

# Алгоритмы регистрации облаков точек

---

Немеш Норберт

6 сентября 2024 г.

# Введение

---

В последнее десятилетие стали достаточно распространены и доступны устройства для получения трехмерных изображений.

# Трёхмерные изображения

В последнее десятилетие стали достаточно распространены и доступны устройства для получения трёхмерных изображений.

Основное преимущество — более точное описание объектов.

# Трехмерные изображения

В последнее десятилетие стали достаточно распространены и доступны устройства для получения трехмерных изображений.

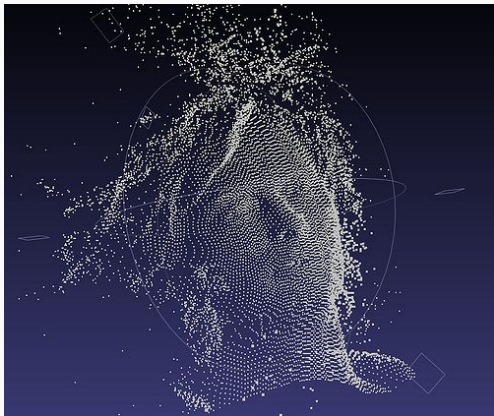
Основное преимущество — более точное описание объектов.

Применения:

- Восстановление 3D моделей
- Инспекция состояния зданий
- Локализация в робототехнике
- ...

Более распространенное название — облако точек.

Более распространенное название — облако точек.



## Облака точек. Определение

Точка — кортеж чисел содержащий координаты точки и возможно некоторые дополнительные атрибуты учитывающие специфику сенсора. Координаты точки даются в системе координат сенсора.



# Облака точек. Определение

Точка — кортеж чисел содержащий координаты точки и возможно некоторые дополнительные атрибуты учитывающие специфику сенсора. Координаты точки даются в системе координат сенсора.

Пример: координаты  $x$ ,  $y$ ,  $z$  и цвет в формате *RGB*.

# Облака точек. Определение

Точка — кортеж чисел содержащий координаты точки и возможно некоторые дополнительные атрибуты учитывающие специфику сенсора. Координаты точки даются в системе координат сенсора.

Пример: координаты  $x$ ,  $y$ ,  $z$  и цвет в формате *RGB*.

Облако точек — множество точек и возможно некоторая дополнительная информация учитывающая специфику сенсора из которого получено облако.

$$C = (\{p_1, \dots, p_n\}, \dots)$$

Пример дополнительной информации: порядок точек в облаке.

# Типы облаков точек

---

# Способы получения облаков точек

Структура облака и свойства точек зависят от способа получения.

Структура облака и свойства точек зависят от способа получения.

Пассивный способ — используем свет приходящий от объекта и с помощью дополнительных вычислений строим облако точек.

# Способы получения облаков точек

Структура облака и свойства точек зависят от способа получения.

Пассивный способ — используем свет приходящий от объекта и с помощью дополнительных вычислений строим облако точек.

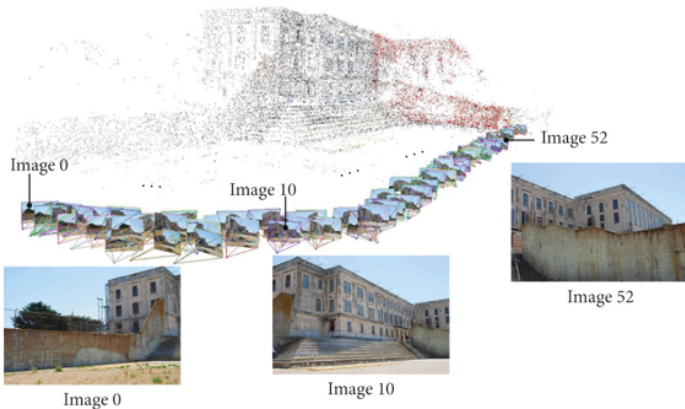
Активный способ — излучаем свет и по отражениям восстанавливаем облако точек.

## Пассивный способ. Structure from motion

Движемся вокруг объекта и делаем снимки, по изменению положения ключевых точек на изображении и известной собственной траектории восстанавливаем облако

# Пассивный способ. Structure from motion

Движемся вокруг объекта и делаем снимки, по изменению положения ключевых точек на изображении и известной собственной траектории восстанавливаем облако



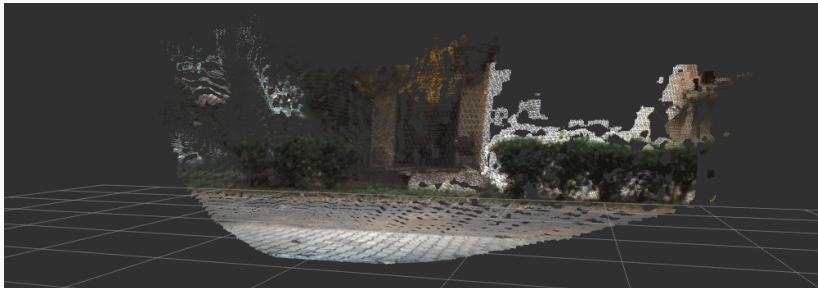


## Пассивный способ. Multi view stereo

Получаем изображение с двух близкорасположенных камер.  
Зная взаимное расположение камер и их внутренние характеристики восстанавливаем облако точек.

## Пассивный способ. Multi view stereo

Получаем изображение с двух близкорасположенных камер.  
Зная взаимное расположение камер и их внутренние характеристики восстанавливаем облако точек.



## Активный способ. Time of flight

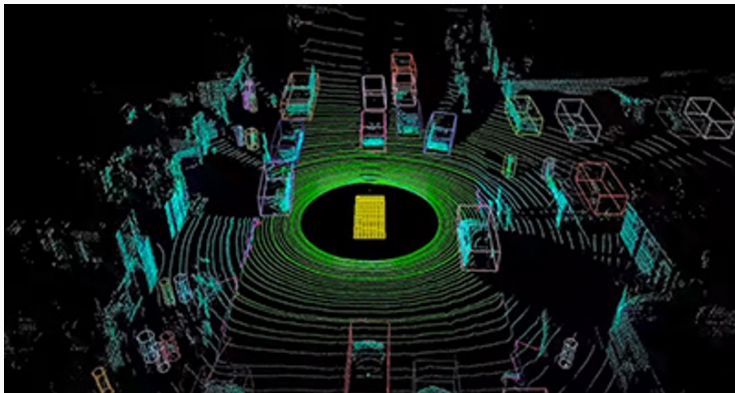
Выпускаем в различных направлениях сигналы, находим время до возврата, оцениваем расстояние до препятствия и восстанавливаем облако точек.

Пример: лидар

# Активный способ. Time of flight

Выпускаем в различных направлениях сигналы, находим время до возврата, оцениваем расстояние до препятствия и восстанавливаем облако точек.

Пример: лидар

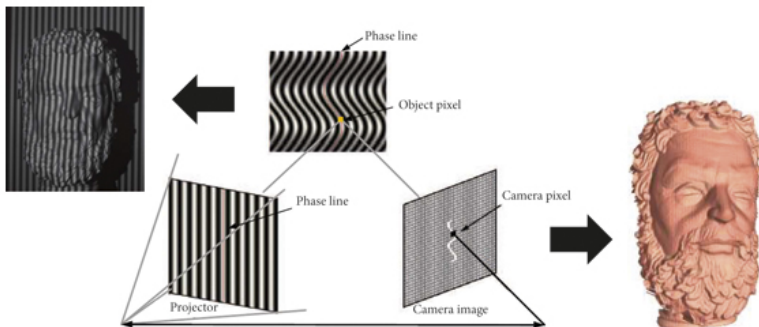


## Активный способ. Structured light

Проецируем на объект свет определенного формата, по искажениям восстанавливаем облако точек.

## Активный способ. Structured light

Проецируем на объект свет определенного формата, по искажениям восстанавливаем облако точек.



## Ограничения:

- Окклюзии
- Плохое покрытие точками на больших расстояниях
- Ограниченный угол обзора
- Шумные показания сенсоров, неточные координаты
- У лидаров могут быть лишние точки, из-за отраженных лучей

# Регистрация облаков точек

---

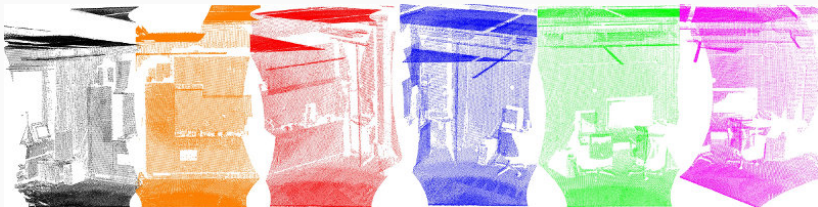


# Проблемы описания окружения

Проблема: одного облака недостаточно, надо получать трехмерные изображения с разных позиций.

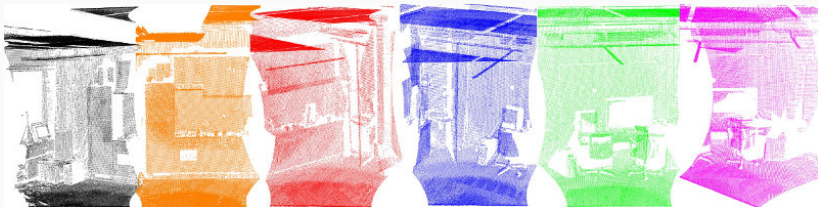
# Проблемы описания окружения

Проблема: одного облака недостаточно, надо получать трехмерные изображения с разных позиций.



# Проблемы описания окружения

Проблема: одного облака недостаточно, надо получать трехмерные изображения с разных позиций.



Решение: “склеиваем” точки в некоторой общей системе координат.

Более строго этот процесс называется регистрацией облаков.

## Постановка задачи

У нас есть два облака точек. Мы хотим найти такое преобразование, которое “наиболее точно встраивало” бы одно облако в другое.

# Постановка задачи

У нас есть два облака точек. Мы хотим найти такое преобразование, которое “наиболее точно встраивало” бы одно облако в другое.

Применения:

- Локализация робота в карте
- Локализация и одновременное построение карты (Simultaneous localization and mapping)
- Виртуальная и дополненная реальность

# Постановка задачи

У нас есть два облака точек. Мы хотим найти такое преобразование, которое “наиболее точно встраивало” бы одно облако в другое.

Применения:

- Локализация робота в карте
- Локализация и одновременное построение карты (Simultaneous localization and mapping)
- Виртуальная и дополненная реальность

Каждое облако задано в системе координат сенсора на момент получения данных. Достаточно найти преобразования между этими системами координат.

## Классификация регистраций. По количеству облаков

Попарная регистрация — нужно зарегистрировать одно облако точек в другом

## Классификация регистраций. По количеству облаков

Попарная регистрация — нужно зарегистрировать одно облако точек в другом

Множественная регистрация — есть более двух облаков точек, нужно собрать их вместе в одно большое облако точек.

Наивным образом сводится к попарной регистрации.



Точная регистрация — ошибка должна быть минимальна

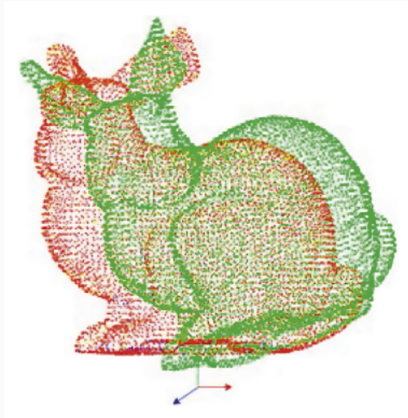
Точная регистрация — ошибка должна быть минимальна

Грубая регистрация — разрешается большая ошибка. Грубая регистрация обычно используется как начальное приближение для точной.

Жесткая регистрация — преобразование между облаками точек должно сохранять расстояния. Чаще всего ищут жесткую регистрацию.

# Классификация регистраций. По свойствам объекта

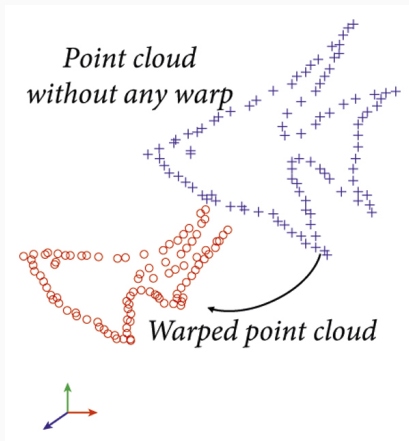
Жесткая регистрация — преобразование между облаками точек должно сохранять расстояния. Чаще всего ищут жесткую регистрацию.



Нежесткая регистрация — преобразование между облаками точек может дополнительно использовать растяжение и инверсию.

# Классификация регистраций. По свойствам объекта

Нежесткая регистрация — преобразование между облаками точек может дополнительно использовать растяжение и инверсию.



Условно есть 5 типов алгоритмов регистрации:

- Итеративные методы ближайшей точки (Iterative closest point)
- Вероятностные методы
- Методы ключевых точек (Interest points)
- Методы машинного обучения
- Все остальное

С регистрацией облаков точек много проблем и нюансов

- Облака точек могут иметь разное количество точек
- Облака точек могут лишь частично перекрываться
- Облака точек содержат выбросы и координаты зашумлены
- Нахождение регистрации может занимать очень вычислительных ресурсов
- Решение задачи, вообще говоря, неединственно
- Нет однозначного определения близости облаков точек



# Преобразования между облаками точек

---

Любая регистрация определяется аффинным преобразованием

$$T : \mathbb{R}^3 \rightarrow \mathbb{R}^3 : x \mapsto Rx + t,$$

где  $t \in \mathbb{R}^3$ ,  $R \in \text{Mat}_{3,3}(\mathbb{R})$ .

Любая регистрация определяется аффинным преобразованием

$$T : \mathbb{R}^3 \rightarrow \mathbb{R}^3 : x \mapsto Rx + t,$$

где  $t \in \mathbb{R}^3$ ,  $R \in \text{Mat}_{3,3}(\mathbb{R})$ .

Жесткие регистрации описываются изометриями, т.е. сдвигом и вращением:  $R^T R = I$ ,  $\det(R) = 1$

Любая регистрация определяется аффинным преобразованием

$$T : \mathbb{R}^3 \rightarrow \mathbb{R}^3 : x \mapsto Rx + t,$$

где  $t \in \mathbb{R}^3$ ,  $R \in \text{Mat}_{3,3}(\mathbb{R})$ .

Жесткие регистрации описываются изометриями, т.е. сдвигом и вращением:  $R^T R = I$ ,  $\det(R) = 1$

Нежесткие регистрации описываются невырожденными аффинными преобразованиями:  $\det(R) \neq 0$ .

## Итеративные методы ближайшей точки (ICP методы)

---

Пусть нам заданы два облака точек  $P = \{p_1, \dots, p_n\}$  и  $Q = \{q_1, \dots, q_m\}$ . Облако  $P$  будем называть референсным, а облако  $Q$  текущим.

Пусть нам заданы два облака точек  $P = \{p_1, \dots, p_n\}$  и  $Q = \{q_1, \dots, q_m\}$ . Облако  $P$  будем называть референсным, а облако  $Q$  текущим.

Задача регистрации заключается в нахождении аффинного преобразования  $T$  такого, что облако  $T(Q) \stackrel{\text{def}}{=} \{T(q_1), \dots, T(q_n)\}$  будет “мало отличаться” от  $P$ .

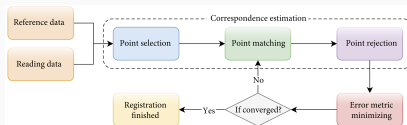
Отличие облаков оценивается некоторой функцией потерь  $L$ , а цель алгоритма минимизировать эту функцию итеративно подбирая преобразование  $T$ .

# Схема ICP алгоритмов

Алгоритм ICP состоит из шагов

- Выбор точек из облаков (point selection)
- Нахождение связей между точками (point matching)
- Отклонение связей (point rejection)
- Минимизация ошибки

Шаги повторяются пока алгоритм не сойдется (например, превышено максимальное количество итераций)



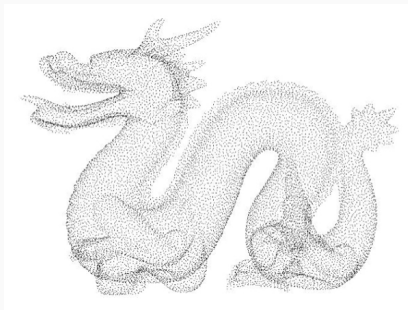


Выбор точек (downsampling) в ICP алгоритме помогает

- Снизить количество точек которые надо обрабатывать
- Выкинуть шумы и выбросы для увеличения точности алгоритма

Выбор точек (downsampling) в ICP алгоритме помогает

- Снизить количество точек которые надо обрабатывать
- Выкинуть шумы и выбросы для увеличения точности алгоритма



Стратегии выбора точек:

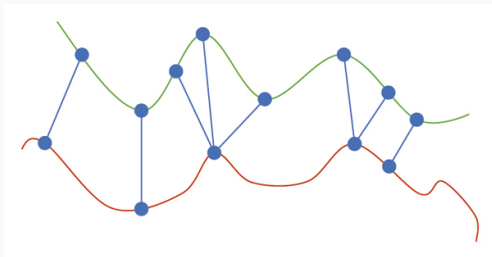
- Случайный выбор (random sampling)
- Каждый раз выбираем новую точку которая находится на некотором расстоянии от предыдущей (distance limit)
- Разобъем пространство на ячейки в каждой ячейке все точки заменим одной - средним всех точек ячейки (uniform sampling)

Нахождение связей необходимо для вычисления функции потерь.

# Схема ICP алгоритмов. Нахождение связок

Нахождение связок необходимо для вычисления функции потерь.

В связку попадают точка  $p$  из референсного облака и точка  $q$  из текущего облака, причем после преобразования  $T$  точка  $T(q)$  — ближайшая к  $p$  в смысле какой-то метрики.



Лобовое решение имело бы сложность  $O(n \cdot m)$ , где  $n$  — количество точек в референсном облаке,  $m$  — в текущем облаке.

Лобовое решение имело бы сложность  $O(n \cdot m)$ , где  $n$  — количество точек в референсном облаке,  $m$  — в текущем облаке.

Конечно, нужно что-то побыстрее. Обычно ускорения добиваются предварительным подсчетом над точками референсного облака. Строится какая-нибудь структура данных которая позволит быстро находить соседнюю точку для заданной. Обычно получаем ускорение до  $O(m \log n)$ . Примеры таких структур данных:

- multi-z-buffer
- kd-tree
- octree

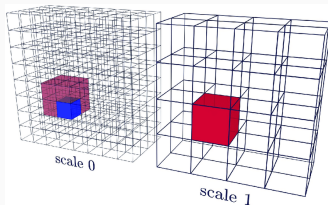
Принцип работы:

- Разрезаем пространство на ячейки (воксели)
- Для каждой точки за  $O(1)$  находим ее ячейку
- В ячейке любым способом ищем соседей



Принцип работы:

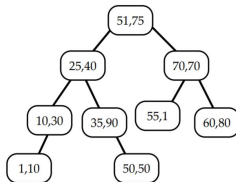
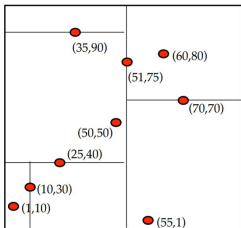
- Разрезаем пространство на ячейки (воксели)
- Для каждой точки за  $O(1)$  находим ее ячейку
- В ячейке любым способом ищем соседей



# Схема ICP алгоритмов. Нахождение связок. Kd-tree

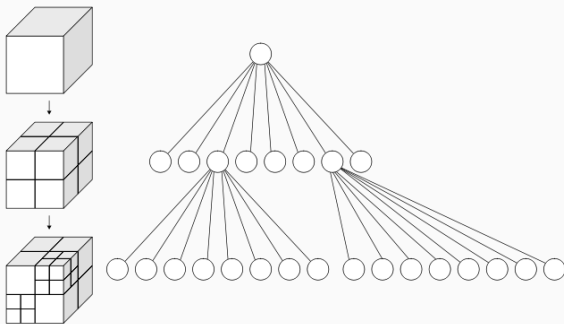
Принцип работы:

- Разрезаем пространство 'пополам' каждый раз переключая нормаль плоскости разрезания ( $Ox$ ,  $Oy$ ,  $Oz$ )
- Этот процесс описывает построение некоторого бинарного дерева
- Поиск ближайшей точки похож на поиск в бинарном дереве

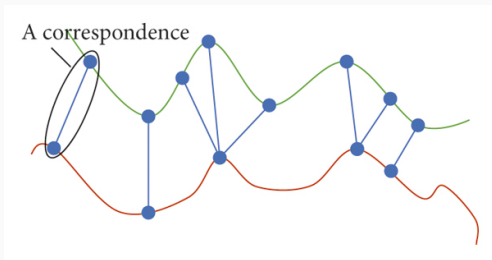


Принцип работы:

- Разрезаем пространство на 8 октантов
- В тех октантах где есть точки повторяем предыдущий шаг
- Поиск ближайшей точки похож на поиск в 8-арном дереве



Одной точке из текущего облака может соответствовать несколько точек из референсного. Нам это будет мешать, от таких связей надо избавляться.



Существует несколько алгоритмов удаления связей:

- Удаление определенного процента самых длинных связей
- Удаление связей с длиной больше заданной
- Удаление связей в которых есть задваивание точек
- Удаление связей с длиной больше медианной
- Удаление связей с несогласованными нормальными
- ...

Существует несколько алгоритмов удаления связей:

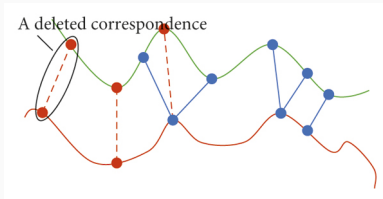
- Удаление определенного процента самых длинных связей
- Удаление связей с длиной больше заданной
- Удаление связей в которых есть задваивание точек
- Удаление связей с длиной больше медианной
- Удаление связей с несогласованными нормальными
- ...

Можно применить сразу несколько алгоритмов

Удаление связок с длиной больше заданной интуитивно понятно, но требует подбора максимального значения. Можно высчитывать медианную длину связок по всему облаку или по окрестности

## Схема ICP алгоритмов. Удаление связок по длине

Удаление связок с длиной больше заданной интуитивно понятно, но требует подбора максимального значения. Можно высчитывать медианную длину связок по всему облаку или по окрестности

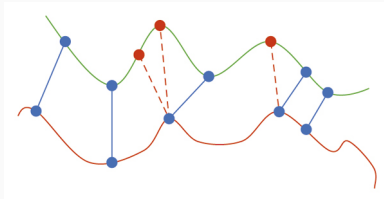




Удаление связок с задваиванием, гарантирует, что у нас каждой точке текущего облака будет соответствовать одна точка из референсного. Не факт, что это будет “удачная” точка.

## Схема ICP алгоритмов. Удаление связок с задваиванием

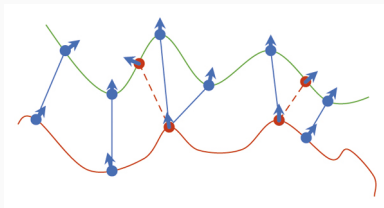
Удаление связок с задваиванием, гарантирует, что у нас каждой точке текущего облака будет соответствовать одна точка из референсного. Не факт, что это будет “удачная” точка.



Удаление связок с по нормалям, убирает точки у которых нормали к поверхности “сильно” разнонаправлены.

## Схема ICP алгоритмов. Удаление связок по нормальям

Удаление связок с по нормальям, убирает точки у которых нормали к поверхности “сильно” разнонаправлены.



Ошибку регистрации облаков можно посчитать когда для каждой точки  $q_i$  из текущего облака  $Q$  найдена связанная точка  $p_i$  из референсного облака  $P$ .

Ошибку регистрации облаков можно посчитать когда для каждой точки  $q_i$  из текущего облака  $Q$  найдена связанная точка  $p_i$  из референсного облака  $P$ .

Обозначим эту ошибку  $L(P, Q, T)$ , тогда искомое преобразование для жесткой регистрации находится из оптимизационной задачи

$$\hat{T} = \operatorname{argmin}_{T \in \operatorname{Isometry}(\mathbb{R}^3)} L(P, Q, T)$$

Типы функций потерь:

- Точка к точке

$$L(P, Q, T) = \sum_{i=1}^m \|p_i - T(q_i)\|^2$$

- Точка к плоскости

$$L(P, Q, T) = \sum_{i=1}^m (n_i(p_i - T(q_i)))^2$$

- Точка к проекции. Находим ближайшую точку в референсном облаке по направлению собственной нормали
- ...

Go-ICP — ищет оптимальное преобразование по всему пространству  $Isometry(\mathbb{R}^3)$  используя метод ветвей и границ

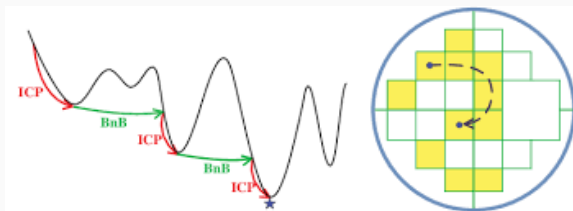
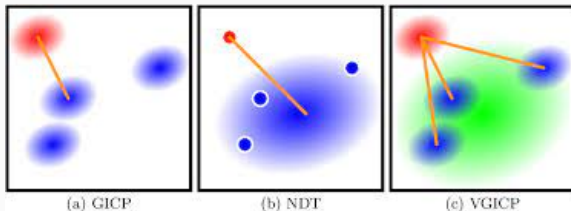


Figure 3. Left: BnB and ICP collaboratively update the upper

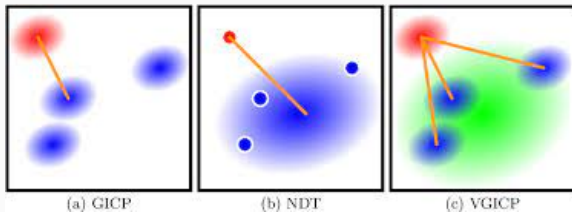


# Известные ICP алгоритмы

GICP (Generalized-ICP) — вводит вероятностную модель на распределение точек в облаке. Варьируя параметры вероятностной модели можно получить ICP алгоритмы с функцией потерь точка к точке или точка к плоскости



VGICP (voxelized GICP) — модификация GICP в которой пространство разбивается на воксели и по точкам в вокселе оцениваются параметры вероятностного распределения GICP. Хорошо адаптируется для видеокарт.



Плюсы:

- Хорошо параллелизуются
- Стабильные и робастные
- Не требуют feature-инжиниринга
- Обобщаются на многомерный случай
- Могут быть использованы в связке с другими алгоритмами

Минусы:

- Требуется начальное приближение
- Требуются предварительные вычисления (например построение kd-дерева)
- Медленные из-за необходимости построения связок
- Во время оптимизации могут застрять в локальном минимуме

# Вероятностные методы

---

Основная идея:

- Все облака являются выборками из некоторого вероятностного распределения
- Обычно вероятностное распределение это смесь Гауссовых распределений (Gaussian mixture model)
- Оптимальное преобразование минимизирует “расстояние” между этими распределениями

Мы обсудим два метода:

- Смесь гауссовских распределений (Gaussian mixture model, GMM)
- Преобразование нормальных распределений (Normal distributions transform, NDT)

Для начала нам потребуется несколько предварительных сведений



Любое облако  $C$  можно описать как смесь гауссовых распределений.

Допустим, что облако точек  $C$  описывается смесью  $N_C$  гауссовских распределений. Пусть

- вес  $i$ -го рапределения –  $\phi_i^C$
- плотность  $i$ -го гауссовского распределения

$$f_i^C(r) = \Gamma(r; \mu_i^C, \Sigma_i^C)$$

Тогда плотность распределения облака

$$f^C(r) = \sum_{i=1}^{N_C} \phi_i^C f_i^C(r)$$

Если к облаку применить преобразование  $T$  задаваемое сдвигом  $t$  и матрицей  $R$ , то плотность распределения изменится следующим образом

$$f^{T(C)}(r) = \sum_{i=1}^{N_c} \phi_i^C f_i^{T(C)}(r)$$

где

$$f_i^{T(C)}(r) = \Gamma(r; \mu_i^C + t, R\Sigma_i^C R^T)$$

Мы хотим найти преобразование  $T$  при котором распределение  $f^P$  референсного облака как можно меньше отличалось от распределения  $f^{T(Q)}$  преобразованного текущего облака  $Q$ .

В методе GMM расстояние между вероятностными распределениями вычисляется как интегральная  $L_2$  норма

$$d(f_1, f_2) = \int_{\mathbb{R}^3} (f_1(r) - f_2(r))^2 dr$$

В итоге мы получаем задачу минимизации

$$\hat{T} = \operatorname{argmin}_{T \in \operatorname{Isometry}(\mathbb{R}^3)} \int_{\mathbb{R}^3} (f^{T(Q)}(r) - f^P(r))^2 dr$$

Поскольку величины

$$\int_{\mathbb{R}^3} f^P(r)^2 dr \quad \text{и} \quad \int_{\mathbb{R}^3} f^{T(Q)}(r)^2 dr$$

не зависят от  $T$  то задача минимизации сводится к следующей

$$\hat{T} = \operatorname{argmin}_{T \in \operatorname{Isometry}(\mathbb{R}^3)} \int_{\mathbb{R}^3} (-2 \cdot f^{T(Q)}(r) \cdot f^P(r)) dr$$

Эта задача может быть решена *EM*-алгоритмом или с помощью *SVM*.

Алгоритм вычислительно тяжелый. Поэтому люди искали альтернативы

Идея метода NDT похожа на GMM, но мы делаем несколько инженерных упрощений:

- Мы разбиваем референсное облако на воксели и считаем, что внутри вокселя точки распределены как смесь нормального и равномерного распределения
- Параметры нормального распределения оцениваются по координатам точек вокселя
- Для нахождения оптимального преобразования мы применяем метод максимального правдоподобия к облаку  $T(Q)$  и используем плотность распределения референсного облака

Обсудим алгоритм подробнее.

Пусть референсное облако разбито на воксели  $\{V_1, \dots, V_N\}$ .

Рассмотрим произвольный воксель  $V$ . Пусть он содержит точки  $x_1, \dots, x_{k_V}$

Будем считать, что внутри вокселя распределение точек является смесью нормального и равномерного.

$$p_V(r) = \phi_1 \Gamma(r; \mu_V, \Sigma_V) + \phi_2 p_2$$

причем

$$\mu_V = \frac{1}{k_V} \sum_{i=1}^{k_V} x_i \quad \Sigma_V = \frac{1}{k_V - 1} \sum_{i=1}^{k_V} (x_i - \mu_V)(x_i - \mu_V)^T$$

Искомое преобразование  $T$  должно максимизировать вероятность того что облако  $T(Q)$  получено из распределения облака  $P$ .

Таким образом мы получаем задачу оптимизации

$$\hat{T} = \operatorname{argmin}_{T \in \operatorname{Isometry}(\mathbb{R}^3)} \prod_{i=1}^m p_{V(T(q_i))}(T(q_i))$$

где  $V(r)$  это воксель в который попадает точка с радиус вектором  $r$



В алгоритме NDT минимизируемую функцию  $p_V$  заменяют на функцию вида

$$\tilde{p}_V(r) = c_1 \exp \left( \frac{-c_2}{2} (r - \tilde{\mu}_V)^T \tilde{\Sigma}_V^{-1} (r - \tilde{\mu}_V) \right)$$

Она ведет себя похожим образом, и более того такая “замена” позволяет записать простые аналитические выражения для градиента и гессиана минимизируемой функции.

Это в свою очередь позволяет быстро решить задачу оптимизации с помощью методов второго порядка.

## Плюсы:

- Хорошо параллелизуются
- Стабильные и робастные
- Не требуют feature-инжиниринга
- Обобщается на многомерный случай
- Могут быть использованы в связке с другими алгоритмами
- Работают быстрее ИСР так как не требуют вычисления связей точек

Минусы:

- Требуется начальное приближение
- Во время оптимизации могут застрять в локальном минимуме

# Методы ключевых точек

---

Часто в облаке не все точки полезны и нужны.

Пример: большая часть точек на ровной стене не несет полезной информации.

Часто в облаке не все точки полезны и нужны.

Пример: большая часть точек на ровной стене не несет полезной информации.

Идея: найти “интересные” точки и оставить в облаке только их.

Часто в облаке не все точки полезны и нужны.

Пример: большая часть точек на ровной стене не несет полезной информации.

Идея: найти “интересные” точки и оставить в облаке только их.

Какие точки “интересные”?

- Углы объектов
- Переход цвета или интенсивности
- Границы объектов

Часто в облаке не все точки полезны и нужны.

Пример: большая часть точек на ровной стене не несет полезной информации.

Идея: найти “интересные” точки и оставить в облаке только их.

Какие точки “интересные”?

- Углы объектов
- Переход цвета или интенсивности
- Границы объектов

Далее “интересные” точки будут называть ключевыми.



Какие найти ключевые точки?

- Посчитать геометрические характеристики ее окрестности
- Переход цвета или интенсивности
- Границы объектов

Какие найти ключевые точки?

- Посчитать геометрические характеристики ее окрестности
- Переход цвета или интенсивности
- Границы объектов

Такие характеристики называются локальными дескрипторами.

Какие найти ключевые точки?

- Посчитать геометрические характеристики ее окрестности
- Переход цвета или интенсивности
- Границы объектов

Такие характеристики называются локальными дескрипторами.

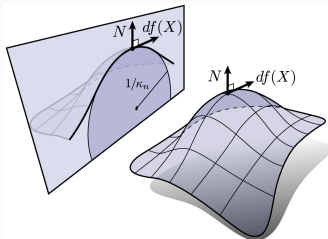
Основное требование к дескрипторам — они должны быть инвариантны относительно преобразований используемых в регистрации. Так как мы чаще всего используем жесткие регистрации, то требуется инвариантность относительно изометрий.

Примеры локальных дескрипторов:

- нормаль к поверхности в точке
- кривизны поверхности в точке
- плотность точек в окрестности

Примеры локальных дескрипторов:

- нормаль к поверхности в точке
- кривизны поверхности в точке
- плотность точек в окрестности



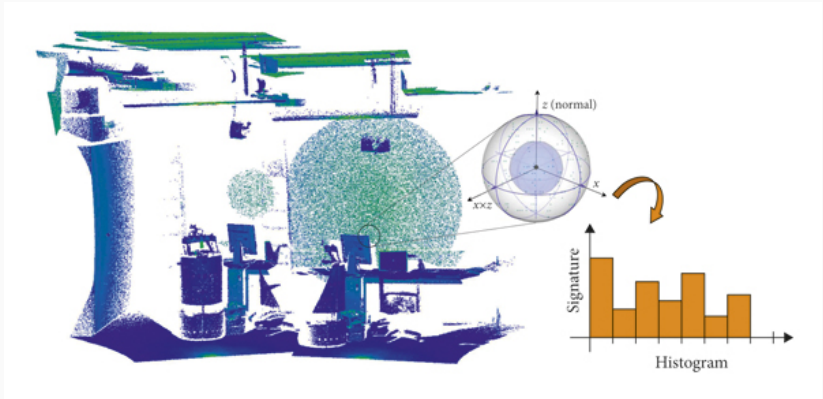
Придумано множество дескрипторов:

- Point Feature Histogram (PFH)
- Signature of Histograms Orientations (SHOT)
- Intrinsic shape signature (ISS)
- Border Aware Repeatable Directions (BOARD)
- ...

Для примера обсудим SHOT дескриптор. Он вычисляется следующим образом:

- Для каждой точки облака подсчитать какую либо характеристику (например угол между нормалью в точке и осью  $z$ )
- Для каждой точки выбрать окрестность и по всем точкам из окрестности построить гистограмму выбранной характеристики
- Данная гистограмма и будет дескриптором точки

# Методы ключевых точек. Продвинутые дескрипторы





- Как только получены дескрипторы всех точек можно начинать процесс регистрации.
- Теперь сопоставлять между собой мы будем не сами координаты точек, а их локальные дескрипторы.
- Сопоставляя дескрипторы мы получаем связки между точками
- Перед сопоставлением дескрипторов можно уменьшить количество точек в референсном и текущем облаке, например с помощью кластеризации.
- Как только получены связки можно вернуться к шагам алгоритма ICP

Плюсы:

- Дают хорошее начальное приближение
- Большой набор готовых дескрипторов
- Можно использовать как начальное приближение для более точных алгоритмов регистрации

Минусы:

- Не дают точного решения
- Медленно работают

В этом обзоре мы не обсудили регистрации точек методами машинного обучения, но это слишком большая тема чтобы уместить ее в этой презентации. Здесь стоит сказать, что лидерами в этой области являются нейронные сети.

Спасибо за внимание!