Создание трехмерной модели реального объекта или сцены.

Описание:

Этот код представляет собой полный процесс компьютерного зрения, направленный на выполнение следующих задач: обнаружение характерных точек на наборе изображений в формате BMP, вычисление соответствий ключевых точек, оценка фундаментальной матрицы, вычисление матрицы сущности, 3D-реконструкция и сохранение результата в формате BMP. Процесс включает следующие этапы:

- Загрузка изображений и преобразование в оттенки серого
- Обнаружение ключевых точек
- Сопоставление характерных точек
- Оценка фундаментальной матрицы и матрицы сущности
- Оценка положения камеры
- 3D-реконструкция
- Визуализация облака точек и генерация файлов BMP.

Загрузка изображений и преобразование в оттенки серого:

Реализована функция чтения и обработки BMP-изображений, включая преобразование их в градации серого, устранение шума (среднее сглаживание), а также простую обработку для балансировки освещения.
Общий процесс кода выглядит следующим образом:

1. Чтение заголовка файла BMP и информационного заголовка для проверки правильности формата файла.
2. Чтение данных пикселей и преобразование изображения в градации серого.
3. Применение среднего сглаживания для уменьшения шума.
4. Увеличение контрастности изображения с помощью обработки балансировки освещения.
5. Возврат обработанных данных изображения в градациях серого.

Удаление шума с помощью среднего сглаживания:

- Для уменьшения шума применяется простое среднее сглаживание изображения.
- Для каждого пикселя, не находящегося на краю, вычисляется среднее значение всех пикселей в его окрестности размером 3x3.
- Это позволяет сгладить изображение и снизить влияние случайного шума.

- Граничные пиксели остаются неизменными, чтобы избежать проблем с выходом за пределы изображения.

Балансировка освещения

- Для балансировки освещения изображения код выполняет нормализацию пикселей.
    - Определяются минимальное значение пикселя (min_pixel) и максимальное значение пикселя (max_pixel) на всем изображении.
    - С использованием этих значений пиксели нормализуются в диапазон [0, 255].

При этом выполняется чтение данных пикселей и преобразование значений RGB каждого пикселя в градации серого:
gray = int(0.299 * r + 0.587 * g + 0.114 * b)

В результате возвращается матрица пикселей изображения normalized_pixels, обработанного по следующим этапам: градация серого, устранение шума и балансировка освещения. Каждый элемент матрицы представляет собой значение градации серого для одного пикселя (в диапазоне от 0 до 255).

Обнаружение ключевых точек:

Модуль обнаружения ключевых точек реализован с помощью класса **KeypointDetector**.

Вычисление горизонтальных и вертикальных градиентов:
Используется оператор Собеля для вычисления горизонтального (Ix) и вертикального (Iy) градиентов изображения, чтобы получить информацию о краях.

Обнаружение углов Harris:
Метод обнаружения углов Harris вычисляет значение отклика углов (R) для каждого пикселя. На основе порогового значения определяется, является ли пиксель углом. Для улучшения обнаружения углов применяется гауссовое сглаживание.

Подавление немаксимумов:
Используется окно размером 3x3 для подавления немаксимумов. Сохраняются только точки с максимальным значением отклика, что позволяет получить ключевые точки.

Вычисление дескрипторов:
Вычисляются дескрипторы, подобные SIFT. Область вокруг каждой ключевой

точки делится на 4x4 подрегиона. Для каждого подрегиона проводится статистика направлений градиентов, чтобы создать дескрипторы.

Гауссов фильтр:
Гауссов фильтр используется для сглаживания изображения.**Функция gaussian_kernel(self, size, sigma)** генерирует гауссово ядро, которое применяется для свёртки:

Гауссово ядро устраняет шум и делает изображение более гладким.

Каждый элемент ядра рассчитывается на основе двухмерной гауссовой функции. Ядро нормализуется так, чтобы сумма его элементов равнялась 1.

Код выполняет извлечение признаков изображения, обнаруживает ключевые точки с помощью метода углов Harris и создаёт дескрипторы с использованием метода, аналогичного SIFT. Основные этапы включают:

1. Вычисление градиентов изображения.
2. Обнаружение ключевых точек (углы Harris).
3. Гауссовое сглаживание.
4. Генерация дескрипторов.

Сопоставление ключевых точек:

Сопоставление признаков выполняется с помощью класса **FeatureMatcher**.

Реализована функция сопоставления признаков изображений, включающая начальное сопоставление по евклидовому расстоянию и устранение ошибок сопоставления с использованием алгоритма RANSAC.

- **Евклидовое расстояние** используется для сравнения схожести двух дескрипторов. Для каждой ключевой точки находятся ближайший сосед и второй ближайший сосед.

  Формула Евклидова расстояния:

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

Использование **zip(desc1, desc2)** для итерации по элементам двух дескрипторов, вычисления суммы квадратов разностей для каждой пары элементов и последующего извлечения квадратного корня для получения расстояния.

Применяется **тест на соотношение (ratio test)** для фильтрации ложных соответствий и предотвращения ошибок сопоставления.

Также реализована функция **match_descriptors_with_ransac()**, которая с помощью метода RANSAC устраняет ложные соответствия.

Рабочий процесс RANSAC:

Инициализация переменных:

best_inliers используется для хранения набора лучших внутренних точек.

best_model хранит лучший модельный параметр.

Итерация:

Выполняется **max_iterations** итераций. На каждой итерации случайно выбирается одна пара соответствий, которая принимается за корректное соответствие.

Оценка модели:

На основе выбранной пары соответствий оценивается модель. Здесь модель может представлять преобразование между ключевыми точками.

Тестирование соответствий:

Проверяются все пары соответствий и подсчитывается количество внутренних точек. Если соответствие удовлетворяет условию, что расстояние до модели меньше заданного порога **ransac_threshold**, то оно считается внутренней точкой.

Обновление лучшей модели:

Если количество внутренних точек текущей модели превышает количество внутренних точек предыдущей лучшей модели, обновляется текущая лучшая модель и набор внутренних точек.

Итог:

Возвращаются соответствия, прошедшие фильтрацию RANSAC, то есть набор внутренних точек.

Оценка модели:

estimate_model(self, kp1, kp2):

- Метод используется для оценки параметров модели.
- В данном коде он просто возвращает позиции двух ключевых точек **(kp1, kp2)**, что представляет собой простую модель прямого соединения этих ключевых точек.

Определение внутренних точек:

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

- Определяет, является ли соответствие внутренней точкой модели.
- **kp1** и **kp2** – это два ключевых соответствующих пункта, **model** – модель, оцененная на основе случайно выбранного соответствия.
- Используется **евклидово расстояние** в качестве критерия. Если расстояние между двумя ключевыми точками меньше заданного порога **threshold**, то соответствие считается внутренней точкой.
- Определение внутренних точек позволяет определить, насколько соответствие соответствует текущей оценённой модели.

Оценка фундаментальной и сущностной матриц:
Оценка фундаментальной и сущностной матриц выполняется с помощью класса **FundamentalAndEssentialMatrixEstimator**.

Оценка фундаментальной матрицы:

Фундаментальная матрица (F) описывает геометрическую связь между двумя изображениями. Она оценивается с использованием случайного выбора пар соответствий точек, метода восьми точек и алгоритма RANSAC для повышения устойчивости.

Вычисление сущностной матрицы:

На основе фундаментальной матрицы и матрицы внутренних параметров камеры (K) вычисляется сущностная матрица (E). Сущностная матрица описывает относительное положение двух изображений с учётом известных внутренних параметров камеры.

Оценка фундаментальной матрицы:

estimate_fundamental_matrix(self):

- Оценивает фундаментальную матрицу на основе пар соответствующих точек.
- Алгоритм:
    1. Используется RANSAC для итеративного выбора случайных 8 пар точек.
    2. Для каждой оценённой фундаментальной матрицы вычисляется количество внутренних точек.
    3. Сохраняется фундаментальная матрица с наибольшим числом внутренних точек.
    4. Если лучшая модель найдена и число внутренних точек достаточно велико, производится оптимизация фундаментальной матрицы с использованием всех внутренних точек.

estimate_fundamental_matrix_from_points(self, pts1, pts2):

- Вычисляет фундаментальную матрицу, используя заданные пары точек:

    1. Нормализует точки для повышения стабильности вычислений.
    2. Строит матрицу A и решает её с помощью SVD.
    3. Принуждает фундаментальную матрицу иметь ранг 2, устанавливая последний сингулярный компонент равным нулю.

Оценка сущностной матрицы:

estimate_essential_matrix(self, F):

- Вычисляет сущностную матрицу (E) на основе матрицы внутренних параметров камеры (K) и фундаментальной матрицы (F).
- Алгоритм:

    1. Выполняется SVD-разложение сущностной матрицы.
    2. Ограничиваются сингулярные значения, чтобы они соответствовали свойствам сущностной матрицы. Значения устанавливаются как [1, 1, 0].

Оценка положения камеры:
Из сущностной матрицы извлекаются положение камеры (R, t), то есть матрица вращения и вектор трансляции.

- **Извлечение возможных положений камеры**: Из сущностной матрицы извлекаются четыре возможных варианта положения камеры.
- **Выбор корректного положения камеры**: Путём проверки результатов трёхмерной реконструкции выбирается правильное положение камеры.

Трёхмерная реконструкция:
Класс **Reconstruction3D** реализует трёхмерную реконструкцию на основе известного положения камеры и соответствующих пар точек.

- **Триангуляция точек**: Каждая пара соответствующих точек проецируется на плоскость камеры двух изображений, а затем с использованием матриц камеры вычисляется положение трёхмерных точек.
- **Создание трёхмерного облака точек**: В результате создаётся набор трёхмерных точек, которые представляют форму объекта в пространстве.

Визуализация облака точек и генерация файла BMP:
Для удобства визуализации результаты трёхмерной реконструкции сохраняются в виде облака точек, которое можно представить как изображение в формате BMP.

- **Визуализация облака точек**: Функция **visualize_point_cloud()** используется для вывода диапазона облака точек и координат некоторых точек.
- **Создание файла BMP**: Функция **create_bmp_from_points()** проецирует облако точек на двухмерную плоскость и создаёт изображение в формате BMP, что позволяет более наглядно оценить результаты трёхмерной реконструкции.

Недостатки и ограничения:

1.Недостаточная устойчивость к извлечению и сопоставлению признаков:

- В коде используется детектор углов Harris, который менее устойчив к изменениям освещения, ракурса и частичной окклюзии по сравнению с более современными алгоритмами (такими как SIFT или ORB). Простой метод вычисления дескрипторов не обеспечивает достаточной дискриминативной способности для точного сопоставления.
- На этапе сопоставления признаков используется метод полного перебора с фильтрацией по евклидовому расстоянию и тесту на соотношение. Такой подход является медленным и менее эффективным.

2.Недостаточная предобработка изображений:

- Для удаления шума используется простой метод среднего сглаживания, который неэффективен в сложных условиях. Более современные методы, такие как медианный фильтр или двусторонний фильтр, не применяются, из-за чего не сохраняются детали краёв изображения.

- Балансировка освещения выполняется простой нормализацией на уровне всего изображения, без учёта локальных изменений освещения. Это может привести к потере деталей, особенно в областях с сильным контрастом между светлыми и тёмными участками.

3.Низкая точность трёхмерной реконструкции:

- Для трёхмерной реконструкции используются простые проекционные матрицы P1 и P2, которые задаются вручную и не учитывают точную калибровку и параметры реальной камеры. Это снижает точность отображения реальной системы координат камеры.
- В коде используется метод ортогональной проекции, без расчёта перспективной проекции, что ухудшает точность результатов трёхмерной реконструкции. Также используемая триангуляция выполняется простым методом, без учёта ошибок ретропроекции и их оптимизации, что снижает точность полученного облака точек.