Домашнее задание №1: Основы компьютерного зрения — Сопоставление особенностей (Feature Matching)

Цель задания:
Закрепить навыки работы с детекторами и дескрипторами особенностей, научиться находить соответствия между изображениями и оценивать их качество.

Задача 0: Поиск или создание изображений для анализа

Используйте камеру смартфона с высоким разрешением сделайте фотографию контрастного предмета с двух разных ракурсов при одинаковом разрешении. Или найдите пару изображений в интернете, подходящих под указанные требования.

Задача 1: Базовое сопоставление особенностей

1. **Загрузите два изображения**:
   - Выберите два изображения одного объекта/сцены, снятых под разными углами или с небольшим смещением (например, фото книги на столе и её же с поворотом).
   - Преобразуйте их в градации серого.
2. **Обнаружение и описание особенностей**:
   - Используйте детектор **ORB** (через `cv.ORB_create()`) для нахождения ключевых точек и вычисления их дескрипторов на обоих изображениях.
3. **Сопоставление особенностей**:
   - Примените **Brute-Force Matcher** (`cv.BFMatcher()`) с метрикой расстояния Хэмминга (`normType=cv.NORM_HAMMING`).
   - Найдите все возможные совпадения между дескрипторами двух изображений.
4. **Фильтрация совпадений**:
   - Отфильтруйте "хорошие" совпадения с помощью **Lowe's ratio test** (соотношение расстояний между ближайшими соседями: `0.75`).
5. **Визуализация**:
   - Нарисуйте первые 20 лучших совпадений на исходных изображениях с помощью `cv.drawMatches()`.
   - Сохраните результат в файл `matches.jpg`.

Задача 2: Предобработка изображений фильтрами

1. **Применение фильтров**:
   - Реализуйте функции для применения к изображениям:
     - Гауссова размытия (`cv.GaussianBlur`) с ядром 5x5
     - Медианного фильтра (`cv.medianBlur`) с ядром 5
     - Билатерального фильтра (`cv.bilateralFilter`) с параметрами `d=9, sigmaColor=75, sigmaSpace=75`
   - Создайте 3 версии каждого исходного изображения (всего 6 обработанных изображений).
2. **Сравнение результатов**:
   - Для каждого фильтра выполните детекцию особенностей (ORB) и сопоставление (как в Задаче 1).
   - Сохраните визуализацию совпадений для каждого случая (например, `matches_gaussian.jpg`, `matches_median.jpg`).
   - Заполните таблицу:

Фильтр	Количество совпадений до Lowe's test	Количество совпадений после Lowe's test
Исходное	...	...
Гауссов	...	...
Медианный	...	...
Билатеральный	...	...

Задача 3: Анализ результатов

1. Объясните, почему некоторые ключевые точки оказались ложными совпадениями.
2. Как влияет параметр nfeatures в ORB на количество и качество сопоставлений?
3. Что делает Lowe's ratio test и почему он улучшает результат?
4. Как фильтрация повлияла на:
   - Количество ключевых точек?
   - Уровень шума в совпадениях?

    - Общее качество сопоставления?
5. Какой фильтр показал себя лучше всего для ваших тестовых изображений? Почему?
6. В каких сценариях оправдано применение фильтров перед сопоставлением особенностей?

Требования к коду

- Используйте OpenCV (версия 4.x) и Python 3.8+.
- Код должен быть модульным: отдельные функции для загрузки изображений, детекции особенностей, сопоставления и визуализации.
- Добавьте комментарии к ключевым этапам.

Пример кода (подсказка)

```
import cv2 as cv
import matplotlib.pyplot as plt

# Загрузка изображений
img1 = cv.imread("image1.jpg", cv.IMREAD_GRAYSCALE)
img2 = cv.imread("image2.jpg", cv.IMREAD_GRAYSCALE)

# Инициализация ORB
orb = cv.ORB_create(nfeatures=500)
kp1, des1 = orb.detectAndCompute(img1, None)
kp2, des2 = orb.detectAndCompute(img2, None)

# Сопоставление
bf = cv.BFMatcher(cv.NORM_HAMMING, crossCheck=False)
matches = bf.knnMatch(des1, des2, k=2)

# Фильтрация (Lowe's ratio test)
good_matches = []
for m, n in matches:
    if m.distance < 0.75 * n.distance:
        good_matches.append(m)

# Визуализация
result = cv.drawMatches(img1, kp1, img2, kp2, good_matches[:20], None, flags=2)
plt.imshow(result), plt.show()
```

```
def apply_filters(image):
    gaussian = cv.GaussianBlur(image, (5,5), 0)
    median = cv.medianBlur(image, 5)
    bilateral = cv.bilateralFilter(image, 9, 75, 75)
    return gaussian, median, bilateral
```

Дополнительное задание (опционально)

- Реализуйте сопоставление с помощью **FLANN-метода** и сравните его скорость и точность с Brute-Force.
- Примените **гомографию** (`cv.findHomography()`) для совмещения изображений на основе отфильтрованных совпадений.

Критерии оценки

- **5 баллов**: Выполнены все пункты базового задания, код работает без ошибок.
- **+1 балл**: Ответы на вопросы анализа содержательные и точные.
- **+2 балла**: Выполнено дополнительное задание.
- **-1 балл**: Отсутствуют комментарии или нарушен стиль кода.

Срок сдачи: 7 дней. Формат: архив с кодом, изображениями и коротким отчетом в PDF (скриншоты результатов + ответы на вопросы).

Удачи! Если возникнут сложности — пишите в чат курса.

Конечно! Добавление задачи с фильтрами отлично дополнит задание, особенно если связать её с улучшением качества сопоставления. Вот модифицированная версия ДЗ: