Реконструкция 3D-моделей реальных объектов или сцен

Синопсис:

Этот код представляет собой полный процесс компьютерного зрения, предназначенный для обнаружения характерных точек из набора изображений в формате BMP, вычисления отношения соответствия ключевых точек, оценки базисной матрицы, вычисления матрицы сущности, 3D-реконструкции и сохранения результата в виде изображения в формате BMP. Она включает в себя следующие этапы:

- **Загрузка изображений и масштабирование серого**
- **Обнаружение критической точки**
- **сопоставление характерных точек**
- **Оценка базовых и основных матриц**
- **Оценка положения камеры**
- **3D-реконструкция**
- **Визуализация облака точек и создание файлов BMP.**

Загрузка изображений и масштабирование серого:

Реализованы функции для чтения и обработки изображений в формате BMP, включая преобразование изображений в серую шкалу, денуазинг (фильтрацию по среднему значению) и простую балансировку света.

Общая схема этого кода выглядит следующим образом:

1. Прочитайте заголовок файла и заголовок сообщения файла BMP, чтобы убедиться, что файл отформатирован правильно.
2. Считывает пиксельные данные и преобразует изображение в серую шкалу.
3. Изображения подвергаются средней фильтрации для уменьшения шума.
4. Повышение контрастности изображения с помощью процесса балансировки света.
5. Наконец, возвращаются обработанные данные изображения в серой шкале.

Обесцвечивание пикселей (фильтрация по среднему значению)

- Для уменьшения шума к изображению применяется простой фильтр средних значений.
- Для каждого пикселя, не являющегося краем, берется среднее значение всех пикселей в его окрестности 3x3.
- Это сглаживает изображение и уменьшает влияние случайного шума.
- Для предотвращения проблем, связанных с выходом за границы, пиксели краев остаются с неизменными исходными значениями.

баланс освещённости

- Чтобы сбалансировать эффект освещения изображения, код нормализует пиксели изображения.
    - Находит минимальное значение пикселя `min_pixel` и максимальное значение пикселя `max_pixel` для всего изображения.
    - Используйте эти значения для нормализации пикселей к диапазону `[0, 255`

который считывает данные о пикселях и преобразует RGB-значение каждого пикселя в значение серой шкалы. grey = int(0.299 * r + 0.587 * g + 0,114 * b)

В результате получается матрица нормализованных_пикселей изображения, в которой каждый элемент представляет собой значение шкалы серого цвета для пикселя (от 0 до 255).

Обнаружение критической точки:

Модуль обнаружения ключевых точек реализуется классом `KeypointDetector`.

- **Вычисление горизонтального и вертикального градиента**: горизонтальный (`Ix`) и вертикальный (`Iy`) градиенты изображения вычисляются с помощью алгоритма Собеля для получения информации о краях изображения.
- **Обнаружение углов Харриса**: значение углового отклика (`R`) каждого пикселя вычисляется методом обнаружения углов Харриса и определяется пороговым значением, чтобы определить, является ли он углом или нет. Для лучшего обнаружения угловых точек к изображению также применяется гауссово сглаживание.
- **Подавление без максимума**: Для подавления без максимума используется окно 3x3, и точка с наибольшим угловым откликом сохраняется, чтобы получить
  Ключевые моменты.
- **Вычисление дескрипторов**: SIFT-подобные дескрипторы вычисляются путем разделения области вокруг каждой ключевой точки на подрегионы 4x4 и подсчета направлений градиента в каждом подрегионе для создания дескрипторов признаков.

Фильтр Гаусса: сглаживает изображение с помощью фильтра Гаусса.

gaussian_kernel(self, size, sigma) Генерирует матрицу гауссова ядра для свертки.

- Гауссово ядро используется для удаления шума и придания изображению большей гладкости.
- Каждый элемент ядра получается на основе двумерной гауссовой функции, которая вычисляется и нормируется по всему ядру так, чтобы сумма была равна 1.

Весь код реализует извлечение признаков из изображения, находит ключевые точки на изображении с помощью определения угловой точки Харриса и генерирует дескрипторы с помощью SIFT-подобного метода. Основные этапы включают вычисление градиента изображения, обнаружение ключевых точек (угловая точка Харриса), гауссово сглаживание и генерацию дескрипторов признаков.

Сопоставление точек характеристик:

Сопоставление признаков выполняется классом FeatureMatcher.
Сопоставление признаков изображения осуществляется с использованием евклидова расстояния для первоначального сопоставления признаков и алгоритма RANSAC для удаления ложных совпадений.

Сходство двух дескрипторов сравнивается с помощью евклидова расстояния, чтобы найти ближайших и вторых ближайших соседей каждой ключевой точки.

欧氏距离的公式是：

$$\text{distance} = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

Используйте zip(desc1, desc2), чтобы пройтись по элементам двух дескрипторов, вычислить сумму квадратов разностей каждой пары элементов, а затем возвести их в квадрат, чтобы получить расстояние.
Используйте тест на соотношение, чтобы отсеять ложные совпадения и избежать неправильно подобранных пар.
Функция match_descriptors_with_ransac() также включена для дальнейшего устранения несоответствий с помощью метода RANSAC.

- **Рабочий процесс RANSAC:**

1. Инициализируйте некоторые переменные: `best_inliers` для хранения лучшего набора внутренних точек и `best_model` для хранения лучшей модели.
2. Выполните `max_`итераций, каждый раз случайным образом выбирая пару совпадений и предполагая, что это правильное совпадение.
3. Модель оценивается на основе выбранных пар совпадений, в данном случае модель может представлять собой преобразование между ключевыми точками.
4. Все совпадающие пары проверяются и подсчитывается количество внутренних точек. Пара, удовлетворяющая расстоянию до модели меньше порогового значения `ransac_threshold`, то она считается внутренней точкой.
5. Обновление лучшей модели: если количество внутренних точек в текущей модели превышает предыдущее лучшее количество внутренних точек, обновите лучшую модель и набор внутренних точек.

- После фильтрации RANSAC возвращается конечный набор совпадений, т.е. внутреннее множество точек.

Оценка модели:

estimate_model(self, kp1, kp2):

- Метод используется для оценки параметров модели.
- В этом коде возвращаются только позиции двух ключевых точек `(kp1, kp2)`, что указывает на то, что модель представляет собой простое отношение, напрямую связывающее эти две ключевые точки.

Судите по внутренним точкам:

is_inlier(self, kp1, kp2, model, threshold):

- Определяет, является ли точка совпадения внутренней точкой модели.
- `kp1` и `kp2` - две ключевые точки совпадения, а `модель` - модель, оцененная по случайной выборке совпадающих пар.
- Используя евклидово расстояние в качестве критерия оценки, совпадение считается внутренней точкой, если расстояние между двумя ключевыми точками меньше заданного порогового `значения`.
- Судить о внутренней точке нужно прежде всего для того, чтобы определить, подходит ли это совпадение к текущей расчетной модели.

Оценка базовых и основных матриц:

Оценкой фундаментальной и существенной матриц занимается класс

`FundamentalAndEssentialMatrixEstimator`.

- **Оценка базовой матрицы**: базовая матрица (`F`) - это матрица, которая описывает геометрические отношения между двумя изображениями и оценивается путем случайного выбора совпадающих пар точек с использованием восьмиточечного метода и RANSAC для повышения надежности.
- **Вычисление внутренней матрицы**: вычислите внутреннюю матрицу (`E`) из матрицы базиса и внутренней матрицы отсчета камеры (`K`), которая описывает относительную позу между двумя изображениями, когда внутренняя привязка камеры известна.

Расчетная базовая матрица

`estimate_fundamental_matrix(self)`:

-

- Оценка базисной матрицы путем сопоставления пар точек.
- Итерации выполнялись с помощью RANSAC, при этом для оценки случайным образом выбиралось 8 точек за раз.
- Для каждой оцененной базисной матрицы подсчитайте соответствующее количество внутренних точек и сохраните базисную матрицу с наибольшим количеством внутренних точек.

- Если найдена наилучшая модель и количество внутренних точек достаточно велико, все внутренние точки используются для оптимизации базисной матрицы.

`estimate_fundamental_matrix_from_points(self, pts1, pts2)`:

-

- Оцените базисную матрицу, используя заданные соответствия.
- точки нормализации для повышения устойчивости вычислений, построить матрицу `A` и решить базисную матрицу методом SVD.
- Заставляет базисную матрицу иметь ранг 2, т.е. последнее сингулярное значение устанавливается равным нулю.

Оценка матрицы существенности

`estimate_essential_matrix(self, F)`:

- Вычислите матрицу существенности `E`, используя внутреннюю опорную матрицу камеры `K` и базисную матрицу `F`.
- SVD-разложение существенной матрицы с ограничениями на сингулярные значения (установите сингулярные значения на `[1,1,0]`, чтобы обеспечить выполнение свойств существенной матрицы.

Оценка положения камеры:

Из основной матрицы извлекается положение камеры (`R`, `t`), то есть матрица поворота камеры и вектор перевода.

- **Извлечение возможных поз камеры**: из матрицы сущностей извлекаются четыре возможные позы камеры.
- **Выбор правильной позы камеры**: выбор правильной позы камеры путем проверки результатов реконструкции 3D-точек

Трехмерная реконструкция:

3D-реконструкция реализуется классом `Reconstruction3D`, который вычисляет положение каждого совпадения в 3D-пространстве путем триангуляции пар известных поз камеры и точек совпадения.

- **Триангулированные точки**: позиции 3D-точек рассчитываются с помощью матрицы камеры путем проецирования точек соответствия на плоскость камеры двух изображений.
- **Генерация облака 3D-точек**: конечный результат - набор 3D-точек, представляющих форму объекта в пространстве.

Визуализация облака точек и создание файлов BMP:

Для визуализации результаты 3D-реконструкции можно сохранить в виде облака точек в формате BMP.

- **Визуализация 3D-облака точек**: функция `visualize_point_cloud()` используется для простой печати протяженности облака точек и информации о координатах некоторых из них.
- **Создание BMP-файлов**: функция `create_bmp_from_points()` проецирует 3D-облако точек на 2D-плоскость и генерирует BMP-изображение, что позволяет более интуитивно наблюдать результаты 3D-реконструкции.

Недостатки и ограничения:

1. Недостаточная надежность извлечения и сопоставления признаков:

- В коде используется детектор угловых точек Харриса, и этот метод не так устойчив к изменениям освещения, углу обзора и окклюзии, как более современные алгоритмы обнаружения признаков (SIFT, ORB). Метод вычисления дескрипторов прост и не обеспечивает достаточной дискриминационной способности для обеспечения точности сопоставления.
- На этапе сопоставления признаков использовался метод насильственного сопоставления и фильтрации совпадений с помощью евклидова расстояния и тестов на соотношение. Скорость сопоставления низкая.

2. Недостаточная предварительная обработка изображений:

- Для обесцвечивания изображений используется простой фильтр средних значений, который не может эффективно справиться с шумом в сложных сценах. Более продвинутые методы обесцвечивания (например, медианный фильтр или билатеральный фильтр) не использовались для сохранения деталей краев изображения.
- Обработка светового баланса просто нормализует все изображение, не принимая во внимание такие моменты, как локальные изменения освещенности, что может привести к потере деталей, особенно в областях со значительным контрастом между светлым и темным.

3. Низкая точность 3D-реконструкции:

- Проекционные матрицы P1 и P2, используемые в 3D-реконструкции, представляют собой простые матрицы, определяемые вручную, которые не имеют точной калибровки и настройки реальных параметров камеры и не могут точно отражать реальную систему координат камеры.
- Метод проецирования, используемый в коде, представляет собой простую ортогональную проекцию, в которой отсутствует расчет перспективной проекции, что приводит к низкой точности результатов 3D-реконструкции. Метод триангуляции относительно прост и не учитывает средства оптимизации, такие как ошибка репроецирования, что влияет на точность 3D-облака точек.