УДК 004.43

МАТЕМАТИЧНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ СИСТЕМИ КОМП'ЮТЕРНОГО СПОСТЕРЕЖЕННЯ

Кравчук Я.Я. [1)], **Лазорко А.Р.**[2)], **Костик Б.П.**[3)], **Сагайдак П.Р.**[4)], **Лісний А.Я.**[5)]
Західноукраїнський національний університет
[1)]магістрант; [2)]магістрант;[3)]аспірант;[4)]аспірант;[5)]аспірант

I. Постановка проблеми

В області автоматизованої обробки відеоданих можна виділити ряд завдань, пов'язаних з обробкою інформації про спостереження за даними, що надходять із кількох точок зору. Прикладами подібних завдань є спостереження/контроль з використанням системи камер; формування фотосхем на основі знімків, що одержуються з багатооб'єктивних камер; визначення просторових характеристик об'єктів сцени на основі аналізу стереозображень. Розвиток обчислювальних засобів і датчиків зображень призводить до постійного розширення сфери застосування автоматизованих систем комп'ютерного спостереження з декількома полями.

II. Мета роботи

Метою дослідження є розробка математичного та програмного забезпечення для системи комп'ютерного спостереження.

III. UML-діаграма класів компоненти для зіставлення зображень

На рисунку 1 представлена структурна UML-діаграма класів програмних компонентів зіставлення зображень. Клас I_Correlator реалізує послідовну схему обробки набору знімків:

1) створення корелятора та ініціалізація параметрів;
2) обробка чергового кадру;
3) оновлення поточних параметрів корелятора;
4) одержати масив відповідностей.

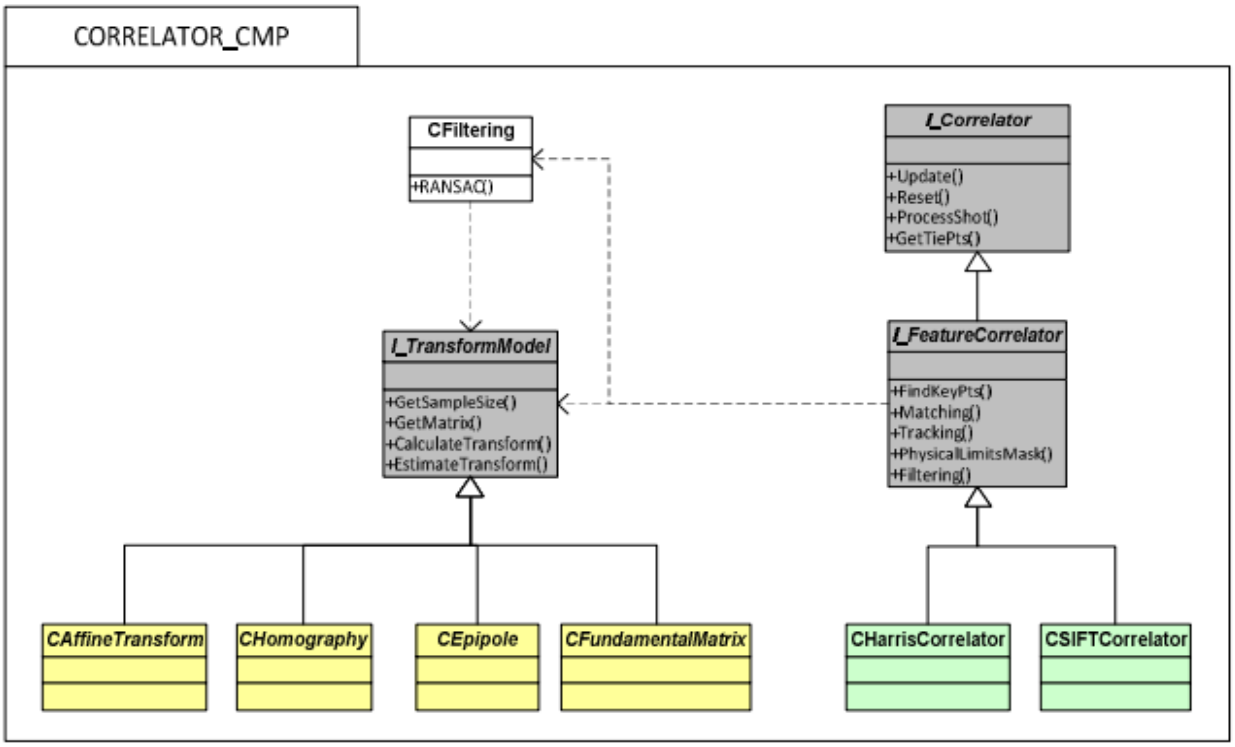

Рисунок 1 –UML-діаграма класів компонентів для зіставлення зображень

У класі I_FeatureCorrelator (успадкованому від I_Correlator) перевизначений метод обробки знімка ProcessShot і введені наступні абстрактні віртуальні методи:

1) FindKeyPoints – пошук характерних точок та формування масиву дескрипторів характерних точок;

2) Matching – зіставлення дескрипторів, з можливістю використання бінарної матриці фізичних обмежень (для її визначення існує метод PhysicalLimitsMask);

3) Filtering - фільтрація знайдених відповідностей на основі моделі обмежень (перевизначена для класу-інтерфейсу I_TransformModel);

4) Tracking – відстеження відповідності на парі знімків.

Для двох описаних схем кореляції реалізовано класи CHarrisCorrelator та CSIFTCorrelator (успадковані від абстрактного класу I_FeatureCorrelator).

Метод RANSAC реалізований у класі CFiltering як статична функція, цей метод використовується у функції-член Filtering класу I_FeatureCorrelator.

Розглянуті моделі обмежень реалізовані в класах CFundamentalMatrix (фундаментальна матриця), CHomography (гомографія) та CEpipole. Ці класи успадковані від абстрактного класу I_TransformModel. Клас I_TransformModel містить чотири абстрактні віртуальні функції:

1) отримання мінімального розміру вибірки для визначення параметрів перетворення;

2) отримання матриці перетворення;

3) обчислення параметрів перетворення;

4) оцінка узгодженості перетворення на наборі відповідностей.

Одним з поширених підходів до оптимізації обчислювального процесу в реалізації алгоритмів зіставлення зображень є використання апаратних засобів, що забезпечують паралельне виконання алгоритмів. У застосуванні цього підходу в прикладних задачах можна виділити два основні аспекти:

- вибір архітектури апаратних засобів з урахуванням способів подання та можливостей паралельної обробки даних;

- розробка та оптимізація алгоритмів стосовно обраної архітектури апаратного забезпечення.

Для обробки зображень, дані яких допускають матричне представлення, останнім часом широко застосовуються графічні процесори загального призначення (General Purpose Graphic Processor Unit, GPU).

Можливості програмування GPU загального призначення роблять їх привабливими для застосування в завданнях комп'ютерного зору з метою прискорення обробки та зниження навантаження на центральний процесор обчислювальної системи.

Модель програмування GPU CUDA передбачає, що всі дані надаються у вигляді сітки з блоків ниток (gridofthreadblocks). Програмування GPU полягає в написанні програмної реалізації для однієї функції виконуваної усіма нитками CUDA.

Для моделі програмування CUDA можна виділити такі основні операції:

1) виділення пам'яті на GPU;

2) копіювання даних з основної пам'яті (пам'ять CPU) у виділену пам'ять на GPU;

3) виклик функцій, що виконуються на GPU;

4) копіювання результатів обробки з пам'яті GPU в область пам'яті CPU;

5) звільнення пам'яті на GPU.

У даній моделі суттєві часові витрати пов'язані з копіюванням даних по шині, до якої підключено GPU, – шині PCI Express. Зменшення кількості викликів операцій копіювання між пам'яттю CPU↔GPU є типовим прийомом прискорення виконання алгоритмів [1-3]. Як правило, значне прискорення можливе при початковому проектуванні алгоритмів з урахуванням можливостей GPU, а також переваги CPU над GPU при реалізації ітераційних алгоритмів.

Висновок

Для задачі зіставлення зображень сформульовано та проаналізовано загальну схему на основі ознакових методів, на прикладі двох прикладних завдань (порівняння знімків, при аерофотозйомці та зйомці з БПЛА) описані два підходи до пошуку характерних точок та формування їхніх дескрипторів. Для програмної реалізації спроектованих програмних наводиться структурна UML-діаграма класів.

Список використаних джерел

1. Kirk, David B., and Wen-mei W. Hwu. Programming Massively Parallel Processors: A Hands-on Approach. Morgan Kaufmann, 2016.
2. Sanders, Jason, and Edward Kandrot. CUDA by Example: An Introduction to General-Purpose GPU Programming. Addison-Wesley, 2010.
3. Cook, Shane. CUDA Programming: A Developer's Guide to Parallel Computing with GPUs. Morgan Kaufmann, 2013.
4. Cheng, John, Max Grossman, and Ty McKercher. Professional CUDA C Programming. Wiley, 2014.
5. Storti, Duane, and Mete Yurtoglu. CUDA for Engineers: An Introduction to High-Performance Parallel Computing. Addison-Wesley, 2015.