Анализ алгоритмов выявления и сопоставления особых точек на изображениях с использованием OpenCV-Python

О.К. Головнин[1], Д.В. Рыбников[1]

[1]*Самарский национальный исследовательский университет им. академика С.П. Королева, Московское шоссе 34а, Самара, Россия, 443086*

Аннотация

В работе проведен сравнительный анализ алгоритмов AKAZE, BRISK, KAZE, ORB и SIFT, предназначенных для выделения особых точек на изображениях, в сочетании с алгоритмами BF и FLANN для сопоставления найденных особых точек между различными изображениями. Вычислительный эксперимент проведен с использованием обертки OpenCV-Python. Для каждой пары алгоритмов выполнена оценка времени обнаружения особых точек и подсчитано количество обнаруженных точек, а также оценено время сопоставления особых точек парами алгоритмов и подсчитано количество сопоставленных точек. Анализ показал, что наилучший результат по интегральному показателю оценки эффективности достигается при использовании пары алгоритмов ORB-FLANN.

Ключевые слова

OpenCV, локальный детектор, дескриптор особой точки, компьютерное зрение

1. Введение

Алгоритмы выявления и сопоставления особых точек на изображениях широко используются при разработке систем мониторинга дорожной обстановки и автономного вождения [1], в средствах биометрической идентификации [2] и при реконструкции трехмерных сцен в дополненной реальности [3]. Библиотека компьютерного зрения OpenCV содержит реализации известных алгоритмов выделения, сопоставления и анализа особых точек [4], однако вопрос выбора алгоритмов детектор-сопоставитель остается открытым, особенно если речь идет о реализации систем, требующих сравнения и поиска похожих изображений с учётом их геометрических трансформаций в режиме онлайн.

В этой работе проведен сравнительный анализ алгоритмов AKAZE, BRISK, KAZE, ORB и SIFT, предназначенных для выделения особых точек, в сочетании с алгоритмами BF и FLANN для их сопоставления между изображениями, при этом используется обертка OpenCV-Python.

2. Методика оценки и результаты вычислительного эксперимента

Вычислительный эксперимент проведен на рабочей станции под управлением Windows 10, оснащенной процессором Intel Core i5-8600K и 16GB DDR4 2666 MHz. Сравнение пар алгоритмов проведено на наборе данных Лундского университета GustavIIAdolf [5], выбранного ввиду возможности его применения для реконструкции трехмерной сцены. Программная реализация осуществлена на языке высокого уровня Python 3.6.0 с использованием библиотек opencv-python 4.5.1.48 и numpy 1.19.5. В качестве средства записи результатов исследований использовалась библиотека openpyxl 3.0.5.

Для сравнения пар алгоритмов использована метрика, предложенная в [6]:

$$\mathrm{N} = k\,[(fd + fm)(td_{min} + tm_{min})/(td + tm)/(fd_{max} + fm_{max})], \qquad (1)$$

где N – интегральная оценка; k – константа; fd – число точек, обнаруженных данным детектором; fd_{max} – максимальное число обнаруженных точек данным детектором; fm –

число точек, сопоставленных на смежных изображениях данным алгоритмом сопоставления; fm_{max} – максимальное число сопоставленных точек данным алгоритмом сопоставления; td – время, затраченное на обнаружение точек fd; td_{min} – минимальное время, затраченное на обнаружение точек; tm – время, затраченное на сопоставление точек fm; tm_{min} – минимальное время, затраченное на сопоставление точек.

В ходе выполнения эксперимента выполнена покадровая оценка времени обнаружения особых точек детекторами и подсчитано количество обнаруженных точек, а также покадрово оценено время сопоставления особых точек парами алгоритмов и подсчитано количество сопоставленных точек. На Рисунке 1 приведён интегральный показатель оценки эффективности пар алгоритмов, рассчитанный для указанного набора данных согласно (1).

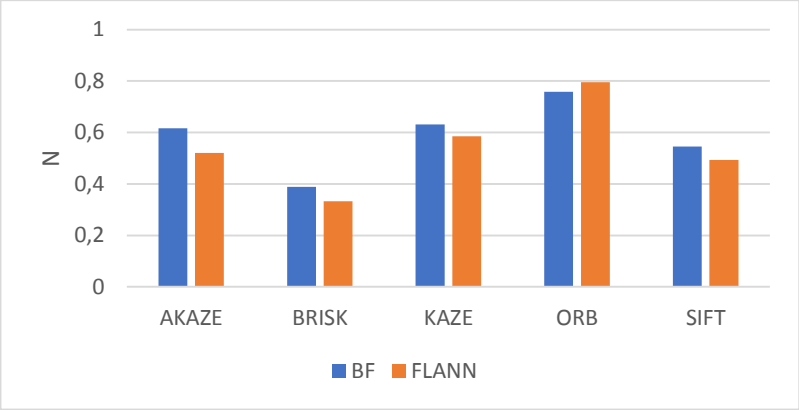

Рисунок 1: Результат оценки эффективности пар алгоритмов

В ходе сравнения пар алгоритмов установлено, что выбор алгоритма сопоставления FLANN, который позиционируется как достаточно быстрый, не дает гарантий получения выигрыша в производительности против BF, особенно если алгоритм использует стандартные настройки коэффициентов. Кроме того, BF справляется с задачей сопоставления точек гораздо лучше FLANN. Наилучший результат в совокупности показала пара ORB-FLANN, что достигнуто благодаря детектору FAST и дескриптору BRIEF в составе ORB.

3. Заключение

Таким образом, проведен сравнительный анализ 10 пар алгоритмов детектор-сопоставитель, при этом использовалась обертка OpenCV под экосистему Python. Анализ показал, что наилучший результат по интегральному показателю оценки эффективности достигается при использовании пары алгоритмов ORB-FLANN.

4. Литература

[1] Golovnin, O.K. Video processing method for high-definition maps generation / O.K. Golovnin, D.V. Rybnikov // Int. Conf. on Ind. Engineering and Modern Technologies. – 2020. – P. 1-5.
[2] Mohanraj, V. Robust face recognition system in video using hybrid scale invariant feature transform / V. Mohanraj // Procedia Computer Science. – 2016. – Vol. 93. – P. 503-512.
[3] Cheng, M. An augmented reality image registration method based on improved ORB / M. Cheng, L. Zhang, L. Liu // J. of Physics: Conf. Series. – 2020. – Vol. 1544(1). – P. 012113.
[4] Pulli, K. Real-time computer vision with OpenCV / K. Pulli, A. Baksheev, K. Kornyakov, V. Eruhimov // Communications of the ACM. – 2012. – Vol. 55(6). – P. 61-69.
[5] Lund University's motion pipeline data set [Electronic resource]. – Access mode: http://www.maths.lth.se/matematiklth/personal/calle/dataset/dataset.html (26.01.2021).
[6] Noble, F.K. Comparison of OpenCV's feature detectors and feature matchers / F.K. Noble // Int. Conf. on Mechatronics and Machine Vision in Practice. – 2016. – P. 1-6.