлежащего данному объекту.

Следует отметить, что, имея индекс первого треугольника принадлежащего кусочно-плоскостному объекту можно получить полный список вершин или список треугольников, принадлежащих данному объекту за линейное время от количества треугольников в него входящих.

На втором этапе, используя кусочно-плоскостные объекты, выделяются более сложные объекты, такие как сфера, цилиндр, конус и тор (далее будем называть такие объекты фигурами). Следующие процедуры используются в алгоритме такого выделения.

4.1 Вспомогательные процедуры второго этапа.

Определение связности кусочно-плоскостных объектов.

Определение: два кусочно-плоскостных объекта назовём связными, если существуют хотя бы два треугольника с общим ребром таких, что один принадлежит первому кусочно-плоскостному объекту, а второй – второму.

Вообще говоря, данное определение может быть менее строгим и также включать в себя кусочно-плоскостные объекты, треугольники которых имеют общую вершину или соединяются через некоторый третий треугольник. Однако реализация более строгого ограничения обладает меньшей трудоёмкостью, как в поиске смежных плоскостей, так и при самом выделении фигур (строгое определение связности сокращает количество смежных плоскостей, на основе которых выделяются фигуры). Меньшая строгость в определении связности ведёт к интуитивно более качественному выделению.

На рисунке 7 представлен пример – кусок поверхностной триангуляции, с обозначениями – индексами плоскостных объектов, которым принадлежат треугольники. Пусть, на некотором шаге обнаружения связей рассматривается выделенный треугольник. Тогда, если следовать строгому определению, то следует добавить связи между кусочно-плоскостными объектами 1-2 и 1-3 (связи симметричны, поэтому также имеют место связи 2-1 и 3-1). Если к строгому определению добавить также возможность связности «через» вершину, то следует определить связи: 1-2, 1-3, 1-4 и 1-5, а также симметричные им. И, наконец, если определить связность как смежность двух смежных по ребру или по вершине треугольников или наличие связи «через» треугольник по рёбрам, тогда при рассмотрении выделенного на рисунке треугольника, следует отметить связи: 1-2, 1-3, 1-4, 1-5 и 2-3, а также симметричные им.

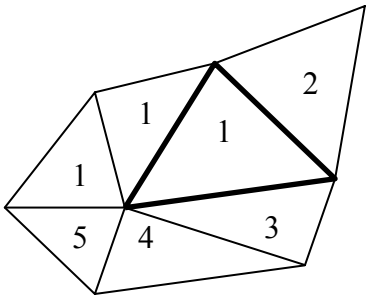

Рисунок 7 – набор треугольников триангуляции, с отметками – индексами кусочно-плоскостных объектов, к которым они принадлежат.

Для хранения связей между кусочно-плоскостными объектами, в каждом из них имеется список смежных объектов.

Алгоритм: для получения всех списков смежности исходное множество треугольников просматривается полностью и для каждого треугольника просматриваются его соседи (по рёбрам), если треугольник и его сосед принадлежат двум различным кусочно-плоскостным объектам, то в списки связности компонент добавляются соответствующие записи.

Трудоёмкость: линейная относительно количества треугольников в исходной поверхностной триангуляции.

Определение положения одного плоскостного объекта относительно другого.

Данная процедура необходима для определения выпуклости набора плоскостных объектов, а также для решения вопроса о том, по какую сторону от плоскостного объекта находится фигура, которую он ограничивает (касается).

Вход: два кусочно-плоскостных объекта a и b.

Выход: разность между количеством точек, принадлежащих объекту b которые находятся с положительной стороны плоскости a (то есть при подстановке в уравнение плоскости a, дающих положительный результат) и количеством точек b находящихся с отрицательной стороны плоскости a.

На рисунке 8 показаны две плоскости (a и b) и точки, кусочно-плоскостного объекта b. Подавляющее большинство (12 из 14) лежит по одну из сторон (на рисунке – выше) плоскости, образованной плоскостным объектом a.

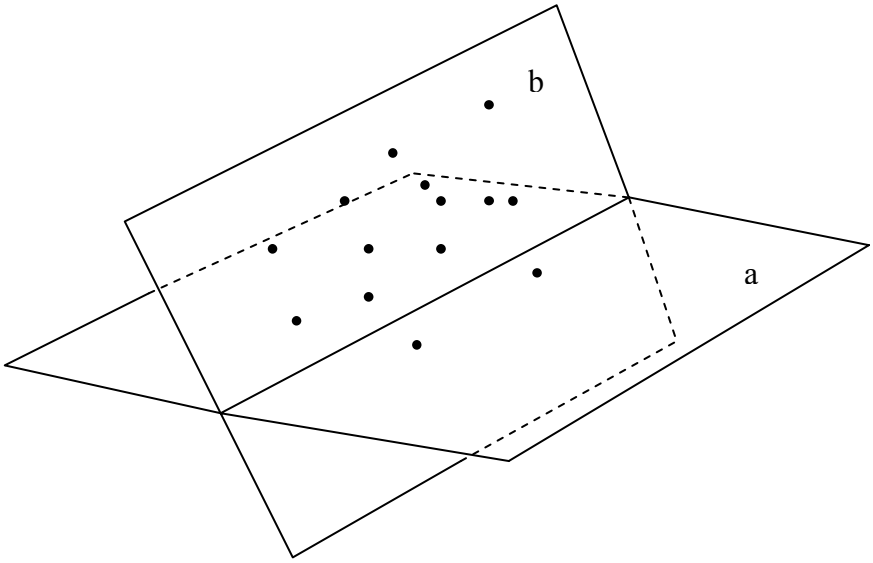

Рисунок 8 – две плоскости образованные кусочно-плоскостными объектами a и b, а также точки объекта b.

Допущение: считается, что кусочно-плоскостной объект b располагается над плоскостью, образованной кусочно-плоскостным объектом a если результат данной процедуры положителен, то есть более 50% точек b находятся с положительной стороны a.

Алгоритм: последовательно выбираются все точки, принадлежащие плоскостному объекту b, и подставляются в уравнение плоскости объекта a.

Трудоёмкость: линейная относительно количества точек объекта b.

Определение выпуклости набора плоскостных объектов.

Для выделения выпуклых фигур, таких как сфера, цилиндр и конус необходимо, чтобы набор кусочно-плоскостных объектов, по которым строятся эти объекты, также был выпуклым. По аналогии с определением выпуклости для Евклидовой геометрии (определение 1) определим выпуклость для набора кусочно-плоскостных объектов (определение 2).

Определение 1: Выпуклым, называется такое множество точек, что любая линейная комбинация двух из них также принадлежит данному множеству.

Определение 2: Выпуклым, назовём такое множество кусочно-плоскостных объектов, что, взяв любые три из них (a,b и c), точки одного из них (c) лежат в части пространства, ограниченной двумя другими кусочно-плоскостными объектами (a и b).

Определение 2 несколько поверхностно. Во-первых, оно не даёт ответ на вопрос, какая часть пространства ограничена плоскостями a и b, поскольку две плоскости делят пространство на 4 части и не выделяют из них какую-либо одну. Во-вторых, для реальных данных, не все точки с могут лежать в этой части пространства, но подавляющее их большинство. Для ответа на эти вопросы предлагается следующее конструктивное определение выпуклости:

Множество кусочно-плоскостных объектов является выпуклым, если для любых трёх из них (a,b и c) выполняется следующее:

1. подавляющее большинство точек c лежит по ту же сторону от b, что и подавляющее большинство точек a;

2. подавляющее большинство точек c лежит по ту же сторону от a, что и подавляющее большинство точек b.

Алгоритм: последовательно выбираются все тройки плоскостей и, для каждой из них, проверяется выполнимость двух условий описанных выше (это осуществляется при помощи процедуры определения положения одного плоскостного объекта относительно другого)

Трудоёмкость: кубическая относительно количества плоскостных объектов (если считать процедуру определения взаимного положения двух плоскостных объектов простейшим действием). Однако, учитывая то, что для выделения фигур требуется ограниченное количество таких объектов (на пример, для сферы четыре) то можно сказать, что трудоёмкость алгоритма O(1). Таким образом, полная трудоёмкость имеет тот же порядок, что и трудоёмкость процедуры определения взаимного положения двух плоскостных объектов.

Алгоритм Левенберга-Маркварда.

Описание данного алгоритма можно найти также в работе [9], здесь приводится его почти дословный перевод с сохранением всех обозначений.

Алгоритм Левенберга-Маркварда находит такой вектор, p, который минимизирует функцию $J(p) = \sum d_i^{\,2}(p)$, где $d_i(p)$ - расстояние от точки X_i до поверхности, определённой вектором параметров p. В случае сферы, $p = (x, y, z, r)$ и $d_i(p) = \left|X_i - X\right| - r$. Одной из идей алгоритма является идея аппроксимировать функцию J как линейную функцию от p, $\hat{J}(p)$, что представлено в формуле (5).

$$J(p) \approx \hat{J}(p) = \sum (d_i(p_0) + \nabla d_i(p_0) p)^2 \qquad (5)$$

Где $\nabla d_i(p_0)$ - градиент $d_i(p)$ полагающийся изначально равным p_0. Аппроксимация считается верной внутри региона некоторого радиуса. Далее принимается решение, как минимизировать $\hat{J}(p)$. Направление поиска вычисляется, основываясь на $\hat{J}(p)$. В этом направлении производится поиск (естественно, не выходя за рамки региона, внутри которого аппроксимация считается верной) точки p_{new}, такой, что $J(p_{new}) < J(p_0)$. Когда p_{new} вычислена, она становится новой p_0 для другой итерации.

Основой для алгоритма является тот факт, что решение p^*, для каждой итерации может быть представлено в виде формулы (6).

$$p^* = p(\lambda) = -(F_0^{\,T} F_0 + \lambda D^T D)^{-1} F_0^{\,T} d(p_0) \qquad (6)$$

Где F_0 - это матрица, в которой $\nabla d_i(p_0)$ является i-ой строкой, D - подходящая весовая матрица, $d(p_0)$ - вектор остатков $d_i(p_0)$ и $\lambda \geq 0$ - переменная, названная параметром метода Левенберга-Маркварда. Алгоритм Левенберга-Маркварда представлен ниже. Матрица $F_0^T F_0 + \lambda D^T D$ названа матрицей H, а также вектор $F_0^T d(p_0)$ - v. Алгоритм включает модификации предложенные J.C. Nash, которые заключаются в использовании весовой матрицы таким образом, что $D^T D$ - суть единичная матрица плюс диагональ матрицы $F_0^T F_0$. Использование единичной матрицы в определении D влечёт то, что матрица H является положительно определённой. (D никогда не вычисляется.) Так как H - также симметрична, система $Hx = -v$ может быть решена, используя разложение Холецкого. В качестве коэффициентов для увеличения и уменьшения λ используются 10 и 0.4 соответственно. Наконец, после каждой итерации следует нормализовать вектор параметров p, хотя нормализация не изменяет значение функции $J(p)$.

Алгоритм:

Входом является некоторый начальный вектор p_0, содержащий определённые параметры. На выходе получается вектор p_0, который минимизирует функцию J.

Установить $\lambda \leftarrow 0.0001$

Цикл

Уменьшить λ; Нормализовать p_0.

Присвоить $U \leftarrow F_0^T F_0$; $v \leftarrow F_0^T d(p_0)$; $J_0 \leftarrow \sum d_i^{\,2}(p_0)$

Цикл

Увеличить λ;

Присвоить $H \leftarrow U + \lambda(I + diag(U))$

Решить систему $Hx = -v$

Присвоить $p_{new} \leftarrow p_0 + x$; $J_{new} \leftarrow \sum d_i^{\,2}(p_{new})$

Если сошлось, Присвоить $p_0 \leftarrow$ *Нормализованный* p_{new}; вернуть p_0

Выход, если $J_{new} < J_0$ или предельное количество итераций исчерпано.

Если $J_{new} < J_0$, $p_0 \leftarrow p_{new}$

Выход, если предельное количество итераций исчерпано.

Описанных выше процедур достаточно, чтобы можно было описать общий алгоритм выделения. Все они находят применение в нём, вне зависимости от того, какая поверхность выделяется.

4.2. Общий алгоритм.

Для выделения таких фигур как сфера, цилиндр и конус можно (абстрагируясь от деталей) описать общий алгоритм выделения.

Алгоритм:

Шаг 1: выбирается необходимое количество (для выделения текущего типа фигуры) связных кусочно-плоскостных объектов, таким образом, чтобы выполнялось два условия для этих объектов: чтобы они были выпуклы, и топология связей между ними не повторялась (более детально вопросы топологии связей будут рассмотрены в описании алгоритма непосредственно для каждой из фигур).

Шаг 2: для данного набора кусочно-плоскостных объектов строится (в первом приближении) фигура, которую они ограничивают исходя только из уравнений плоскостей.

Шаг 3: на данном этапе по точкам, составляющим выбранные на шаге 1 кусочно-плоскостные объекты, итеративно, отбрасывая точки находящиеся достаточно далеко от текущего приближения параметров фигуры, строится новое приближение по методу наименьших квадратов с использованием алгоритма Левенберга-Маркварда.

Шаг 4: после того, как получено некоторое приближение параметров фигуры на предыдущем шаге, оно используется для определения принадлежности других плоскостных объектов данной фигуре. Начиная от выбранных на шаге 1 плоскостных объектов, рассматриваются, в ширину, все связные с ними кусочно-плоскостные объекты на предмет принадлежности выделяемой фигуре. Если какой-либо объект оказывается принадлежащим данной фигуре, он прикрепляется к данной фигуре и исключается из дальнейшего рассмотрения (для данного типа фигур), все связные с ним объекты подвергаются аналогичной проверке. Если объект не принадлежит данной фигуре, то для него выполнение алгоритма прекращается. Считается, что некоторый плоскостной объект принадлежит текущей фигуре, если хотя бы половина его точек лежит на её поверхности (с некоторым допуском).

Шаг 5: если на предыдущем шаге было выделено достаточное количество кусочно-плоскостных объектов (параметром задаётся требуемое общее количество вершин в них, принадлежащих поверхности этой фигуры, необходимое для её выделения), то данная фигура выделяется, иначе все изменения, сделанные на шаге 4 отменяются.

Шаг 6: предыдущие шаги повторяются до тех пор, пока на первом шаге не будут выбраны все возможные наборы плоскостных объектов.

Данный алгоритм выполняется отдельно для выделения сфер, конусов и цилиндров, для которых хранятся отдельно списки использованных плоскостей, списки выделенных фигур и плоскостей, принадлежащих этим фигурам.

4.3. Выделение сфер.

Любую сферу можно точно определить через четыре плоскости, которые её касаются. Далее более детально по шагам описан алгоритм выделения фигур, представленный выше в его приложении к выделению сфер.

На первом шаге требуется найти все варианты расположения связей между четырьмя плоскостями так, чтобы они не повторялись при выборе этих четырёх плоскостей. Это необходимо для того, чтобы выбирать четыре плоскости по спискам смежности, а не по исходному массиву, что увеличивает трудоёмкость выбора и постараться не выбирать одно и то же множество плоскостей дважды. На рисунке 9 схематично представлены два варианта топологий связей. Круги представляют собой плоскостные объекты, жирным выделены те плоскости, которые берутся из начального множества плоскостных объектов. Слева, три кусочно-плоскостных объекта берутся из

списка смежных с расположенным в центе. Справа, два кусочно-плоскостных объекта берутся из списка смежных к выделенному кусочно-плоскостному объекту, а третий из списка смежных к одному из этих двух.

Рисунок 9 – топология связей четырёх связных кусочно-плоскостных объектов.

На втором шаге строится модель сферы по четырём плоскостям, касающимся её. Для того чтобы определить сферу, решается система линейных уравнений, каждое из которых представляет собой расстояние от плоскости до центра сферы. Расстояние от некоторой плоскости, заданной уравнением $Ax + By + Cz + D = 0$ до точки $P_0(x_0, y_0, z_0)$ может быть вычислено по формуле (7).

$$d = \frac{Ax_0 + By_0 + Cz_0 + D}{\sqrt{A^2 + B^2 + C^2}} \tag{7}$$

Где d - расстояние, определённое по величине и по знаку. Иногда d называют отклонением точки от плоскости, отклонение положительно, если точка $P_0(x_0, y_0, z_0)$ и начало координат лежат по разные стороны от плоскости, и отрицательно, если по одну сторону. Поскольку коэффициенты всех четырёх плоскостей известны, формуле (7) можно вычислить знаменатель и поделить на него, в результате получим формулу (8), которая выражает расстояние r от плоскости до центра сферы.

$$r = \left| A'x_0 + B'y_0 + C'z_0 + D' \right| \tag{8}$$

Модуль появляется здесь потому, что в формуле (7) отклонение имеет произвольный знак, а радиус сферы всегда положителен. Для того чтобы узнать какой знак будет иметь выражение после раскрытия модуля, примем во внимание, что центр сферы будет лежать по ту же сторону от плоскости, по которую лежит любой другой кусочно-плоскостной объект (подавляющее множество его точек), из выбранных для выделения сферы. Обозначим знак перед радиусом r после раскрытия модуля в формуле (8) символом E'. И после некоторых перестановок получим уравнение (9).

$$A'x_0 + B'y_0 + C'z_0 - E'r = -D' \tag{9}$$

Аналогично получаются оставшиеся три уравнения для остальных плоскостных объектов, в результате получается система (10).

$$\begin{pmatrix} A'_1 & B'_1 & C'_1 & -E'_1 \\ A'_2 & B'_2 & C'_2 & -E'_2 \\ A'_3 & B'_3 & C'_3 & -E'_3 \\ A'_4 & B'_4 & C'_4 & -E'_4 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ z_0 \\ r \end{pmatrix} = \begin{pmatrix} -D'_1 \\ -D'_2 \\ -D'_3 \\ -D'_4 \end{pmatrix} \tag{10}$$

Решение системы (10) даёт точку $P_0(x_0, y_0, z_0)$ - центр сферы, которую касаются четыре плоскостных объекта, а также его радиус. Точки каждого кусочно-плоскостного объекта располагаются по одну и другую сторону от плоскости, им образованной, на расстоянии, не превышающем некоторого η. Поэтому, имеет смысл предположить, что реальный радиус R сферы равен $r + \eta$.

На третьем шаге из четырёх плоскостей образующих кусок поверхности сферы берутся все точки, и, используя приближение, полученное в результате решения системы (10) в качестве начального для метода наименьших квадратов, получается более точное решение. Для получения решения по методу наименьших квадратов используется алгоритм Левенберга-Марквардa, на выходе которого получается решение – некоторый локальный минимум, поэтому, чтобы этот локальный минимум оказался также глобальным очень важно выбрать хорошее начальное приближение. Выделение сферы по четырём плоскостям является хорошим приближением, поскольку основывается на достаточно большом множестве точек, которые составляют эти плоскостные объекты. Далее приводятся некоторые формулы, необходимые для выделения сферы методом наименьших квадратов с использованием алгоритма Левенберга-Марквардa.

Параметры, задающие сферу: X - центр сферы и r - её радиус. Формула (11) представляет собой уравнение, выражающее расстояние от поверхности сферы до точки.

$$d(X_i) = \left|X_i - X\right| - r \tag{11}$$

Формула (12) представляет собой функцию, минимум которой ищется в методе наименьших квадратов – суть сумму квадратов расстояний от поверхности сферы до множества точек.

$$J(X, r) = \sum (\left|X_i - X\right| - r)^2 \tag{12}$$

Где сумма берётся по всем точкам множества (в данном случае, по всем точкам, составляющим четыре плоскости, по которым выделяется сфера). Нормализация значений, в методе Левенберга-Марквардa в случае выделения сферы не требуется. Далее, приводятся производные, необходимые для этого метода (формулы (13) – (16)).

$$\frac{\partial d_i}{\partial x} = \frac{-(x_i - x)}{\left|X_i - X\right|} \tag{13}$$

$$\frac{\partial d_i}{\partial y} = \frac{-(y_i - y)}{|X_i - X|} \tag{14}$$

$$\frac{\partial d_i}{\partial z} = \frac{-(z_i - z)}{|X_i - X|} \tag{15}$$

$$\frac{\partial d_i}{\partial r} = -1 \tag{16}$$

После получения некоторого более хорошего приближения с помощью метода Левенберга-Маркварда, процесс повторяется с точками, которые находятся на расстоянии от сферы не превышающем $\eta / 2$, потом с точками на расстоянии $\eta / 4$ и т.д.

Четвёртый и пятый шаги общего алгоритма выделения фигур, описанного выше, не обладают какими-либо особенностями, присущими только сферам, поэтому остаются такими, какие они есть в общем алгоритме.

4.4. Выделение цилиндров.

Для выделения цилиндра требуется три плоскости, касающиеся его поверхности, однако, трех плоскостей оказывается много для точного определения цилиндра, а двух недостаточно. Для точного определения цилиндра требуется три плоскости, любые две из которых пересекают третью по параллельным прямым.

На первом шаге алгоритма выделения берутся три произвольные связные кусочно-плоскостные объекты, их выпуклость проверяется при помощи алгоритма определения выпуклости набора кусочно-плоскостных объектов. Для выбора трёх таких объектов следует также решить вопрос о топологии связей. В данном случае имеется лишь один вариант, расположения связей, представленный на рисунке 10. Выделенный кусочно-плоскостной объект берётся из начального множества полученных на предыдущем этапе плоскостей. Остальные два берутся уникальным образом из списка смежности выделенного объекта.

Рисунок 10 – Схематичное представление топологии связей между тремя кусочно-плоскостными объектами при выделении цилиндра.

На втором шаге требуется по трём выделенным плоскостям построить модель цилиндра, которого они касаются. Пусть на прямой – суть центральной оси цилиндра имеется некоторая точка $P_0(x_0, y_0, z_0)$. Тогда расстояние от каждой из трёх плоскостей до этой точки можно вычислить, аналогично с выкладками для сферы, по формуле (17).

$$r = |A'x_0 + B'y_0 + C'z_0 + D'| \tag{17}$$

Тогда, аналогично раскрытию модуля для сферы, и обозначив знак перед радиусом цилиндра r после раскрытия модуля в формуле (17) символом E' после некоторых перестановок, получается формула (18).

$$A'x_0 + B'y_0 + C'z_0 - E'r = -D' \tag{18}$$

Выписав для каждой из трёх плоскостей аналогичные уравнения, получается система (19).

$$\begin{aligned} A'_1x_0 + B'_1y_0 + C'_1z_0 - E'_1r &= -D'_1 \\ A'_2x_0 + B'_2y_0 + C'_2z_0 - E'_2r &= -D'_2 \\ A'_3x_0 + B'_3y_0 + C'_3z_0 - E'_3r &= -D'_3 \end{aligned} \tag{19}$$

Результатом применения прямого хода метода Жордано-Гаусса к системе (19) является система (20).

$$\begin{aligned} A''_1x_0 + B''_1y_0 + C''_1z_0 - E''_1r &= -D''_1 \\ B''_2y_0 + C''_2z_0 - E''_2r &= -D''_2 \\ C''_3z_0 - E''_3r &= -D''_3 \end{aligned} \tag{20}$$

Выразив r из последнего уравнения системы (20) получим равенство (21).

$$r = \frac{-D''_3 - C''_3z_0}{-E''_3} \tag{21}$$

Если C''_3 равно 0, то это значит, что радиус не зависит от координаты z_0 и плоскости, которые были выбраны, пересекаются по параллельным прямым. Естественно заметить, что такое, в принципе вряд ли возможно, на реальных данных следует оценить влияние C''_3 на r и, если оно невелико, считать, что C''_3 равно нулю. Далее, радиус выражается по формуле (22).

$$r = \frac{-D''_3}{-E''_3} \tag{22}$$

А саму прямую – центральную ось цилиндра можно выразить, если в два первых уравнения системы (19) подставить радиус из уравнения (22). В итоге получается система (23), которая представляет собой уравнения двух плоскостей, которые в пересечении дают прямую. Из данной системы уже можно получить точку на этой прямой и вектор направления этой прямой, чтобы использовать в дальнейшем.

$$A_1'x_0 + B_1'y_0 + C_1'z_0 - E_1'\frac{-D_3''}{-E_3''} = -D_1'$$
$$A_2'x_0 + B_2'y_0 + C_2'z_0 - E_2'\frac{-D_3''}{-E_3''} = -D_2' \qquad (23)$$

На третьем шаге алгоритма, аналогично тому, как это делается при выделении сферы, итеративно при помощи алгоритма Левенберга-Маркварда параметры цилиндра уточняются.

Для определения цилиндра используются следующие параметры: X - точка на оси цилиндра, A - направление оси цилиндра, r - радиус цилиндра. Чтобы определить остальные функции, необходимые для метода Левернберга-Маркварда, необходимо вначале определить некоторые вспомогательные функции.

Функция $f(X_i, X, A)$ определяет расстояние от точки X_i до прямой, определённой точкой X и направлением $a = A/|A|$. То есть, $f_i = f(X_i, X, A) = |a \times (X_i - X)|$, что может быть выражено формулой (24).

$$f_i = \sqrt{u^2 + v^2 + w^2}, \text{ где } \begin{matrix} \mathrm{u} = c(y_i - y) - b(z_i - z) \\ v = a(z_i - z) - c(x_i - x) \\ w = b(x_i - x) - a(y_i - y) \end{matrix} \qquad (24)$$

Это выражение для f_i, как показано в [9], используется потому, что обладает численной стабильностью.

Функция $g(X_i, X, A)$ определяет расстояние от точки X_i до плоскости, заданной точкой X и вектором нормали $a = A/|A|$. То есть, $g_i = g(X_i, X, A) = a(x_i - x) + b(y_i - y) + c(z_i - z)$.

Производные функций f_i и g_i представлены формулами (25).

$$\begin{aligned}
\frac{\partial g_i}{\partial x} &= -a & \frac{\partial f_i}{\partial x} &= [ag_i - (x_i - x)]/f_i \\
\frac{\partial g_i}{\partial y} &= -b & \frac{\partial f_i}{\partial y} &= [bg_i - (y_i - y)]/f_i \\
\frac{\partial g_i}{\partial z} &= -c & \frac{\partial f_i}{\partial z} &= [cg_i - (z_i - z)]/f_i \\
\frac{\partial g_i}{\partial A} &= (x_i - x) - ag_i & \frac{\partial f_i}{\partial A} &= g_i[ag_i - (x_i - x)]/f_i \\
\frac{\partial g_i}{\partial B} &= (y_i - y) - bg_i & \frac{\partial f_i}{\partial B} &= g_i[bg_i - (y_i - y)]/f_i \\
\frac{\partial g_i}{\partial C} &= (z_i - z) - cg_i & \frac{\partial f_i}{\partial C} &= g_i[cg_i - (z_i - z)]/f_i
\end{aligned} \qquad (25)$$

Производные определённые выше в правом столбце не определены в том случае, когда $f_i = 0$ (точка X_i лежит на прямой), в этом случае вектор градиенты представляет собой $\left(\sqrt{1-a^2}, \sqrt{1-b^2}, \sqrt{1-c^2}, g\sqrt{1-a^2}, g\sqrt{1-b^2}, g\sqrt{1-c^2}\right)$.

Расстояние от поверхности цилиндра до точки выражается формулой (26).

$$d(x_i) = f_i - r \tag{26}$$

Функция, которая выражает сумму квадратов расстояний от точек, до предполагаемой поверхности цилиндра, и, которую требуется минимизировать в методе наименьших квадратов, выражается формулой (27).

$$J(X, A, r) = \sum (f_i - r)^2 \tag{27}$$

Где сумма берётся по всем точкам, которые вошли в три кусочно-плоскостных объекта, по которым строится приближение фигуры.

Процесс нормализации промежуточных данных в методе Левенберга-Марквардa применительно к выделению цилиндра заключается в том, чтобы, во-первых, в качестве точки X взять точку на оси цилиндра ближайшую к началу координат, и, во-вторых, нормализовать вектор направления оси цилиндра: $A \leftarrow A/|A|$.

Производные по параметрам, задающим цилиндр, и использующиеся в методе наименьших квадратов приводятся в формулах (28).

$$\begin{aligned} \frac{\partial d_i}{\partial x} &= (f_i)'_x & \frac{\partial d_i}{\partial A} &= (f_i)'_A \\ \frac{\partial d_i}{\partial y} &= (f_i)'_y & \frac{\partial d_i}{\partial B} &= (f_i)'_B \\ \frac{\partial d_i}{\partial z} &= (f_i)'_z & \frac{\partial d_i}{\partial C} &= (f_i)'_C \\ \frac{\partial d_i}{\partial r} &= -1 \end{aligned} \tag{28}$$

После того, как параметры выделяемого цилиндра итеративно были уточнены, происходит переход на шаги 4 и 5 алгоритма, которые сохраняются такими же, для выделения цилиндра, как и в общем алгоритме.

4.5. Выделение конусов.

Для того чтобы выделить конус, необходимо в точности три плоскости, которые касаются его поверхности. И, естественно, информация о том, в какой части пространства (из частей, на которые делят пространство три плоскости) лежит конус. Такая информация

может быть получена из данных о плоскостных объектах, которые образуют данные плоскости.

На первом шаге требуется выбрать три выпуклых связных плоскости, этот вопрос решается для конусов так же, как и для цилиндров.

На втором шаге, по трём плоскостям требуется построить саму фигуру. Для построения конуса, вначале, необходимо определить параметры, с помощью которых он задаётся. Это X - точка на оси конуса, но не вершина, A - вектор направления оси (направленный в сторону вершины), s - расстояние от точки X до поверхности конуса, ψ - полуугол при вершине конуса. Пусть на прямой – суть центральной оси конуса имеется некоторая точка $P_0(x_0, y_0, z_0)$. Тогда расстояние от каждой из трёх плоскостей до этой точки можно вычислить, аналогично с выкладками для сферы, по формуле (29).

$$s = \left| A'x_0 + B'y_0 + C'z_0 + D' \right| \tag{29}$$

Тогда, аналогично раскрытию модуля для сферы, и обозначив знак перед расстоянием до этой точки s после раскрытия модуля в формуле (29) символом E' после некоторых перестановок, получается формула (30).

$$A'x_0 + B'y_0 + C'z_0 - E's = -D' \tag{30}$$

Выписав для каждой из трёх плоскостей аналогичные уравнения, получается система (31).

$$\begin{aligned} A'_1x_0 + B'_1y_0 + C'_1z_0 - E'_1s &= -D'_1 \\ A'_2x_0 + B'_2y_0 + C'_2z_0 - E'_2s &= -D'_2 \\ A'_3x_0 + B'_3y_0 + C'_3z_0 - E'_3s &= -D'_3 \end{aligned} \tag{31}$$

Применив прямой ход метода Жордано-Гаусса к системе (31) получается система (32).

$$\begin{aligned} A''_1x_0 + B''_1y_0 + C''_1z_0 - E''_1s &= -D''_1 \\ B''_2y_0 + C''_2z_0 - E''_2s &= -D''_2 \\ C''_3z_0 - E''_3s &= -D''_3 \end{aligned} \tag{32}$$

В которой, если C''_3 равно 0, то это значит, что s не зависит от координаты z_0 и плоскости, которые были выбраны, пересекаются по параллельным прямым. Из чего следует, что выделить конус по этим трём плоскостям невозможно. Вплоть до текущего момента выкладки для выделения цилиндра и для выделения конуса совпадают. На данном моменте в зависимости от того, насколько близким к нулю является C''_3, делается выбор о необходимости выделения конуса и цилиндра. Пусть C''_3 не равно нулю, тогда имеет смысл выделять конус.

Полагая $s = 0$ и решая систему (31) получается некоторая точка, которая является точкой пересечения трёх плоскостей, а также вершиной конуса. Далее, полагая $s = 1$ и решая ту же систему, получается точка, которая лежит на расстоянии 1 от всех трёх плоскостей, а также лежит на центральной оси цилиндра. Обозначим эту точку, X, тогда вектор направления A, необходимый для задания конуса можно вычислить, найдя разность между вершиной конуса и точкой X. Для того, чтобы определить оставшийся параметр ψ необходимо составить треугольник. Первой вершиной, которого, является вершина конуса (обозначим вершину конуса Y), второй вершиной является точка X и третьей вершиной является точка пересечения перпендикуляра опущенного из точки X на одну из плоскостей. Угол, при последней вершине в этом треугольнике будет прямым. Полуугол ψ, при вершине конуса будет равен углу данного треугольника, при вершине Y. Поскольку все три точки треугольника известны, его можно найти. Конус определён.

Третий шаг уточнения параметров конуса аналогичен третьему шагу для цилиндра и сферы. Далее приводятся некоторые формулы, необходимые для метода Левенберга-Марквардa.

Расстояние от поверхности конуса до точки выражается формулой (33).

$$d(x_i) = f_i \cos(\psi) + g_i \sin(\psi) - s \tag{33}$$

Где функции f_i и g_i, а также их производные определены выше в описании выделения цилиндра. Функция, которая выражает сумму квадратов расстояний от точек, до предполагаемой поверхности конуса, и, которую требуется минимизировать в методе наименьших квадратов, выражается формулой (34).

$$J(X, A, r) = \sum (f_i \cos(\psi) + g_i \sin(\psi) - s)^2 \tag{34}$$

Где сумма берётся по всем точкам, которые вошли в три кусочно-плоскостных объекта, по которым строится приближение фигуры.

Процесс нормализации промежуточных данных в методе Левенберга-Марквардa применительно к выделению конуса заключается в следующем:

1) в качестве точки X берётся точка на оси конуса ближайшая к началу координат;

2) нормализуется вектор направления оси конуса $A \leftarrow A/|A|$;

3) $\psi \leftarrow \psi (\mathrm{mod}\, 2\pi)$;

4) если $\psi > \pi$, то $[\psi \leftarrow \psi (\mathrm{mod}\, \pi); A \leftarrow -A]$;

5) если $\psi > \frac{\pi}{2}$, то $\psi \leftarrow \pi - \psi$;

6) если $s < 0$, то $[s \leftarrow -s; A \leftarrow -A]$.

Производные, по параметрам, задающим конус, выражаются следующими формулами (35).

$$\begin{aligned}
\frac{\partial d_i}{\partial x} &= (f_i)'_x \cos(\psi) + (g_i)'_x \sin(\psi) & \frac{\partial d_i}{\partial A} &= (f_i)'_A \cos(\psi) + (g_i)'_A \sin(\psi) \\
\frac{\partial d_i}{\partial y} &= (f_i)'_y \cos(\psi) + (g_i)'_y \sin(\psi) & \frac{\partial d_i}{\partial B} &= (f_i)'_B \cos(\psi) + (g_i)'_B \sin(\psi) \\
\frac{\partial d_i}{\partial z} &= (f_i)'_z \cos(\psi) + (g_i)'_z \sin(\psi) & \frac{\partial d_i}{\partial C} &= (f_i)'_C \cos(\psi) + (g_i)'_C \sin(\psi) \\
\frac{\partial d_i}{\partial s} &= -1 & \frac{\partial d_i}{\partial \psi} &= -f_i \sin(\psi) + g_i \cos(\psi)
\end{aligned} \tag{35}$$

После того, как параметры выделяемого конуса итеративно были уточнены, происходит переход на шаги 4 и 5 алгоритма, которые сохраняются такими же, для выделения цилиндра, как и в общем алгоритме.

4.6. Выделение торов.

К сожалению, тор, не является выпуклой фигурой, поэтому выделение его по плоскостям затруднительно. Поэтому, на данном этапе предлагается один из трёх вариантов:

1) вообще не рассматривать тор, поскольку куски тора, в общем случае встречаются достаточно редко (около 10% случаев);

2) не проводить его выделение по плоскостям, а выделять так же, как это делается в случае выделения плоскостей;

3) брать некоторое количество соседних плоскостей, например 3-4, учитывая топологию выбора, так, как это делается при выделении сферы или цилиндра и непосредственно переходить к третьему шагу после первого.

Далее приводятся формулы для выделения тора, необходимые на третьем шаге алгоритма. Тор может быть определён, используя следующие четыре параметра: X - центр тора, A - вектор направления оси тора, r - малый радиус тора и R большой радиус тора. Расстояние между точкой и поверхностью тора определяется формулой (36).

$$d(x_i) = \sqrt{g_i^{\,2} + (f_i - r)^2} - R \tag{36}$$

Функция, которую требуется минимизировать в методе наименьших квадратов, представлена формулой (37).

$$J(X, A, r, R) = \sum \left[\sqrt{g_i^{\,2} + (f_i - r)^2} - R \right]^2 \tag{37}$$

Для нормализации, при выделении тора достаточно нормализовать вектор $A \leftarrow A/|A|$. Производные по параметрам, определяющим тор, представлены формулами (38).

$$\frac{\partial d_i}{\partial x} = \left[g_i (g_i)'_x + (f_i - r)(f_i)'_x\right]/(d_i + R)$$
$$\frac{\partial d_i}{\partial y} = \left[g_i (g_i)'_y + (f_i - r)(f_i)'_y\right]/(d_i + R)$$
$$\frac{\partial d_i}{\partial z} = \left[g_i (g_i)'_z + (f_i - r)(f_i)'_z\right]/(d_i + R)$$
$$\frac{\partial d_i}{\partial A} = \left[g_i (g_i)'_A + (f_i - r)(f_i)'_A\right]/(d_i + R)$$
$$\frac{\partial d_i}{\partial B} = \left[g_i (g_i)'_B + (f_i - r)(f_i)'_B\right]/(d_i + R)$$
$$\frac{\partial d_i}{\partial C} = \left[g_i (g_i)'_C + (f_i - r)(f_i)'_C\right]/(d_i + R)$$
$$\frac{\partial d_i}{\partial r} = -(f_i - r)(d_i + R)$$
$$\frac{\partial d_i}{\partial R} = -1 \qquad (38)$$

После того, как параметры тора определены, выделение самого тора происходит аналогично тому, как это делается в общем алгоритме, то есть шаги 4 и 5 остаются без изменений.

5. Реализация алгоритма

5.1. Классы для хранения и обработки данных

Для реализации данного алгоритма были спроектированы и реализованы классы, отражённые в таблице 2.

Таблица 2 Классы для хранения и обработки данных.

Название класса	Реализован	Описание
PPoint	да	Хранит координаты и цвет каждой трёхмерной точки, имеет методы для рисования данной точки в цвете и без цвета.
PTriangle	да	Хранит ссылки на три точки, образующие треугольник, а также три ссылки на соседние треугольники. Имеет методы рисования в цвете и без цвета.
PComponent	да	Виртуальный класс компоненты (плоскость, сфера, цилиндр и т.п.). Содержит информацию о цвете компоненты, виртуальный метод определения параметров компоненты по заданному массиву точек. Также содержит общую реализацию алгоритма Левенберга-Маркварда для таких фигур как сфера, цилиндр, конус и тор. Формулы, специфичные для каждого типа фигуры вычисляются в соответствующих классах – наследниках.
PPlane	да	Класс наследник PComponent. Представляет собой компоненту-плоскость. Содержит реализацию выделения плоскостей по множеству точек, описанному в разделе 3.3.
PSphere	да	Класс наследник PComponent. Представляет собой компоненту-сферу. Содержит в себе реализацию выделения сферы по четырём плоскостям.
PCone	нет	Класс наследник PComponent. Представляет собой компоненту-конус. Содержит в себе реализацию выделения конуса по трём плоскостям.
PCylinder	Нет	Класс наследник PComponent. Представляет собой компоненту-цилиндр. Содержит в себе реализацию выделения цилиндра по трём плоскостям.
PTorus	Нет	Класс наследник PComponent. Представляет собой компоненту-тор. Содержит функции для использования в алгоритме Левенберга-Маркварда для выделения тора.
CopenGLControl	да	Класс, хранящий методы для обнаружения трехмерных объектов, их рисования и работы с графикой.

Все ссылки на треугольники, плоскости и точки, фактически являются индексами в соответствующих глобальных массивах. Массивы представлены векторами STL, которые являются динамическими структурами и могут перераспределять память, поэтому использование ссылок запрещено.

```
//массив точек
vector <PPoint> points;
//массив треугольников
vector <PTriangle> triangles;
//массив компонент
vector <PPlane> planes;
```

Далее приводятся описания некоторых реализованных классов на языке. C++

```
class PPoint
{
public:
    //Пустой конструктор
    PPoint(void);
    //В конструктор передаётся координата (x,y,z)
    PPoint(double, double, double);
    //В конструктор передаётся координата и цвет точки
    PPoint(double, double, double, unsigned char r,
unsigned char g, unsigned char b);
    PPoint& operator -=(const PPoint &pt);
    PPoint& operator +=(const PPoint &pt);
    PPoint& operator /=(double);
    PPoint& operator *=(double);
    ~PPoint(void);
    double x,y,z;
    unsigned char r,g,b;
    //печать точки средствами OpenGL
    void Print(void);
    //печать точки в цвете
    void ColorPrint(void);
};

class PTriangle
{
public:
    //Пустой конструктор
    PTriangle(void);
    //Конструктор с сылками на три точки и соседние
треугольники
    PTriangle(int, int, int, int, int, int);
    ~PTriangle(void);
    //ссылки на вершины треугольника
    int apex[3];
    //ссылки на треугольники-соседи
    int triangle[3];
    //флаг показывающий был ли выведен данный треугольник
    int output_flag;
```

```
    //индекс кусочно-плоскостного объекта
    int surface;
    //рисование треугольника
    void Print(void);
    //рисование треуголькика в цвете
    void ColorPrint(void);
};

class PComponent
{
public:
    PComponent(void);
    // находящий ближайшую к данным точкам поверхность
    virtual void Determine(PointsSet* pts);
    // находит кратчайшее расстояние от точки до данной
поверхности
    virtual double Distance(int pt);
    ~PComponent(void);
    // индекс поверхности
    int surface;
     // первый треугольник, который принадлежит данной
поверхности
    int firsttriangle;
    // смежные кусочно-плоскостные объекты
    set<int> adjacent;
    // цвет поверхности
    unsigned char r,g,b;
     //в алгоритме Левенберга-Марквардa требуется
нормализация, поэтому данная функция переопределена в
каждом классе-потомке. Все параметры, которые требуется
нормализовать хранятся в объекте класса.
    virtual void Normalize(void);
     //Общая реализация метода Левенберга-Марквардa
    void LevenbergMarquardt(double *p0, PointsSet *ps, int
params);
     //Нормализация параметров, необходимая в алгоритме
Левенберга-Марквардa, параметры фигуры передаются в
качестве параметров функции.
    virtual void Normalize(double * p);
     //Расстояние от точки до поверхности, заданной
параметрами фигуры передающимися в массиве p
    virtual double Distance(int pt, double * p);
     //Вычисляет вектор расстояний от точек до поверхности
фигуры, параметры которой задаются массивом p0
    virtual void Distance(PointsSet * ps, double * p0,
double * d);
     //Используя массив исходных точек и параметры фигуры,
передающиеся в p0, функция вычисляет частные производные по
параметрам фигуры, данный метод используется в алгоритме
Левенберга-Марквардa
    virtual void Derivative(PointsSet * ps, double * p0,
double ** dd);
```

```
    //Функция возвращает массив точек, принадлежащих данной
поверхности
    vector<int> GetPoints();
};
```

Следует отметить, что последний класс является базовым для всех фигур, в том числе плоскости. Виртуальные функции переопределены в потомках, и содержат различное внутреннее содержание, в зависимости от фигуры. Поскольку и плоскости и иные фигуры являются потомками класса PComponent, многие функции не определены для плоскостных объектов и наоборот.

5.2. Описание системы

Поскольку выделение кусочно-плоскостных объектов является отдельным этапом всего алгоритма, и заслуживает отдельного внимания, реализация всего алгоритма была разделена на две версии. Обе версии прилагаются в отдельных папках. В одной версии происходит выделение плоскостей, в другой алгоритм представлен полностью – с выделением сфер. Далее приводится общее описание системы.

Как видно из описания классов, алгоритм обнаружения объектов расположен в функции Recognize() класса COpenGLControl. На рисунке 11 представлен внешний вид программы. Окно слева, которое располагается на окне диалога, представляет собой окно OpenGL, вся работа с которым (включая полосы прокрутки) происходит внутри COpenGLControl. COpenGLControl содержит методы для обнаружения объектов и методы для рисования сцены. Класс CMfc5Dlg представляет собой класс для работы с интерфейсом и передаёт в объект класса COpenGLControl следующие данные:

Значения Checkboxов «Points», «Facets», «Planes», которым соответствуют члены класса COpenGLControl view_points, view_facets, view_planes.

Значение поля «Triangles to build object», которому соответствует переменная points_to_build.

Значение поля «Triangles to separate object», которому соответствует переменная points_to_separate.

Значение поля «Tolerance», которому соответствует переменная tolerance.

По нажатию кнопки «Load Points» вызывается функция, отвечающая за событие BN_CLICK в классе CMfc5Dlg, которая вызывает диалог для открытия файла, а затем открывает выбранный файл и читает из него данные.

По нажатию кнопки «Recognize» вызывается функция, отвечающая за событие BN_CLICK в классе CMfc5Dlg, которая вызывает функцию Recognize() из класса COpenGLControl.

На рисунке 11 представлена версия программы, которая распознаёт по входной поверхностной триангуляции кусочно-плоскостные объекты. На рисунке 12 – версия программы, распознающая сферические объекты по поверхностной триангуляции с использованием алгоритма выделения кусочно-плоскостных объектов.

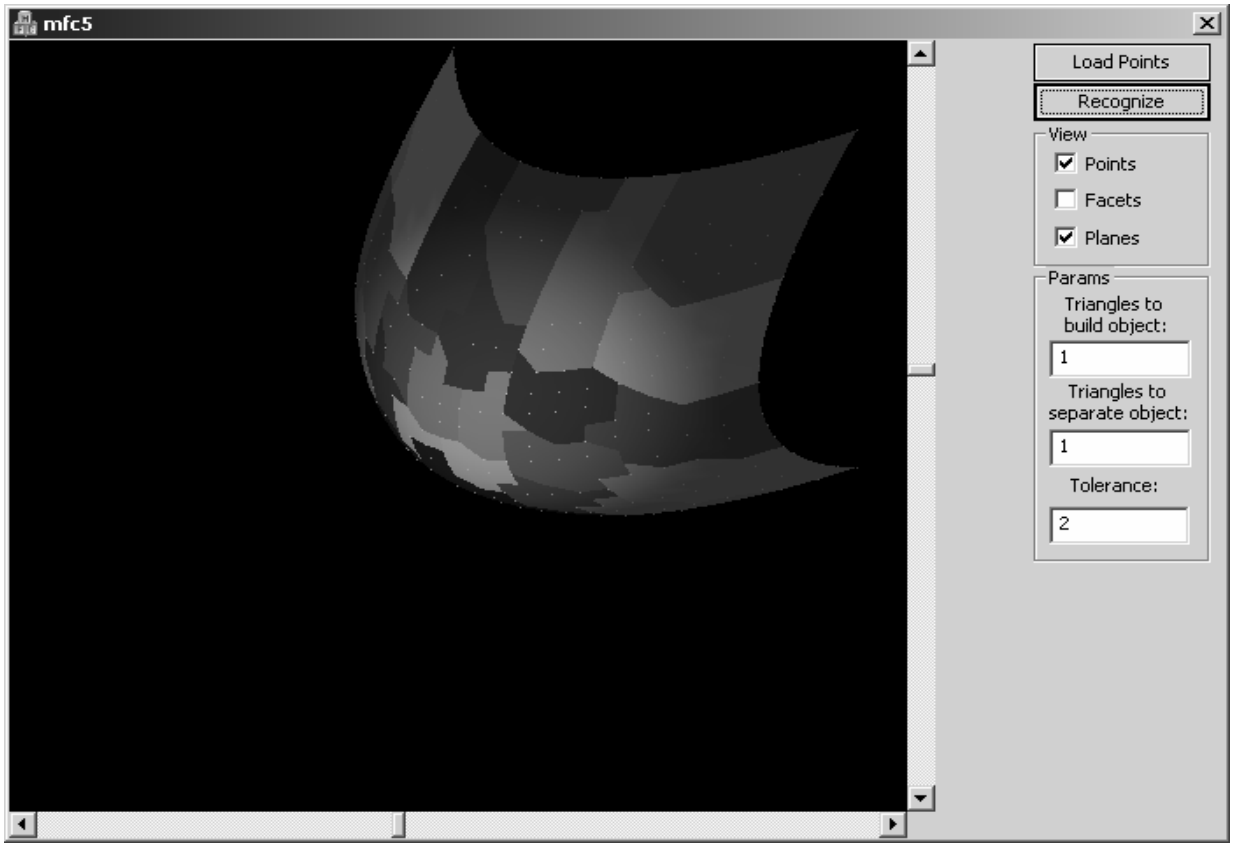

Рисунок 11 - Внешний вид программы обнаружения кусочно-плоскостных объектов по поверхностной триангуляции.

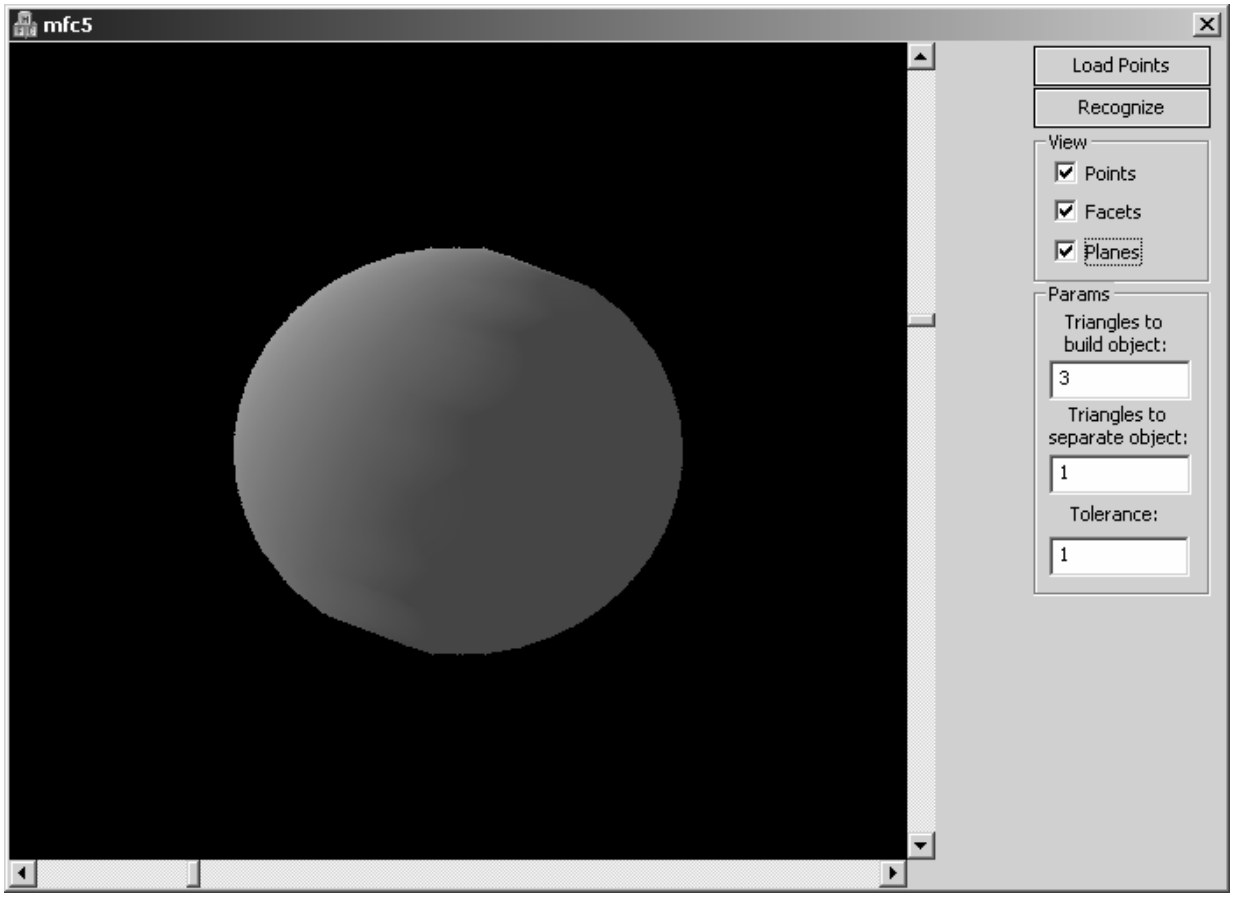

Рисунок 12 – Внешний вид программы выделения сфер по поверхностной триангуляции.

5.3. Формат файла входных данных

Формат файла данных (расширение *.sur от «surface») следующий:
1 строка содержит единственное целое число n – число точек во входном множестве
2 строка содержит единственное целое число m – число треугольников, которые образуют поверхностную триангуляцию исходного множества точек.
n строк – содержат описание каждой точки: три вещественных координаты x, y, z.
m строк – содержат описание каждого треугольника, входящего в поверхностную триангуляцию: три целых числа – индексы трех точек, составляющих данный треугольник (индексы нумеруются с 0). Порядок точек во входном файле определяет их индексы. То есть первая точка получает индекс 0, вторая – 1 и т.д.

5.4. Использование библиотеки линейной алгебры

В данной программе обнаружения объектов используется алгоритм сингулярного разложения, реализованный в этой библиотеке линейной алгебры [10], с помощью которого, удаётся по множеству точек построить аппроксимирующую плоскость с точки зрения метода наименьших квадратов. В программе эта библиотека используется как статическая: в отдельном проекте собирается библиотека в файл nl.lib, который позднее подключается к проекту вместе с заголовочным файлом nl.h, однако, учитывая то, что библиотека написана на языке Си, для её включения требуется указать компилятору, что вызовы функций реализованы в стиле Си:

```
extern "C"
{
#include "nl.h"
}
```

В документации к библиотеке приведено множество различных примеров и описание работы библиотеки. Далее приводится пример использования библиотеки в приложении обнаружения объектов:

```
double **M, **U, **V, *w;

//выделяет память под матрицы
M = nl_dmatrix_create(m, n);
U = nl_dmatrix_create(m, n);
V = nl_dmatrix_create(n, n);
//выделяет память под вектор
w = nl_dvector_create(n);
//singular value decomposition
svd_decomp(M, m, n, w, 0, U, 1, V, &ierr);
//Какие-то действия с разложением (определение коэффиициентов для
задания плоскости – A, B, C, D)
...
//
//освобождает память
nl_dmatrix_free(M, m);
nl_dmatrix_free(U, m);
nl_dmatrix_free(V, n);
nl_dvector_free(w);
```

6. Описание применения системы

6.1. Пример распознавания фигур

Для распознавания плоскостей или сфер следует выбрать соответствующую версию программы и проделать следующие шаги:

1. Открыть приложение, запустив mfc5.exe
2. Нажав на кнопку «Open File» выбрать необходимый файл с данными точек и поверхностной триангуляции. После работы нет необходимости закрывать файл, достаточно закрыть приложение или открыть другой файл с данными.
3. Чтобы отобразить исходное множество точек нужно отметить галочкой справа в группе «View» (см. рис. 11) поле «Points». Чтобы отобразить поверхностную триангуляцию – отметить поле «Facets». Чтобы отобразить выделенные поверхности – поле «Planes». Изначально, после загрузки данных из файла, множество выделенных поверхностей пусто, поэтому они не отображаются, для того, чтобы их выделить, нужно нажать копку «Recognize»
4. Чтобы приступить к поиску изначально требуется установить параметры поиска. Triangles to build object – количество треугольников, которое необходимо для того, чтобы построить по ним аппроксимирующую плоскость. Triangles to separate object – минимальное количество треугольников, которое нужно, чтобы выделить кусочно-плоскостной объект. «Tolerance» - погрешность измерений. Для того чтобы считать треугольник принадлежащим плоскости, необходимо, чтобы все его три точки лежали на расстоянии, не превышающем Tolerance от неё.
5. После установки параметров можно приступить к выделению объектов, для этого необходимо нажать кнопку «Recognize».

Заключение

В процессе выполнения работы были решены следующие задачи и проблемы:

1. Проблема нахождения некоторого начального приближения в методе наименьших квадратов, при поиске решения для нелинейных поверхностей. В данной работе впервые приводится идея выделения таких поверхностей в два этапа – через кусочно-плоскостные объекты. Что позволяет довольно просто найти хорошие начальные приближения.

2. Проблема наличия в исходных данных шумов. В предложенном алгоритме предусматривается выделение объекта через соседние треугольники, используя различные виды связности, которые позволяют пропускать некоторые вершины – которые были случайно получены и фактически являются шумом.

3. Задача выделения фрагментов фигур, а не фигур целиком. Предложенный алгоритм предназначен для выделения именно не связных между собой частей разнообразных фигур. Границы и поверхность таких частей задаются множеством треугольников поверхностной триангуляции.

4. Проблема временной сложности. Исходным является множество, которое может содержать до сотен миллионов точек, поэтому трудоёмкость алгоритмов в этой области является критичной. Трудоёмкость первого этапа алгоритма является линейной от количества исходных точек. На втором этапе работа ведётся уже с кусочно-плоскостными объектами, которых гораздо меньше, чем точек. К тому же для построения аппроксимаций сложных фигур используются только соседние кусочно-плоскостные объекты, что резко снижает трудоёмкость выделения.

5. Важной задачей является разработка алгоритмов, которые бы были достаточно независимы от человека, то есть не нуждались в подсказках «эксперта» или нуждались в них только в очень редких случаях. Данный алгоритм выделяет куски разнообразных объектов, при том, после выделения не проводит классификацию между различными типами поверхностей. То есть, отдельно существуют выделенные куски плоскостей и куски сфер, которые, вообще говоря, могут пересекаться. Поэтому, провести анализ количества спорных ситуаций, при которых следует обращаться к «эксперту», представляется невозможным. Однако в алгоритме имеются все предпосылки для того, чтобы снизить количество таких обращений. То есть, в результате алгоритма имеется достаточно корректных данных (различных выделенных объектов и их описаний) чтобы на следующем этапе сделать адекватный выбор.

По результатам работ опубликован ряд тезисов конференций: [8], [11] и [12]. За участие в одной из конференций [11] автор был отмечен дипломом третьей степени «за содержательный доклад».

Список использованных источников

1. GeoKosmos – Воздушное лазерное сканирование [Электронный ресурс]: Сайт. – Режим доступа: http://www.geokosmos.ru/news/articles/area/, свободный.

2. GeoKosmos – Наземное лазерное сканирование [Электронный ресурс]: Сайт. – Режим доступа: http://www.geokosmos.ru/news/articles/earth/, свободный.

3. Sithole G., Vosselman G. Comparison of filtering algorithms // 3-D reconstruction from airborne laserscanner and InSAR data. [Электронный ресурс]: Материалы конференции. – 2003.-Session 4 – Режим доступа: http://www.isprs.org/commission3/wg3/workshop_laserscanning/, свободный

4. Sithole G., Vosselman G. Filtering of airborne laser scanner data based on segmented point clouds // Laser Scanning 2005. [Электронный ресурс]: Материалы конференции. – 2005.-Session 4 – Режим доступа: http://www.commission3.isprs.org/laserscanning2005/, свободный

5. Voegtle T., Steinle E. On the quality of object classification and automated building modeling based on laserscanning data // 3-D reconstruction from airborne laserscanner and InSAR data. [Электронный ресурс]: Материалы конференции. – 2003.-Session 5 – Режим доступа: http://www.isprs.org/commission3/wg3/workshop_laserscanning/, свободный

6. Rabbani T., Heuvel F. Efficient hough transform for automatic detection of cylinders in point clouds // Laser Scanning 2005. [Электронный ресурс]: Материалы конференции. – 2005.-Session 5 – Режим доступа: http://www.commission3.isprs.org/laserscanning2005/, свободный

7. Verbree E., Oosterom P. The STIN method: 3d-surface reconstruction by observation lines and Delaunay TENs // 3-D reconstruction from airborne laserscanner and InSAR data. [Электронный ресурс]: Материалы конференции. – 2003.-Session 5 – Режим доступа: http://www.isprs.org/commission3/wg3/workshop_laserscanning/, свободный

8. Костюк Ю.Л., Пешехонов С.В. Распознавание объектов определённой структуры по данным дистанционного зондирования с использованием поверхностной триангуляции // Исследование, разработка и применение высоких технологий в промышленности (материалы межд. конф.), Т.6. Гл.8. – СПб., 2006. – С.238

9. Shakarji C.M. Least-Squares Fitting Algorithms of the NIST Algorithm Testing System // Journal of Research of the National Institute of Standards and Technology. – 1998. – V. 103, - №6, - P.633-635 [Электронный ресурс]: - Режим доступа: http://nvl.nist.gov/pub/nistpubs/jres/103/6/j36sha.pdf, свободный.

10. Тагунов А.М. Золотых Н.Ю. Библиотека NL – Numerical Library [Электронный ресурс]: Сайт. – Режим доступа:http://www.uic.nnov.ru/~zny/nl/doc/html/index.html, свободный.

11. Пешехонов С.В. Обнаружение кусочно-плоскостных объектов по данным лазерного зондирования с использованием поверхностной триангуляции. // Научная сессия ТУСУР-2006 (материалы докладов Всероссийской научно-технической конференции студентов, аспирантов и молодых учёных, Томск, 4-7 мая 2006г.), Ч. 2.- Томск: Издательство «В-Спектр», 2006. –С.87

12. Пешехонов С.В. Обнаружение кусочно-плоскостных объектов по данным дистанционного зондирования с использованием поверхностной триангуляции //

Научное творчество молодёжи: Матералы X Всероссийской научно-практической конференции (21-22 апреля 2006 г.) Ч.1 – Томск: Изд-во Том. ун-та, 2006. – С. 174-175

13. Киммел П. Borland C++ 5: пер. с англ. – СПб.: BHV – Санкт-Петербург, 1999. – 976 с., ил.

14. Препарата Ф., Шеймос М. Вычислительная геометрия: Введение: Пер. с англ. – М.: Мир, 1989. – 478 с.

15. Тихомиров Ю. Программирование трехмерной графики – СПб.: BHV – Санкт-Петербург, 1998. – 256 с., ил.

Приложение А. Список исходных и исполняемых файлов программы

Исходные файлы

globaldata.h – содержит глобальные массивы точек, треугольников и кусочно-плоскостных объектов

mfc5.cpp – основной класс приложения

mfc5.h – основной класс приложения, заголовочный файл

mfc5dlg.cpp – реализация класса диалогового окна

mfc5dlg.h – описание класса диалогового окна

nl.h – заголовочный файл для библиотеки линейной алгебры

nl.lib – библиотека линейной алгебры

openglcontrol.cpp – реализация методов класса окна OpenGL, а также алгоритм обнаружения объектов

openglcontrol.h – заголовок класса

opengldevice.cpp – реализация вспомогательного класса для инициализации OpenGL

opengldevice.h – заголовок вспомогательного класса

pcomponent.cpp – реализация методов виртуального класса «компонента» или «поверхность»

pcomponent.h – заголовок класса

pplane.cpp – реализация класса «плоскость»

pplane.h – заголовок класса

ppoint.cpp – реализация класса «точка»

ppoint.h – заголовок класса

ptriangle.cpp – реализация класса «треугольник»

ptriangle.h – заголовок класса

psphere.cpp – реализация класса «сфера»

psphere.h – заголовок класса

stdafx.cpp – прекомпилируемые заголовки модулей

stdafx.h – прекомпилируемые заголовки модулей

Исполняемые файлы

mfc5.exe – основной файл приложения