Подходы к решению задачи нахождения ближайших точек траекторий

Описание задачи

Даны n траекторий в трехмерном пространстве с Евклидовой метрикой. Каждая траектория представляет собой набор из m точек (x, y, z), каждая из которых характеризуется временем t и вектором скорости $\vec{v}$. Конфликтом двух траекторий назовем временной участок, на котором расстояние между траекториями меньше d, где d - заранее определенная константа.

Наивный подход

Идея алгоритма: сканирующая прямая по времени. Будем обрабатывать отрезки траекторий в порядке возрастания времени начала отрезков. В каждый момент времени t будем поддерживать множество текущих отрезков траекторий, заметим, что это множество будет меняться только во время излома одной из траекторий. В момент излома мы должны удалить из нашего множества старый отрезок ломаной и добавить новый. Так же заметим, что при удалении старого отрезка траектории и добавлении нового, еще необработанные конфликты траекторий могут возникнуть только с новым отрезком. Теперь задача свелась к определению конфликта между двумя отрезками. Эта задача не представляет собой сложности: нужно найти участки двух отрезков, на которых расстояние между ними меньше d. Получается, что определение конфликта между двумя отрезками требует $O(1)$ времени, а значит весь процесс добавления нового отрезка траектории с поиском всех конфликтов занимает $O(n)$ времени. Поскольку суммарно все траектории содержат nm отрезков, то вычислительная сложность всего алгоритма $O(n^2m)$. При изменении одной из траекторий время пересчета конфликтов займет $O(nm)$.

Геометрическое хеширование

При наивном подходе при рассмотрении очередного отрезка для поиска конфликтов перебираются все остальные траектории. На практике же как правило все траектории перебирать не имеет смысла, достаточно ограничиться ближайшими. Для поиска ближайших траекторий можно применять геометрическое хеширование, далее рассмотрим один из возможных вариантов, обеспечивающий точное решение задачи. Все трехмерное пространство разобъем на кубические секторы с длиной стороны

$2d$. Определим также временное окно $\Delta_t = \frac{d}{v_{max}}$, где $v_{max} = max(|v|)$. Для каждого момента времени с шагом Δ_t построим граф потенциальных конфликтов. Зафиксируем n точек - положение траекторий в рассматриваемый момент времени, найдем номера кубических секторов, в которых находятся эти точки. Теперь будем строить граф: переберем каждую из n точек и рассмотрим сектор, в котором она лежит и соседние с ним 26 секторов, все точки, обнаруженные в этих секторах соединим в нашем графе ребром с текущей точкой. Технически это можно реализовать следующим образом: вычислим для каждой точки координаты её сектора и положим в структуру *map* пару (Номер сектора, номер точки). Далее для каждой точки перебираем координаты соседних секторов и находим через *map* все точки, лежащие в этих секторах. В зависимости от реализации структуры *map* получаем асимптотики: для сбалансированного дерева $O(n\log(n) + C)$, для хеш-таблицы: $O(n + C)$, где C - количество потенциальных конфликтов.

Теперь покажем корректность алгоритма, а именно: если в какой то момент времени произошел конфликт двух траекторий, то в графе потенциальных конфликтов будет ребро, соединяющее эти две траектории. Предположим противное: в момент времени t произошел конфликт двух траекторий, а в графе нет соответствующего ребра. Рассмотрим моменты времени, предшествующий и последующий после конфликта, в которые мы перестраивали граф: t_1, t_2: $t_1 \leq t \leq t_2$. Поскольку в обоих графах нет ребра между этими траекториями, то значит в моменты времени t_1 и t_2 точки траекторий не лежали в соседних секторах, а значит расстояния между ними были больше размера блока, то есть $> 2d$. Но мы знаем, что в момент конфликта расстояние было меньше d, а это означает, что между моментами времени t_1 и t_2 расстояния между траекториями уменьшилось с $2d$ до d, а затем снова увеличилось до $2d$, получается что суммарно точки за это время преодолели расстояние как минимум $2d$. Откуда получаем, что они суммарно должны были передвигаться со скоростью $> \frac{2d}{\Delta_t} = \frac{2d}{\frac{d}{v_{max}}} = 2v_{max}$, а это означает, что хотя бы одна из них двигалась со скоростью $> v_{max}$. Суммарная вычислительная сложность алгоритма при использовании хеш-таблиц: $O(\frac{T}{\Delta_t} \cdot (n + C))$

Ссылки

http://www.ams.sunysb.edu/ jsbm/papers/cdc97.pdf