Обзор COLMAP

Сергей Кривохатский

СПбГУ, Современное программирование

2 декабря 2022 г.

Что мы изучили?

- 2D-алгоритмы
  - выделение и отслеживание уголков на видео
  - wide-baseline matching
- модель камеры
  - представление позиций в 3D
  - проецирование
- переход из 2D в 3D
  - вычисление движения камеры между двумя кадрами
  - триангуляция точек
  - вычисление позиции камеры по 2D-3D соответствиям
  - bundle adjustment

Что мы реализовали?

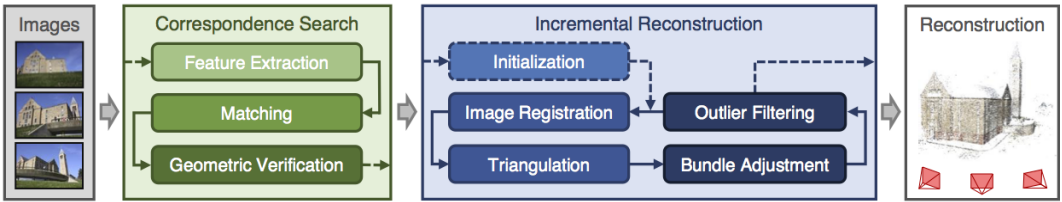
- мы (почти) написали свой оффлайн трекер камеры
- при этом считали, что
  - внутренние параметры камеры известны
  - на вход приходит видео
  - размер видео не очень велик
- реализованный трекер способен получить близкие к правде результаты
- не учитывается ситуация, в которой камера возвращается почти на то же место, где уже была
- важный вывод: *детали реализации и общий алгоритм вычислений очень важны*

Что рассмотрим сегодня?

- разберем современный подход к Structure from Motion
- лекция основана в основном на COLMAP

Разреженная модель центра Рима, построенная по 21000 изображений

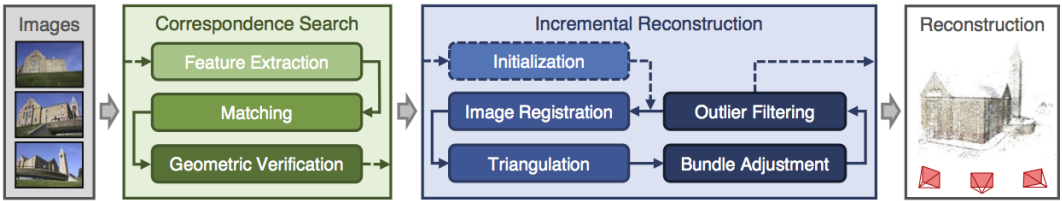
Инкрементальный подход к SfM



Схему инкрементального алгоритма можно разбить на две части

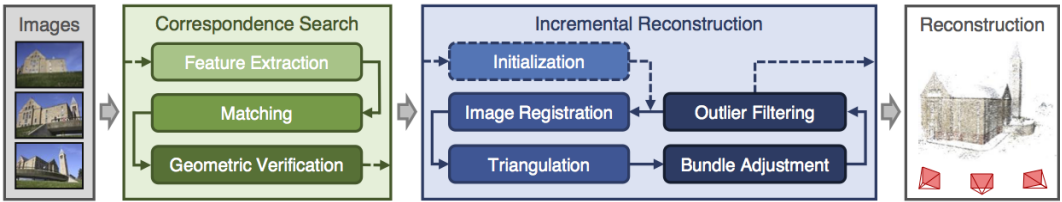
- поиск точечных соответствий
- инкрементальная реконструкция

Инкрементальная реконструкция I



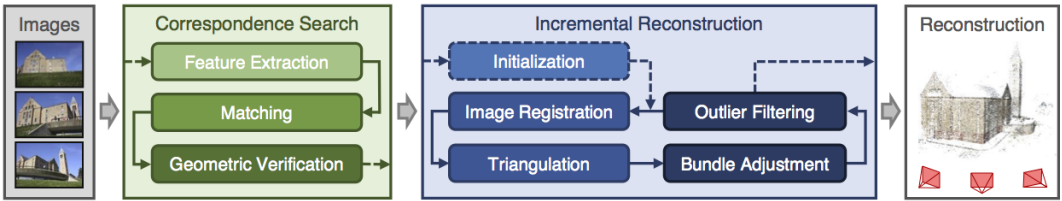
- инициализация — первое восстановление движения камеры и 3D-точек по двум кадрам
- от выбора пары изображений зависит успех всего процесса

Инкрементальная реконструкция II



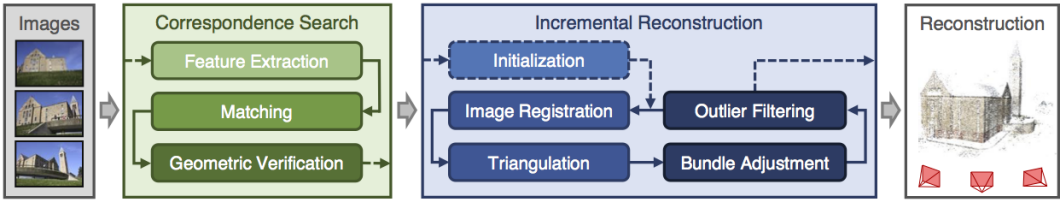
- регистрация изображения — определение позиции камеры по 2D-3D соответствиям
- имеет значение, в каком порядке проводить регистрацию
- ошибки при регистрации повлекут за собой ошибки при триангуляции 3D-точек

Инкрементальная реконструкция III



- триангуляция — вычисление 3D-позиций точек по их 2D-позициям на двух (или нескольких) кадрах
- ошибки при триангуляции повлекут за собой ошибки при регистрации изображений

Инкрементальная реконструкция IV



- как бы мы ни старались, идеальных регистрации и триангуляции добиться нельзя
- ошибки со временем накапливаются, что скорее всего быстро приведет к непотребным результатам
- для борьбы с этим применяется bundle adjustment
- BA для больших сцен — это дорого и сложно

Поиск точечных соответствий

- в первую очередь вычислим ключевые точки и их дескрипторы для каждого изображения
- мы умеем искать 2D-2D соответствия для двух изображений, но что делать с несколькими кадрами?
- для небольшого числа изображений можно за $\mathcal{O}(N^2)$ матчить все пары — это будет лучше всего
- для большого числа нужны более быстрые стратегии

Поиск соответствий между кадрами

- **пространственный:** если в метаданных изображений есть геометки, матчим каждый кадр только с ближайшими
- **словарное дерево:** по всем кадрам строится база, позволяющая искать похожие изображения, затем каждый кадр матчится с K наиболее похожими
- **последовательный:** если на входе видео или серия последовательных фото, можно матчить соседние кадры и детектировать возвращение назад словарным деревом
- **транзитивный:** есть матчи между кадрами A и B, B и C, тогда матчим A и C (перед этим используется другая стратегия)

Граф сцены

На выходе этапа поиска соответствий — *граф сцены*

- вершины соответствуют изображениям
- ребра — парам изображений, для которых есть матчи
- чем плотнее граф, тем точнее (потенциально) решение

Разметка графа сцены

- для каждого ребра считаются значения
  - N_F — число инлаеров фундаментальной матрицы
  - N_H — число инлаеров гомографии
- если $N_H/N_F < \epsilon_{HF}$, то ребро помечается как *произвольное движение*
- иначе из матрицы гомографии вычисляется движение камеры и ребро помечается как *плоская сцена* или *панорамная сцена* — вращение камеры на месте
- разметка полезна при инициализации и триангуляции — чтобы не триангулировать панорамные пары изображений

Инициализация

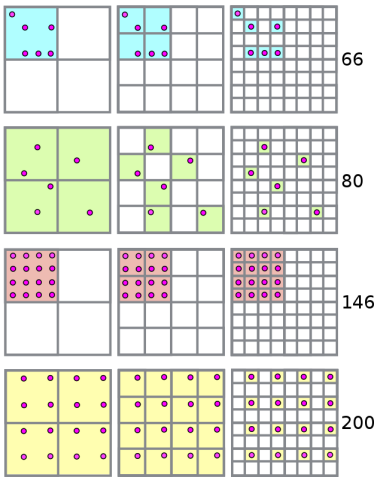
- пару кадров для инициализации — ребро — стараются выбрать в плотной части графа
- инициализация производится по ребру, помеченному как произвольное движение
- важно выбрать пару кадров с хорошим бейзлайном
- если есть пары, для которых известны внутренние параметры камеры, берутся они

Выбор лучшего следующего кадра I

- одно неудачное решение может повлечь каскадное увеличение числа ошибок
- эксперименты показывают, что для точности решения имеют значение
  - количество 2D-3D соответствий
  - равномерность распределения точек по площади кадра
- вводится порог на минимальное число 2D-3D соответствий и рассматриваются только проходящие его кадры
- вводится эвристическая функция оценки кадра и предлагается выбирать кадр с наилучшей оценкой

Выбор лучшего следующего кадра II

- выбирается число уровней L
- на уровне l кадр разбивается на $K_l = 2^l$ частей по каждой из осей
- каждая ячейка, в которую попадает хотя бы одна точка, прибавляет к оценке вес $w_l = K_l$



Триангуляция I

- особенности матчинга точек с помощью дескрипторов
  - чем меньше бейзлайн, тем скорее всего лучше матчи
  - геометрическая верификация не отсеивает ошибочные матчи вдоль эпиполярных линий
- триангуляция по исходным матчам с двух кадров на практике показывает себя как недостаточно хорошая

Триангуляция II

- найденные на парах кадров 2D-2D соответствия объединяются в *треки* — проекции одной и той же 3D-точки на разных изображениях
- треки позволяют триангулировать по нескольким кадрам или по парам кадров с бóльшим бейзлайном
- велика вероятность объединения треков двух особенностей в один из-за неверных матчей вдоль эпиполярных линий
- поэтому используется триангуляция по нескольким кадрам на основе RANSAC

Триангуляция III

Для одного трека

- сэмплируются уникальные пары кадров, пары с панорамным движением исключаются
- для каждой выборки производится триангуляция
- гипотеза валидна, если
  - угол триангуляции достаточно велик
  - 3D-точка располагается перед обеими камерами на достаточно большой глубине
- выбирается гипотеза с наибольшим числом инлаеров

Триангуляция IV

- если среди выбросов остается больше трех точек, RANSAC запускается еще раз на них
- с помощью таких последовательных запусков RANSAC разделяются ошибочно объединенные треки

Bundle Adjustment

- BA позволяет бороться с накопленными ошибками
- COLMAP выполняет два типа BA
  - локальный: производится после каждой регистрации на подмножестве кадров, соседних новому в графе сцены
  - глобальный: производится на всей сцене всякий раз после увеличения ее размера на определенный процент относительно момента предыдущего BA
- для борьбы с выбросами в локальном BA используется M-оценка Коши

Bundle Adjustment. Фильтрация

- после оптимизации 2D-измерения с большими оценками репроекции отфильтровываются
- для каждой 3D-точки проверяется, существует ли пара кадров, с которых точка видна и угол триангуляции достаточно велик

Bundle Adjustment. Ретриангуляция

Ретриангуляция производится на этапе глобального BA

- сначала ретриангуляция производится перед BA
  - порог на ошибку репроекции точек увеличивается
  - делается это для привязки «сползших» кадров к облаку точек
  - помогает при замыкании петель
- после BA также проводится ретриангуляция
  - порог на ошибку репроекции точек не увеличивается
  - добавляются новые 2D-измерения для 3D-точек
  - вычисляются новые 3D-точки
  - производится попытка объединить треки

Bundle Adjustment. Несколько итераций

- после двух ретриангуляций, BA и фильтрации заметное число 2D-измерений может быть отфильтровано
- повторный запуск BA на очищенных данных обычно улучшает результат
- поэтому BA вместе с ретриангуляциями выполняется итеративно до тех пор, пока выбросы не исчезнут
- в большинстве случаев хватает двух-трех итераций

Выделение избыточных кадров

- при большом числе кадров BA — очень дорогой в смысле вычислений этап
- в наборах неупорядоченных фотографий могут встречаться кадры, практически дублирующие друг друга
- в COLMAP используется эвристический алгоритм объединения таких кадров в группы
- при оптимизации отдельная группа параметризуется как одна камера, т. е. кадры двигаются совместно
- в объединении не участвуют кадры, которые были добавлены или модифицированы незадолго до BA
- объединение позволяет заметно сократить время вычислений

Словарное дерево — Vocabulary Tree

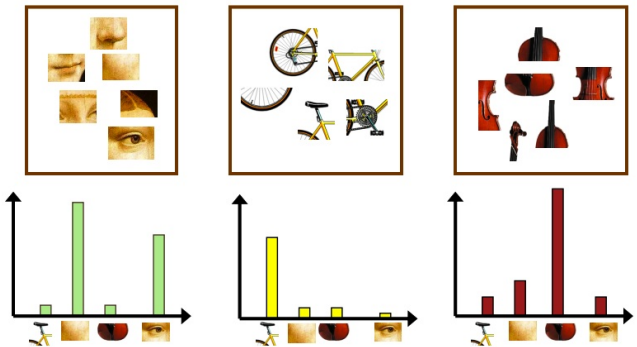
- в чем суть
  - имеется большой набор изображений
  - этот набор предварительно обрабатывается
  - далее на вход подаются запросы — изображения
  - ответ на запрос — похожие изображения из исходного рабора
- существует множество подходов к решению данной задачи
- далее разберем базовый вариант словарного дерева
- словарное дерево — вариант реализации метода *мешок визуальных слов* — Bag of Visual Words (BoVW)

Мешок визуальных слов. Основная идея I

- все точечные особенности изображений исходного набора разбиваются на множество групп по похожести
- каждая такая группа называется визуальным словом

Мешок визуальных слов. Основная идея II

- каждое изображение можно охарактеризовать тем, как часто в нем встречаются те или иные визуальные слова



Мешок визуальных слов. Основная идея III

- каждое изображение можно охарактеризовать тем, как часто в нем встречаются те или иные визуальные слова
- в том числе изображение-запрос
- ответом на запрос будут изображения, наиболее похожие на запрос в смысле схожести наборов визуальных слов

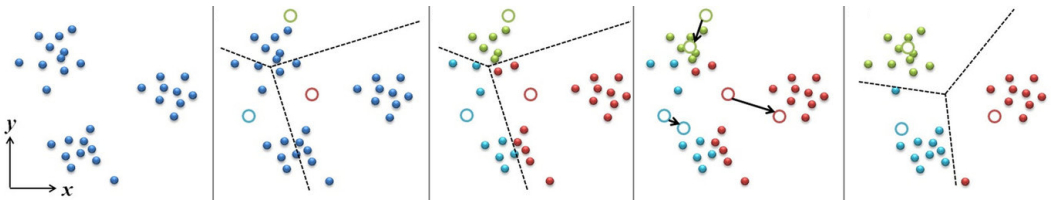
Построение словарного дерева I

- для всех изображений считаются ключевые точки
- для каждой точки считается дескриптор
- получаем набор точек в пространстве дескрипторов
- эти дескрипторы будем разбивать по похожести

Построение словарного дерева II

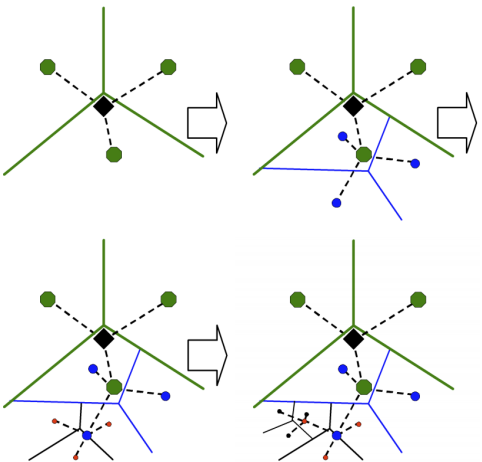
Алгоритм k-means, k — число кластеров

1. выберем k случайных центров для кластеров
2. разобьем точки исходного набора по близости к центрам
3. для каждого кластера посчитаем новый центр — центр масс
4. если что-то поменялось, перейдем к шагу 2



Построение словарного дерева III

- дескрипторы иерархически кластеризуются
- k — число кластеров
- d — число уровней разбиения
- на выходе — k-арное дерево глубины d
- листья — визуальные слова
- разбиение можно построить по большой базе изображений, а затем переиспользовать



Словарное дерево. Описание изображения

- опишем каждое изображение I неким вектором p
- часто используют статистику *TF-IDF*
- элемент вектора p_i соответствует визуальному слову w_i

$$p_i = \frac{\#\text{вхождений } w_i \text{ в } I}{\#\text{дескрипторов в } I} \cdot \log \frac{\#\text{изображений}}{\#\text{изображений со словом } w_i}$$

- обычно вектора-описания изображений нормализуются

$$\hat{p} = p / \|p\|$$

- похожесть двух изображений описывается расстоянием между их векторами-описаниями

Словарное дерево. Обратный индекс

- для каждого слова w_j посчитаем список изображений, в которых оно присутствует
- соответствующий изображению I_i элемент списка пометим элементом вектора-описания данного изображения $\hat{p}_{ij}$, относящимся к слову w_j
- ответ на запрос
  - построим вектор $\hat{q}$ для входного изображения
  - обойдем ненулевые элементы $\hat{q}$ и для изображений из обратного индекса соответствующих слов будем обновлять оценки
  - выберем изображения с лучшими оценками
  - верифицируем эти изображения

Спасибо за внимание!