Практическая работа № 5

Поиск ключевых точек и сопоставление изображений

Цель работы

- Освоить методы поиска ключевых точек на изображениях.
- Изучить построение дескрипторов (SIFT, ORB).
- Научиться сопоставлять изображения с помощью алгоритмов BFMatcher и FLANN.
- Получить практический опыт в задачах компьютерного зрения: сравнение изображений, поиск объекта на сцене.

Теоретическая часть

Ключевые точки (features)

Ключевые точки — это такие элементы изображения (углы, контуры, текстурные детали), которые можно надежно находить даже при изменении масштаба, поворота, освещенности. Они служат своеобразными "ориентирами" для сравнения изображений.

После нахождения ключевых точек для каждой из них вычисляется **дескриптор** — числовой вектор, описывающий локальное окружение точки.

Алгоритмы обнаружения и описания

- **SIFT (Scale-Invariant Feature Transform)**
  - Устойчив к изменению масштаба и поворота.
  - Хорошо работает для сложных изображений.
  - Недостаток: относительно медленный.
- **ORB (Oriented FAST and Rotated BRIEF)**
  - Быстрый алгоритм, пригоден для задач в реальном времени.
  - Использует бинарные дескрипторы.
  - Может уступать по качеству при сильных искажениях.

Алгоритмы сопоставления

- **BFMatcher (Brute Force)** — перебирает все возможные пары дескрипторов и ищет ближайшие.
- **FLANN (Fast Library for Approximate Nearest Neighbors)** — быстрый метод для больших наборов данных.

Фильтрация по методу Лоу

Для исключения случайных совпадений применяется фильтр:
Если ближайший сосед значительно ближе второго, совпадение считается корректным.
Обычно используют пороговое значение 0.7–0.8.

Практическая часть

Шаг 1. Импорт библиотек

```
import cv2

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline
```

Шаг 2. Загрузка изображений

```
img1 = cv2.imread(cv2.samples.findFile('box.png'), cv2.IMREAD_GRAYSCALE)

img2 = cv2.imread(cv2.samples.findFile('box_in_scene.png'),
cv2.IMREAD_GRAYSCALE)


plt.figure(figsize=(10,4))

plt.subplot(1,2,1), plt.imshow(img1, cmap='gray'), plt.title("Изображение 1")

plt.subplot(1,2,2), plt.imshow(img2, cmap='gray'), plt.title("Изображение 2")

plt.show()
```

Шаг 3. Поиск ключевых точек (SIFT)

```
sift = cv2.SIFT_create()
kp1, des1 = sift.detectAndCompute(img1, None)
kp2, des2 = sift.detectAndCompute(img2, None)

img_sift = cv2.drawKeypoints(img1, kp1, None,
flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
plt.imshow(img_sift), plt.title("Ключевые точки (SIFT)")
plt.show()
```

Шаг 4. Сопоставление (BFMatcher + SIFT)

```
bf = cv2.BFMatcher()
matches = bf.knnMatch(des1, des2, k=2)

good_matches = []
for m, n in matches:
    if m.distance < 0.75 * n.distance:
        good_matches.append(m)

img_matches = cv2.drawMatches(img1, kp1, img2, kp2, good_matches[:20],
None, flags=2)
plt.figure(figsize=(12,6))
plt.imshow(img_matches), plt.title("Совпадения (SIFT + BFMatcher)")
plt.show()
```

Шаг 5. Сопоставление (ORB + FLANN)

```
orb = cv2.ORB_create(nfeatures=1000)
kp1_orb, des1_orb = orb.detectAndCompute(img1, None)
kp2_orb, des2_orb = orb.detectAndCompute(img2, None)
```

```
index_params = dict(algorithm=6, table_number=6, key_size=12,
multi_probe_level=1)

search_params = dict(checks=50)

flann = cv2.FlannBasedMatcher(index_params, search_params)


matches_orb = flann.knnMatch(des1_orb, des2_orb, k=2)


good_orb = []

for m,n in matches_orb:

    if m.distance < 0.75*n.distance:

        good_orb.append(m)


img_orb = cv2.drawMatches(img1, kp1_orb, img2, kp2_orb, good_orb[:20], None,
flags=2)

plt.figure(figsize=(12,6))

plt.imshow(img_orb), plt.title("Совпадения (ORB + FLANN)")

plt.show()
```

Контрольные вопросы

1. Что такое ключевые точки и зачем они нужны?
2. Чем отличаются SIFT и ORB?
3. В чем разница между BFMatcher и FLANN?
4. Для чего используется фильтрация по методу Лоу?
5. В каких практических задачах применяется сопоставление изображений?