Basic Сamera Tracker, Part 2

# Задача

С помощью библиотеки OpenCV и выданного преподавателями дополнительного кода добавить в существующий трекер инициализацию позиции камеры по двум кадрам.

# Подробное описание

* Основная цель задания — добавить инициализацию позиции камеры по двум кадрам и настроить вашу реализацию трекинга камеры так, чтобы она работала как можно лучше.
* Всевозможные константы, необходимые для настройки методов трекинга, подобрать самостоятельно.
* Домашнее задание нужно выполнять с использованием выдаваемого нами кода:
  + Реализацию инициализации нужно писать в файле camtrack.py. Нельзя менять интерфейс модуля camtrack.
  + Файл \_camtrack.py содержит вспомогательный код, но реализованный преподавателями. Прежде чем что-то писать самостоятельно, убедитесь, что это еще не сделали за вас.
  + Если очень хочется или нужно, разрешается менять код в файле \_camtrack.py, но так, чтобы не сломать уже существующие интерфейсы.
  + Файл camtrack.py может использоваться как консольное приложение для запуска вашей реализации трекинга и отображения ошибок репроекции.
  + Тестировать результаты трекинга можно с помощью скрипта cmptrack.py.
  + **Прогнать ваше решение сразу на всём тестовом наборе домашнего задания можно (и нужно) с помощью скрипта testrunner.py.**
  + Визуализировать результаты трекинга можно с помощью render.py.

# Требования

* Ваша программа должна успешно запускаться с помощью testrunner.py и корректно завершаться на всех тестах из датасета, расположенного на Google Drive в директории курса. Для запуска используйте конфигурацию тестов dataset\_ha3.yml.
* Для теста bike\_translation\_slow должны выполняться ограничения:
  + максимальная ошибка вращения < 10°,
  + Максимальная ошибка параллельного переноса < 0.1.
* Для как минимум четырёх из остальных шести тестов ваша программа не должна завершать свою работу ранее, чем обработает все кадры, и должны выполняться следующие ограничения:
  + максимальная ошибка по вращению < 20°,
  + максимальная ошибка по параллельному переносу < 0.2.
* Ваш трекер должен выводить в stdout информацию о прогрессе трекинга: какой кадр обрабатывается; число инлаеров, по которым найдена позиция; число триангулированных на текущем этапе точек; размер облака точек в данный момент и т. п.

# Подсказки

* Используйте RANSAC для отсеивания выбросов при вычислении существенной матрицы и матрицы гомографии.
* Определяйте надежность вычисленной существенной матрицы (см. лекции).
* Проверяйте угол триангуляции и ошибки репроекции при вычислении позиций 3D-точек.
* При инициализации трекинга необходимо выбрать достаточно хорошую пару кадров, поэтому устанавливайте строгие пороги при определении успеха инициализации.
* Если не удалось найти пару, для которой выполняются строгие пороги, можете выбрать лучшую из рассмотренных.

## Тестирование

* Набор данных для тестирования трекинга расположен на Google Drive в директории курса.
* Для сравнения результатов трекинга с ground truth используйте cmptrack.py.
* Используйте camtrack.py --show для визуализации ошибок репроекции.

# Дополнительные задания

1. **(до 8 баллов) Bundle Adjustment**Реализовать уточнение результатов трекинга с помощью bundle adjustment. Разрешается пользоваться любыми легко устанавливаемыми (т. е. через pip) библиотеками и использовать численное дифференцирование.
   1. Самый простой вариант с численным дифференцированием — **3 балла**.
   2. Ретриангуляция всех оптимизируемых точек перед запуском оптимизации дает **1 балл**.
   3. Аналитическое дифференцирование дает **4 балла**.
2. **(2 балла) P*n*P c M-оценками**Реализовать робастный метод решения задачи P*n*P путем итеративной минимизации ошибки репроекции с M-оценками.