Дигитално Процесирање на Слика

Тема: Споредба на Локални Дескриптори

Горазд Филиповски 223070

Мотивација

Во областа на компјутерската визија, локалните карактеристики играат клучна улога во препознавање и споредување на објекти, елементи и структури во дигитални слики. Овие локални дескриптори овозможуваат алгоритмите „да забележат“ специфични карактеристики, како што се агли, рабови или карактеристични детали, кои потоа можат да се поврзат или „совпаднат“помеѓу две или повеќе слики.

Мотивацијата за овој проект произлегува од потребата да се разбере како различните типови локални дескриптори се однесуваат во различни сценарија и услови. Постојат повеќе популарни методи, како што се SIFT (Scale-Invariant Feature Transform), ORB (Oriented FAST and Rotated BRIEF), BRIEF (Binary Robust Independent Elementary Features), BRISK (Binary Robust Invariant Scalable Keypoints), AKAZE, KAZE и RootSIFT. Секој метод има свои предности и недостатоци. На пример, некои методи се побрзи и подобро се прилагодуваат за реално-времени апликации, додека други методи обезбедуваат поголема точност или стабилност при разни трансформации (скалирање, ротација, осветлување итн.).

Во денешниот свет, каде што сé почесто се среќаваме со системи за препознавање на лица, автономни возила и мобилни апликации кои користат проширена реалност, изборот на соодветен дескриптор е од клучно значење. Погрешниот избор може да доведе до нестабилност во резултатите, неточни совпаѓања или предолго време на извршување. Затоа, систематското споредување на перформансите на различни дескриптори станува исклучително важно за истражувачите и инженерите во ова поле. Основната цел на овој проект е токму да даде објективна споредба на листа локални дескриптори преку анализа на нивните квалитети на повеќе пара слики, и да покаже каде секој од нив е најпогоден.

Цели на Проектот

Основните цели на овој проект се следните:

1. **Споредба на повеќе дескриптори**: Да се евалуираат и споредат SIFT, ORB, BRIEF, BRISK, AKAZE, KAZE и RootSIFT на множество од различни парови слики со варирано осветлување, агол или содржина.

2. **Оцена на клучните точки (keypoints)**: Да се измери колку точки идентификува секој дескриптор во различни слики и каква е стабилноста на овие точки.

3. **Мерење на „добри совпаѓања“**: Да се анализира бројот на добри совпаѓања (good matches) меѓу клучните точки за секоја комбинација на дескриптори помеѓу двете слики.

4. **Презентирање на визуелни резултати**: Да се прикажат графички споредби (бар-дијаграми, линиски дијаграми, топлински мапи) и примери од конкретни слики со нацртани совпаѓања.

5. **Формирање заклучоци**: Врз база на добиените резултати, да се изведат заклучоци за тоа кој дескриптор е најсоодветен за одредени услови и како тие податоци можат да се искористат за идни системи и истражувања.

Опсег на Проектот

Опфатот на овој проект е фокусиран главно на споредба на перформансите на познати локални дескриптори врз база на:

- **Број на детектирани карактеристични точки**: Колку точки секој алгоритам може да препознае во една слика.

- **Број на успешни („добри“) совпаѓања**: Кога ќе се споредат дескрипторите помеѓу две слики, колку од тие совпаѓања се валидни и со добра точност.

- **Визуелизација на резултатите**: Бар-дијаграми, линиски дијаграми, топлински мапи и директни прикажувања на совпаѓањата помеѓу две слики.

Општо за алгоритмите

Преглед на Локалните Дескриптори

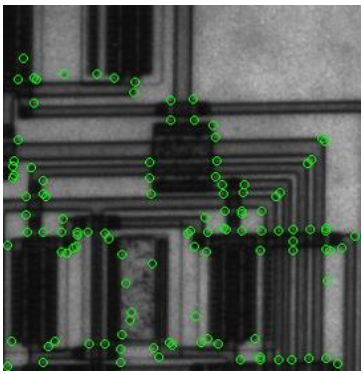
Алгоритмите кои ги вклучуваат локалните дескриптори се дизајнирани да детектираат и опишуваат карактеристични точки во една слика. Со фокус на делови од сликата кои се повторливи и препознатливи, како агли или области со изразени текстури, овие методи овозможуваат споредување (совпаѓање) на истите карактеристики меѓу различни слики, дури и кога има варијации во скалата, аголот, ротацијата или осветлувањето.

Раните методи за локални дескриптори се поврзани со детектори на агли, како што е Harris детекторот, кој потоа е надграден со скалирачка и ориентациска инваријантност (SIFT, SURF). Со текот на времето се појавиле побрзи бинарни дескриптори, како ORB, BRIEF и BRISK, особено погодни за реално-времени апликации кои побаруваат побрзо време на извршување.

Клучни Концепти во Детекција и Опис на Карактеристики

1. **Детекција на Клучни Точки (Keypoint Detection)**

Во овој чекор се бараат области со висока уникатност, обично каде што интензитетот во сликата се менува во повеќе насоки. Техниките се разликуваат: некои се базираат на градиентни пресметки (SIFT, KAZE, AKAZE), додека други користат аголни детектори (ORB) или специфични пристапи (STAR за BRIEF).

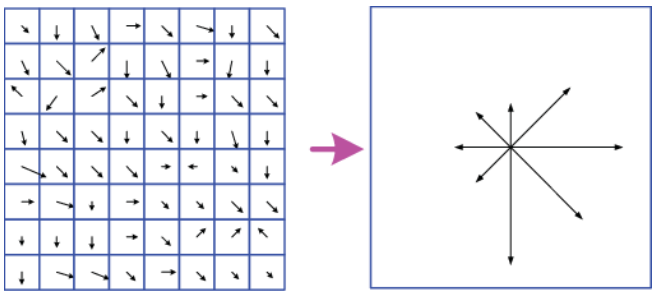


2. **Пресметка на Дескриптори (Descriptor Computation)**

Кога ќе се лоцира клучната точка, се пресметува дескриптор кој обезбедува репрезентација на локалното подрачје околу клучната точка. Дескрипторите можат да бидат:

-**Базирани на реален број (float-based)** (SIFT, KAZE, AKAZE итн.)

- **Бинарни (binary-based)** (ORB, BRIEF, BRISK), кои обично водат до побрзо совпаѓање.

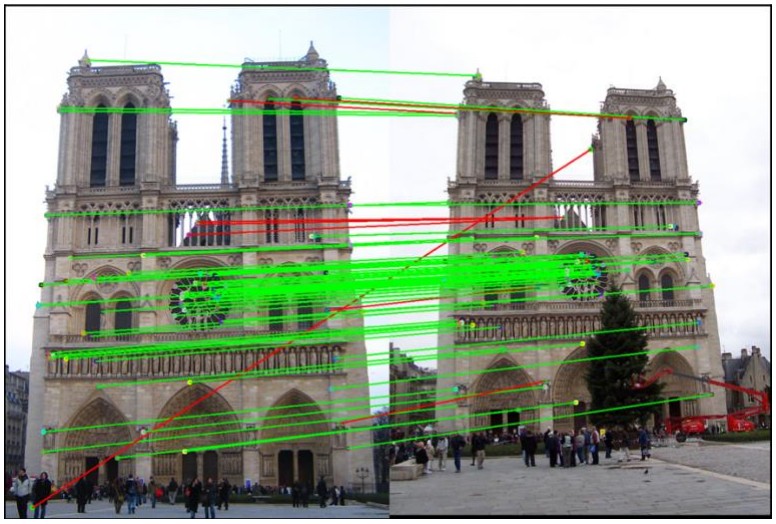


3. **Инваријантност (Invariance)**

Дескрипторите се стремат да бидат инваријантни на промени во скалата, ротацијата, осветлувањето или слични трансформации. Некои методи експлицитно ја адресираат скалата и ротацијата (SIFT, KAZE), додека други зависат од детекторот (како во случајот на BRIEF).

4. **Совпаѓање (Matching)**

По екстракцијата на дескрипторите, совпаѓањето ги поврзува карактеристичните точки меѓу две слики според некоја метрик аза растојание. За дескрипторите базирани на реален број, најчесто се користи Евклидово растојание (L2), а за бинарните—Hamming растојание.



Алгоритми

SIFT (Scale-Invariant Feature Transform)

SIFT е еден од највлијателните алгоритми за детекција и опис:

- **Детекција**: Користи Difference of Gaussians (DoG) за идентификување на потенцијални клучни точки на повеќе скали.

- **Дескриптор**: Конструира хистограм на ориентации на градиентот околу секоја клучна точка, нормализиран за ориентација и скала.

- **Предности**: Висока робустност на скала, ротација и умерени промени во гледна точка.

- **Недостатоци**: Поспоро е споредено со бинарните дескриптори

ORB (Oriented FAST and Rotated BRIEF)

ORB е брза и ефикасна алтернатива на SIFT:

- **Детекција**: Го користи FAST (Features from Accelerated Segment Test) за брзо детектирање на агли.

- **Дескриптор**: Верзија на BRIEF која е поотпорна на ротација.

- **Предности**: Многу брз, погоден за реално-времени апликации, користи бинарни дескриптори со Hamming растојание.

- **Недостатоци**: Не е толку робустен како SIFT при големи промени во скалата или ако сликите се многу заматени.

BRIEF (Binary Robust Independent Elementary Features)

BRIEF е насочен кон едноставни споредби на парови на интензитети во една слика:

- **Детекција**: BRIEF е само дескриптор, па обично се користи со посебен детектор (на пр. FAST, STAR).

- **Дескриптор**: Се добива бинарна низа со споредба на интензитетите на пикселите во подрачјето околу секоја клучна точка.

- **Предности**: Исклучително брз и со мала мемориска потрошувачка.

- **Недостатоци**: Нема вградена инваријантност на скала или ротација.

BRISK (Binary Robust Invariant Scalable Keypoints)

BRISK вклучува и детектор и дескриптор:

- **Детекција**: Користи кружен патерн и пристап со повеќе скали за да најде клучни точки.

- **Дескриптор**: Употребува патерн за парно споредување на интензитетите, опфаќајќи повеќе скали.

- **Предности**: Подобра инваријантност на скала од обичен BRIEF.

- **Недостатоци**: Може да е послаб од SIFT при екстремни трансформации.

AKAZE (Accelerated KAZE)

AKAZE се базира на KAZE алгоритмот, но е побрз:

- **Детекција/Дескриптор**: Користи нелинеарни скалирачки простори за детекција, што овозможува подобро задржување на деталите отколку кај гаусов-базираните методи.

- **Предности**: Добар баланс меѓу брзина и точност, со поддршка за бинарни или флоат дескриптори.

- **Недостатоци**: Сè уште не е толку брз како целосно бинарните пристапи (на пр. ORB).

KAZE

KAZE е сличен принцип на AKAZE, но без оптимизациите за забрзување:

- **Детекција**: Нелинеарната дифузија овозможува пронаоѓање клучни точки во скалирачки простор што подобро ги следи границите во сликата.

- **Дескриптор**: Базиран на реални броеви, пресметан на повеќе скали.

- **Предности**: Многу точен при различни услови.

- **Недостатоци**: Позахтевен за обработка во споредба со многу други методи.

RootSIFT

RootSIFT не е посебен детектор, туку постпроцес на SIFT:

- **Пресметка**: Прво се пресметуваат стандардни SIFT дескриптори, потоа секој дескриптор се L1-нормализира и потоа се зема квадратен корен на секој елемент.

- **Мотивација**: Подобро се справува со „bursty“ карактеристики и ги зголемува метриките за прецизност и одзив споредено со обичен SIFT.

- **Предности**: Едноставно за имплементација, може да даде поголема точност.

- **Недостатоци**: Се базира на SIFT; бара малку дополнителна пресметка.

Повеќе споредбени студии врз овие дескриптори обично истакнуваат неколку трендови:

- **Точност наспроти Брзина**: Дескрипторите базирани на реални броеви (SIFT, KAZE, AKAZE, RootSIFT) покажуваат висока точност, но се побавни.

- **Бинарни Методологии**: ORB, BRIEF и BRISK се предпочитани во системи со ограничени ресурси заради брзината и малата мемориска големина.

- **Инваријантност кон Скала и Ротација**: SIFT, (Root)SIFT, KAZE и AKAZE обезбедуваат понагласена инваријантност; BRIEF не е инваријантен без дополнителен механизам за ротација; ORB делумно ја решава ротацијата со Rotated BRIEF.

Подесување на Проектот и Структура на Директориумите

Софтвер и Библиотеки

За да ги изведеме експериментите за споредба на дескрипторите, користиме неколку клучни Python библиотеки и алатки:

- **Python**: Програмскиот јазик што го користиме за сите скрипти и јупитер тетратки (notebooks).

- **OpenCV (opencv-python и opencv-contrib-python)**: Обезбедува функции за обработка на слики, детекција на карактеристики и пресметка на дескриптори. „contrib“ пакетот е потребен за некои алгоритми како SIFT (во постари верзии на OpenCV) и xfeatures (на пр. StarDetector, BriefDescriptorExtractor).

- **NumPy**: Основен модул за ефикасна нумеричка пресметка, манипулација со низи и трансфер на сликите меѓу функциите.

- **Pandas**: Се користи за табеларно управување со податоци, складирање резултати во DataFrame-ови и лесно групирање или агрегирање.

- **Matplotlib & Seaborn**: Библиотеки за графичко прикажување и визуелизација на податоци. Служат за креирање бар-дијаграми, линиски дијаграми, топлински мапи итн.

- **Jupyter (notebooks)**: Овозможува интерактивно истражување, прикажување на графикони и брзо прототипирање. Дел од анализите и визуелизациите се прават во ова опкружување.

- **PyCharm**: IDE што ја олеснува изработката на Python код, дебагирањето и целокупната организација на проектот.

- **requirements.txt**: Го содржи списокот на потребните зависности за извршување на проектот.

Зошто се користат овие библиотеки

- **OpenCV** има зрела имплементација за стандардни (SIFT, ORB, BRISK, AKAZE, KAZE) и дополнителни (RootSIFT, BRIEF) дескриптори на карактеристики.

- **NumPy** и **Pandas** овозможуваат брза манипулација со нумерички податоци, особено при споредба на метрики за клучни точки и складирање на резултатите.

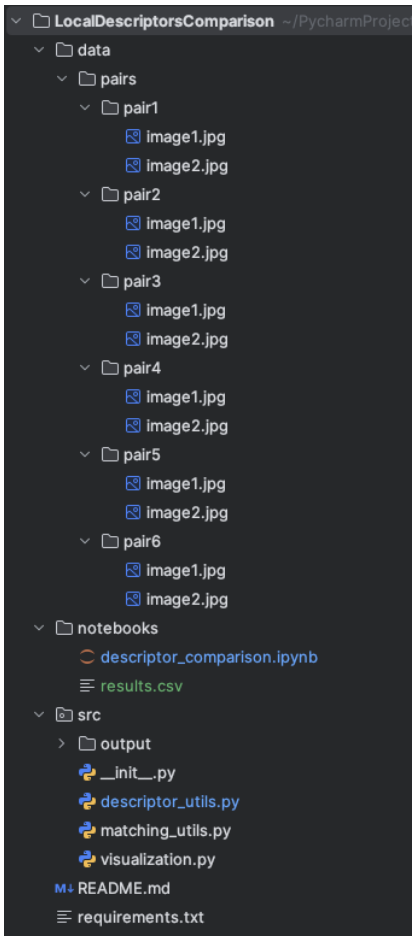
- **Matplotlib** и **Seaborn** се користат за визуелно прикажување на споредбите преку различни графикони и топлински мапи.

- **Jupyter notebooks** овозможуваат удобно комбинирање на код и резултати, што олеснува итеративна работа.

- **PyCharm** овозможува уредна структура на проектот со лесно уредување на изворниот код и управување со датотеките.

Директориуми / Организација на Датотеки

Претставена е поедноставена варијанта; вистинското дрво на директориуми може малку да варира:



Објаснување на Директориумите/Датотеките

- **data/pairs**: Содржи директориуми за секој пар слики (pair1, pair2, ...). Во секој директориум има две слики: image1.jpg и image2.jpg.

- **notebooks**: Ги вклучува Jupyter тетратките (notebooks) за експериментите и анализите. Во овој случај, descriptor_comparison.ipynb ја содржи главната скрипта/анализа.

- **src**: Содржи Python скрипти.

- **descriptor_utils.py**: Функции за екстракција на карактеристики со различни методи (SIFT, ORB, итн.).

- **matching_utils.py**: Функции за совпаѓање на карактеристики меѓу слики и пресметка на релевантни метрики.

- **visualization.py**: Функции за прикажување и визуелизација на совпаѓања, креирање графикони и зачувување излезни слики по потреба.

- **__init__.py**: Означува дека оваа папка е Python пакет, овозможувајќи релативни импорти.

- **requirements.txt**: Список на потребни библиотеки со нивните верзии.

Инструкции за Инсталација

За да започнете со проектот на нова машина:

1. **Клонирајте или Преземете**

- Клонирајте го репозиториумот (доколку постои на GitHub) или преземете ја папката како ZIP.

2. **Креирајте и Активирајте Виртуелно Опкружување (Препорачливо)**

```
python -m venv env
source env/bin/activate       # На macOS/Linux
.\env\Scripts\activate        # На Windows
```

3. **Инсталирајте Ги Зависностите**

Преминете во директориумот LocalDescriptorsComparison (каде што е requirements.txt) и инсталирајте ги библиотеките:

```
pip install -r requirements.txt
```

4. **Проверете Ги Инсталациите**

- Отворете PyCharm (или друга IDE) и уверете се дека интерпретерот е поврзан со вашето виртуелно опкружување.

- Пробајте import cv2, import numpy, и import seaborn во Python конзола или јупитер тетратка за да потврдите дека сé е правилно инсталирано.

5. **Извршете Ги Скриптите**

- Во PyCharm, отворете го descriptor_comparison.ipynb (во папката notebooks)

Методологија

Преглед / Подготовка на Податоците (Dataset Preparation)

Во овој проект, се користат **шест пара** слики, секој со специфични предизвици за локалните дескриптори:

1. **Пар 1**

- **Опис**: Ист 2D објект (книга) но поставен на две различни позадини.

- **Цел**: Тестира отпорност на дескрипторот кон промена на позадината, додека објектот е приближно на иста скала и ориентација.

2. **Пар 2**

- **Опис**: Книга што е **ротирана** на различен начин во двете слики.

- **Цел**: Проценува како секој дескриптор се справува со умерена до нагласена ротација во рамнина на релативно „рамен“ (2D) објект.

3. **Пар 3**

- **Опис**: Иста книга, но овојпат ротирана **и** со променето **осветлување**.

- **Цел**: Ги тестира дескрипторите во услови на комбинирани варијации – ротација и осветлување.

4. **Пар 4**

- **Опис**: Книгата е **скалирана, ротирана** и делумно опкружена со други објекти во втората слика.

- **Цел**: Оценува како дескрипторите се справуваат со промена на скала, ротација и додадени предмети што можат да внесат шум или да предизвикаат делумно покривање на објектот.

До овој момент, објектот (книгата) претставува главно 2D примерок, овозможувајќи тестирање на планарни трансформации и промени во осветлување.

5. **Пар 5**

- **Опис**: **3D статуа** на Abraham Lincoln, фотографирана од две различни гледни точки и со нагласена промена во скалата во втората слика.

- **Цел**: Преминува подалеку од 2D-случаите кон покомплексен, тридимензионален објект, тестирајќи ги дескрипторите со промени во перспектива и површини што не се копланарни.

6. **Пар 6**

- **Опис**: **3D статуа** на Статуата на Слободата, исто така сликана од две различни гледни точки и скали.

- **Цел**: Слично на Пар 5, но со друга статуа; проверува конзистентност на дескрипторите врз уште еден 3D објект, особено при ротација околу z-оската и промена на растојанието до камерата.

Екстракција на Карактеристики (Feature Extraction)

Екстракцијата на карактеристики се изведува преку модулот **descriptor_utils.py**. За секоја слика, ја повикувате функцијата extract_features со зададен метод (на пр. SIFT, ORB, BRIEF, BRISK, AKAZE, KAZE, ROOTSIFT). Оваа функција прави две важни работи:

1. **Детекција на Клучни Точки (Keypoint Detection)**

- Во зависност од методот, или користи вграден детектор-дескриптор пар (на пр. SIFT, ORB, BRISK, KAZE, AKAZE) или посебна комбинација детектор-дескриптор (како кај BRIEF, што подразбира STAR детектор во вашиот код).

2. **Пресметка на Дескрипторите (Descriptor Computation)**

- Откако ќе се детектираат клучните точки, се пресметува дескриптор кој ја карактеризира локалната околина на секоја клучна точка.

- За BRIEF, користите **StarDetector** за детекција и **BriefDescriptorExtractor** за бинарниот дескриптор.

- За RootSIFT, прво се добиваат стандардни SIFT дескриптори, потоа се применува RootSIFT пост-процес (L1-нормализација + квадратен корен).

Зошто STAR + BRIEF?

- BRIEF сам по себе е исклучиво дескриптор и бара надворешен детектор. Тука е избран **STAR** поради брзината и ефективноста при некои типови на текстури.

- Другите методи (SIFT, ORB, BRISK, AKAZE, KAZE) вклучуваат свои детектори, па затоа се едноставно повикани директно.

Совпаѓање на Карактеристики (Feature Matching)

Откако дескрипторите ќе се екстрахираат, тие се совпаѓаат со функции од **matching_utils.py**:

1. **Пристап за Совпаѓање**

- Обично, кодот повикува match_features(desc1, desc2, method="bf", ratio_test=True).

- **BFMatcher** (Brute Force Matcher) ги споредува сите дескриптори од првата слика со сите од втората, користејќи **L2** растојание (за флоат-базирани дескриптори, како SIFT или KAZE) или **Hamming** (за бинарни дескриптори, како ORB или BRIEF).

- Вклучен е **ratio test** (Lowe's ratio test) за да се отфрлат совпаѓања кои не се доволно разликувачки од второнајдобриот кандидат.

2. **Технички Детали**

- Доколку некој од дескрипторите е None, функцијата враќа прана мапа.

- За non-binary дескриптори со FLANN, автоматски се конвертираат во float32, но моменталниот код претпочита BFMatcher.

Метрики за Перформанси (Performance Metrics)

Го оценувате работењето на секој дескриптор преку следниве метрики:

1. **Број на Клучни Точки**

- Се добива преку должината на низите со клучни точки (len(kp1) и len(kp2)). Ова покажува колку „широк" или „рестриктивен" е детекторот.

2. **Број на „Добри Совпаѓања"**

- По примената на ratio test, бројот на преостанати совпаѓања се смета за „добри." Оваа метрика отсликува колку добро дескрипторот ги квантифицира локалните карактеристики, што резултира со сигурни совпаѓања.

Детали за Имплементацијата

Работен Тек од високо ниво (High-Level Workflow)

Процесот за споредба на локалните дескриптори во овој проект може да се сумира на следниот начин:

1. **Вчитување на Слики**

- Скриптата или јупитер тетратката (notebook) итерира преку паровите на слики (pair1, pair2 итн.) во директориумот data/pairs.

- За секој пар се вчитуваат двете слики (image1.jpg, image2.jpg) во grayscale режим.

2. **Екстракција на Карактеристики**

- Со избраниот дескриптор (SIFT, ORB, BRIEF итн.) се повикува extract_features() од descriptor_utils.py.

- Овој чекор идентификува клучни точки во секоја слика и ги пресметува соодветните дескриптори.

3. **Совпаѓање на Дескриптори**

- Скриптата ја користи match_features() од matching_utils.py, најчесто со BFMatcher (Brute Force) и ratio test за отфрлање на слабите совпаѓања.

4. **Пресметка на Метрики**

- За секој сет совпаѓања, кодот го брои бројот на клучни точки во секоја слика и бројот на „добри“ совпаѓања што преостанале по ratio test, преку compute_match_metrics().

5. **Анализа и Складирање на Резултатите**

- Секој резултат (назив на пар, тип на дескриптор, број на клучни точки, број на добри совпаѓања) се додава во листа или DataFrame.

- Податоците потоа се анализираат и сумираат, на пример со pivot табели, бар-дијаграми и топлински мапи.

6. **Визуелизација на Совпаѓањата**

- Опционално, функциите за визуелизација (на пр. draw_matches() или draw_colored_matches()) прикажуваат подмножество на совпаѓања директно врз сликите, поврзувајќи линии меѓу одговарачките клучни точки.

Овој циклус се повторува за секој дескриптор и за секој пар слики, создавајќи сеопфатна слика за тоа како секој метод работи во различни сценарија.

Структура на Кодот

Во src папката има три главни Python датотеки со клучните функционалности:

1. **descriptor_utils.py**

- **Цел**: Екстракција на клучни точки и дескриптори од дадена слика.

- **Главна функција: extract_features(image, method="SIFT")**

- Ја избира соодветната OpenCV детектор/дескриптор комбинација (на пр. cv2.SIFT_create(), cv2.ORB_create()), или поставува посебна постапка (на пр. STAR + BRIEF).

- Враќа листа клучни точки и соодветна низа со дескриптори.

- Го обработува посебниот случај RootSIFT со примена на L1-нормализација и квадратен корен врз стандардните SIFT дескриптори.

2. **matching_utils.py**

- **Цел**: Совпаѓање на дескриптори од две слики и пресметка на релевантни метрики.

- **Главни функции**:

- match_features(desc1, desc2, method="bf", ratio_test=True):

- По дифолт користи BFMatcher. Доколку е потребно, ги конвертира дескрипторите во соодветен тип (float или binary).

- Го применува Lowe’s ratio test за филтрирање на двосмислени совпаѓања.

- compute_match_metrics(keypoints1, keypoints2, matches):

- Враќа речник (dictionary) со број на клучни точки во секоја слика и колку совпаѓања се најдени.

3. **visualization.py**

- **Цел**: Прикажување на совпаѓањата и визуелен приказ на перформансите на дескрипторите.

- **Главни функции**:

- draw_matches(...), draw_thick_matches(...), draw_colored_matches(...):

- Прекрива линии меѓу совпаѓачките клучни точки во двете слики, по желба со различни бои или подебели линии за нагласување на квалитетот на совпаѓањата.

- Резултатите се прикажуваат со Matplotlib, што овозможува лесно зачувување или прикажување во јупитер тетратката (notebook).

Јупитер Тетратки (Notebooks)

Главната анализа се извршува во Jupyter ноутбукот **descriptor_comparison.ipynb**, лоциран во папката notebooks. Овој ноутбук (или .py скрипта, ако е конвертирана) го координира целиот работен тек:

- **Импортира**: Ги вчитува функциите од descriptor_utils.py, matching_utils.py и visualization.py.

- **Итерира**: Придвижува секој пар слики, ги повикува посакуваните дескриптор методи и добива метрики за секоја комбинација (дескриптор, пар на слики).

- **Складира**: Резултатите се чуваат во Pandas DataFrame (df_results) и со библиотеки како Matplotlib и Seaborn се генерираат графикони (бар-дијаграми, топлински мапи, линиски дијаграми итн.).

- **Визуелизација**: По желба, се прикажуваат совпаѓањата врз некои од сликите, за да се види како секој дескриптор се однесува на конкретни примери.

Експериментални Резултати

Статистика за Клучни Точки (Keypoint Statistics)

Главен дел од овој проект е да се разбере колку клучни точки детектира секој дескриптор и како тоа варира низ различни парови слики. Тука ги разгледуваме **num_keypoints1** (клучни точки во првата слика) и **num_keypoints2** (клучни точки во втората слика).

1. **Општи Забелешки**

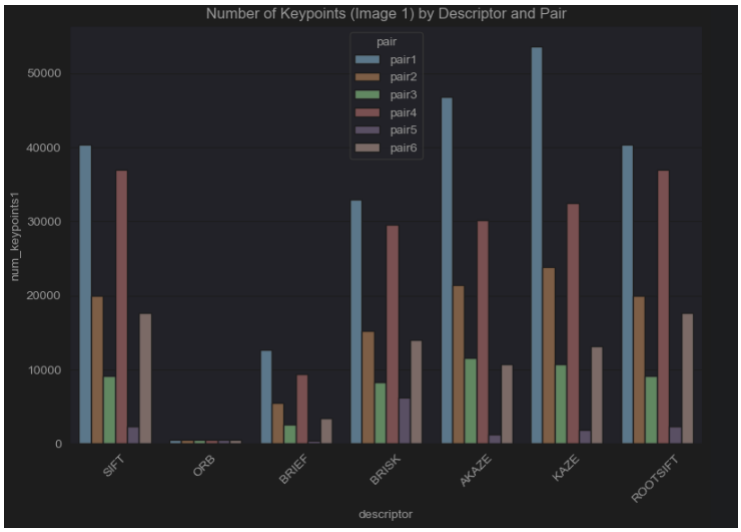
- **SIFT, AKAZE, KAZE и ROOTSIFT** често создаваат многу клучни точки (понекогаш над 40.000 по слика).

- **ORB** по дифолт е ограничен на 500 клучни точки во стандардната OpenCV имплементација, па затоа во сите парови пријавува максимум 500 точки, без оглед на содржината.

- **BRIEF** и **BRISK** се најчесто „некоја средина“, детектирајќи од неколку илјади до околу 30.000 клучни точки во зависност од парот на слики.

2. **Бар-Дијаграм за Клучни Точки**

- Дијаграмот “Number of Keypoints (Image 1) by Descriptor and Pair” покажува значителни врвови кај дескрипторите како KAZE и SIFT во одредени парови (на пр., Pair 1 и Pair 4). ORB е секогаш на 500, согласно неговата ограниченост.



- Во Pair 6 (Статуата на Слободата од различни агли), SIFT и KAZE сè уште детектираат илјадници точки, додека ORB и понатаму има само 500. Овој јасен диспаритет ја нагласува улогата на почетните параметри и дизајнот на самиот дескриптор.

3. **Резултати по Парови слики**

- **Pair 1** (книга со различни позадини): KAZE детектира над 53k клучни точки во првата слика, додека ORB е на 500. SIFT и ROOTSIFT се околу 40k.

- **Pair 5** (статуа на Abraham Lincoln): Бројот на клучни точки се намалува за повеќето дескриптори во споредба со „планарните“ книги, веројатно зашто 3D скулптурата нема толку изразени агли или текстури од одредени агли.

- **Pair 6** (Статуата на Слободата): Слична појава како кај Pair 5—иако SIFT, AKAZE и KAZE детектираат илјадници точки, тоа е помалку отколку кај книжните слики каде што има повеќе „планарни“ детали.

Анализа на Добри Совпаѓања (Good Matches Analysis)

Покрај бројот на клучни точки, бројот на „добри совпаѓања“ (по ratio test) дава претстава за тоа колку ефикасно дескрипторот ги поврзува карактеристиките меѓу двете слики во секој пар.

1. **Pivot Табела и Heatmap**

- Направена е pivot табела (index = pair, columns = descriptor, values = num_good_matches), а потоа визуелизирана со Seaborn heatmap.

- **Pair 1**:

- **KAZE** прикажува најголем број добри совпаѓања (17.494), блиску следен од **AKAZE** (11.702) и **ROOTSIFT** (9.239).

- **ORB** има само 3 добра совпаѓања, што е во согласност со лимитираните 500 клучни точки.

- **Pair 2**:

- **KAZE** повторно се истакнува со 10.390 добри совпаѓања, додека **AKAZE** е на 8.094. SIFT добива 5.373, а ORB е на 146.

- **Pair 3 и Pair 4**:

- Двете покажуваат опаѓање во бројот на добри совпаѓања, веројатно поради комбинирани трансформации (ротација, скала, мешање со други објекти). KAZE и AKAZE и понатаму водат, но вкупните бројки се помали отколку кај Pair 1 или Pair 2.

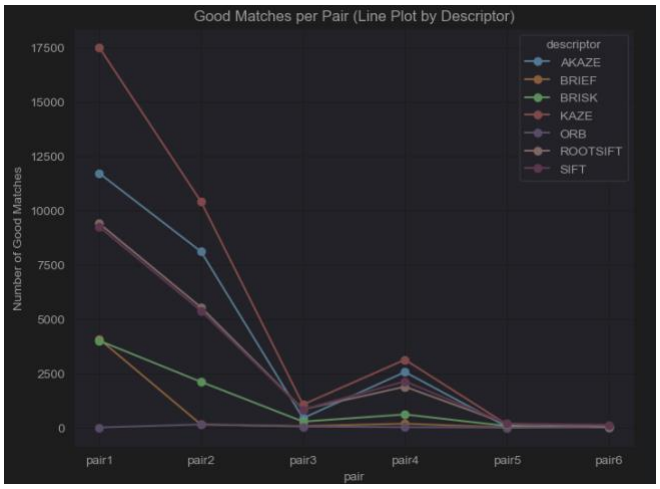
- **Pair 5 и Pair 6** (3D статуи):

- Бројот на совпаѓања е значително помал. SIFT варира од ~127 до ~179 добри совпаѓања, додека KAZE се движи околу 142–46. BRIEF, BRISK, ORB и AKAZE во некои случаеви

имаат уште помал број, укажувајќи на поголемата сложеност за совпаѓање на 3D форми од различни агли.

2. **Линиски Дијаграми (Line Plots)**

- Дијаграмот “Good Matches per Pair (Line Plot by Descriptor)” нагласува дека KAZE и AKAZE водат кај Pair 1–4, додека SIFT е умерено висок.



- Кај Pair 5–6, повеќето дескриптори се движат кон пониски вредности, што ја отсликува сложеноста при совпаѓање на 3D статуи со поголеми промени во перспективата.

3. **Интересни Забелешки**

- **ORB** и неговиот фиксно лимитиран број клучни точки (500) силно го ограничуваат и бројот на можни добри совпаѓања, особено на слики каде што другите дескриптори наоѓаат десетици илјади карактеристики.

- **KAZE и AKAZE** често постигнуваат високи вредности и за клучни точки и за добри совпаѓања, особено во сцени со повеќе текстури и помалку екстремни трансформации (Pair 1, Pair 4).

- **SIFT vs. ROOTSIFT**: ROOTSIFT обично прикажува сличен број добри совпаѓања како SIFT, а понекогаш дури и малку подобро. Во Pair 1, на пример, и двата се околу 9.239.

Визуелизации

Покрај нумеричките табели, креирани се разни графикони и директни визуелизации:

1. **Бар-Дијаграми**

- Ги прикажуваат бројот на клучни точки во сликите (групирани по пар и дескриптор) и бројот на добри совпаѓања. Високите вредности кај SIFT/KAZE лесно се забележуваат; ORB е видливо пониско.

2. **Heatmap**

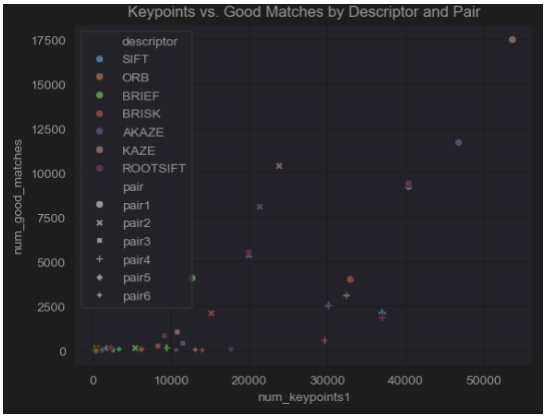
- На матричен приказ ги резимира “Number of Good Matches,” каде потемни или посветли полиња веднаш покажуваат каде одреден дескриптор надмоќно или слабо се справува.

3. **Линиски Дијаграми**

- Прикажуваат како бројот на добри совпаѓања варира низ паровите за секој дескриптор. Корисно е за препознавање значајни падови (како од Pair 1 до Pair 3 кај некои методи).

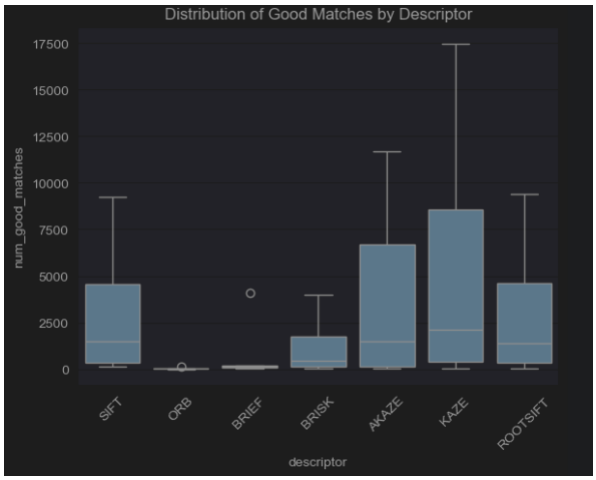
4. **Scatter Plot (Keypoints vs. Good Matches)**

- Потврдува дека поголем број клучни точки не секогаш носи пропорционален раст на добрите совпаѓања. На пример, KAZE може да детектира 50k точки, но не сите се совпаѓаат успешно.



5. **Box Plot**

- Прикажува како добрите совпаѓања се распределени за секој дескриптор, укажувајќи дека KAZE има највисок максимален опсег, додека ORB е секогаш на дното.



6. **Пример Визуелизации на Совпаѓања**

- Доколку се вклучени, овие слики би прикажувале линии што ги поврзуваат совпаѓачките клучни точки. Кај поуспешните дескриптори се гледа густа мрежа на релативно точни совпаѓања, додека кај други (како ORB или BRIEF) може да има значително помал број или невралевантни совпаѓања во ротации или преклопувања.

Клучни Заклучоци

- **KAZE** и **AKAZE** доминираат во поглед на детекција на клучни точки и вкупен број добри совпаѓања, особено во полесни 2D сцени.

- **SIFT** е исто така силен, особено кај планарни сцени (Pair 1–4), додека RootSIFT покажува слични или малку подобри резултати во некои случаи.

- **ORB** е лимитиран со фиксно 500 клучни точки, што го става во неповолна состојба кај покомплексни сцени, но може да биде полезен за реално-времени системи каде што брзината е приоритет.

- **BRIEF** и **BRISK** нудат умерени резултати, кои понекогаш се засенети од SIFT/KAZE во поглед на вкупен број совпаѓања, но може да имаат предност во случаи каде што е потребна брзина и помала големина на дескрипторот.

Овие експерименти покажуваат дека не постои еден „универзално најдобар“ дескриптор: изборот силно зависи од видот на сцената (2D наспроти 3D), нивото на трансформации (ротација, скала, осветлување) и ограничувањата поврзани со брзината (пр. во реално-времени апликации).

Користена Литература

1. **SIFT**
Lowe, D. G. (2004). *Distinctive image features from scale-invariant keypoints.*
International Journal of Computer Vision, 60(2), 91–110.
https://doi.org/10.1023/B:VISI.0000029664.99615.94
2. **ORB**
Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). *ORB: An efficient alternative to SIFT or SURF.*
In Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2564–2571.
https://doi.org/10.1109/ICCV.2011.6126544
3. **BRIEF**
Calonder, M., Lepetit, V., Strecha, C., & Fua, P. (2010). *BRIEF: Binary robust independent elementary features.*
In Proceedings of the European Conference on Computer Vision (ECCV), 778–792.
https://doi.org/10.1007/978-3-642-15561-1_56
4. **BRISK**
Leutenegger, S., Chli, M., & Siegwart, R. (2011). *BRISK: Binary robust invariant scalable keypoints.*
In Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2548–2555.
https://doi.org/10.1109/ICCV.2011.6126542
5. **KAZE**
Alcantarilla, P. F., Bartoli, A., & Davison, A. J. (2012). *KAZE features.*
In Proceedings of the European Conference on Computer Vision (ECCV), 214–227.
https://doi.org/10.1007/978-3-642-33863-2_16
6. **AKAZE**
Alcantarilla, P. F. (2012). *Fast explicit diffusion for accelerated features in nonlinear scale spaces.*
In British Machine Vision Conference (BMVC).
https://doi.org/10.5244/C.26.13
(AKAZE builds upon KAZE with accelerated computations and can produce both binary and float descriptors.)
7. **RootSIFT**
Arandjelović, R., & Zisserman, A. (2012). *Three things everyone should know to improve object retrieval.*
In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2911–2918.
https://doi.org/10.1109/CVPR.2012.6248003
8. **OpenCV Documentation**
OpenCV Developer Team. (n.d.). *OpenCV Online Documentation.*
https://docs.opencv.org/