УДК 004.624

Ніколайчук В.О.
Романюк О.В.

КЛАСТЕРНИЙ МЕТОД ПОШУКУ МОДИФІКОВАНИХ ДУБЛІКАТІВ ЗОБРАЖЕНЬ У ФОРМАТІ JPEG XR З ВИКОРИСТАННЯМ ПЕРЦЕПТИВНИХ ХЕШІВ

Вінницький національний технічний університет

Анотація

Проаналізовано кластерний метод пошуку модифікованих дублікатів зображень у форматі JPEG XR з використанням перцептивних хешів. Проведено аналіз основних методів, що застосовуються, визначено переваги та особливості використання методів для алгоритму кодування JPEG XR.

Ключові слова: JPEG XR, графічні файли, Microsoft, ущільнення зображень, пошук дублікатів, перцептивні хеш-алгоритми, aHash, bHash, dHash, mHash, pHash, wHash, комп'ютерне бачення, SIFT, SURF, FAST, BRIEF, ORB, кластеризація методом k-середніх ключових точок, відстань Геммінга.

Abstract

A new clustering method of finding modified image duplicates in JPEG XR format using perceptual hashes is analyzed. The basic methods used are analyzed, the advantages and peculiarities of using methods for the JPEG XR encoding algorithm are determined.

Keywords: JPEG XR, image files, Microsoft, image compression, duplicate search, perceptual hash algorithms, aHash, bHash, dHash, mHash, pHash, wHash, computer vision, SIFT, SURF, FAST, BRIEF, ORB, k-means clustering, Hamming distance

Вступ

При пошуку дублікатів зображень метрикою міри схожості зображень може виступати метрика, що характеризує відмінність зображень між собою. Схожими зображеннями будуть вважатися ті зображення, відмінність між якими менше деякого порогового значення. Для зображень з однаковими розмірами, зазвичай, такою метрикою відмінності слугує середньоквадратичне відхилення пікселів. Але якщо зображення було хоч якось модифіковано (накладено фільтр, обрізано зображення, додано чи видалено певний елемент), більшість наявних алгоритмів порівняння комп'ютерних зображень, наприклад, порівняння пікселів обох зображень між собою, не виявлять схожість між модифікованим зображенням та першоджерелом [1].

Враховуючи, що найбільш інноваційним форматом зображень є JPEG XR [2], а найкращі результати знаходження дублікатів досягаються при використанні порівняльних хешів [1], підвищення точності знаходження модифікованих дублікатів зображень у форматі JPEG XR за рахунок розробки нового методу з використанням порівняльних хешів є актуальною задачею.

Результати дослідження

Розглянемо основні функції побудови порівняльних хешів для графічних файлів:

1. Середній хеш [3] – перетворює вхідне зображення у відтінки сірого, а потім зменшує його. Далі обчислюється середнє значення всіх сірих значень зображення, а потім пікселі досліджуються по черзі зліва направо. Якщо значення сірого перевищує середнє значення, до хешу додається 1, інакше 0.
2. Блоковий хеш [4] – перетворює вхідне зображення у відтінки сірого, а потім ділить зображення на блоки та генерує значення для кожного блоку – 1 або 0. Ці значення об'єднуються послідовно зліва направо в хеш.
3. Хеш різниці [5] – подібно до середнього алгоритму хешування, алгоритм різниці хешів спочатку генерує зображення у відтінках сірого з вхідного зображення. З кожного рядка перші 8 пікселів досліджуються послідовно зліва направо і порівнюються з сусідом праворуч, що, аналогічно середньому алгоритму хешу, призводить до генерації хешу.

4. Медіанний хеш [3] – алгоритм медіанного хеша працює аналогічно середньому алгоритму хешу, за винятком того, що він не порівнює зі середнім значенням, а порівнює з медіаною.
5. Перцептивний хеш [6] – алгоритм перцептивного хешу також спочатку обчислює зображення у відтінках сірого та зменшує його. Далі до цього зображення застосовується дискретне косинусне перетворення спочатку на рядок, а потім на стовпець. Пікселі з високими частотами тепер розташовані у верхньому лівому куті, тому зображення обрізається у верхньому лівому куті. Далі обчислюється медіана сірих значень цього зображення та генерується зображення, аналогічно алгоритму медіанного хешу.
6. Вейвлет хеш [7] – аналогічно алгоритму середнього хешу, алгоритм хеш-вейвлет також генерує зображення у відтінках сірого значення розміром 8 × 8. Далі до зображення застосовується двовимірне вейвлет-перетворення. Потім, аналогічно алгоритму перцептивного хешу, кожен піксель порівнюється з медіаною, і обчислюється хеш.

Серед усіх наявних алгоритмів побудови порівняльних хешів, методи перцептивного та блокового хешу показали найкращі результати, але алгоритм перцептивного хешу обраховувався майже удвічі швидше, тому було вирішено обрати саме цей метод побудови порівняльного хешу для графічних зображень [8].

Оскільки використання одного перцептивного хешу не дає високу точність при порівнянні модифікованих зображень, було запропоновано генерувати не один хеш на ціле зображення, а кілька хешів – по одному для кожної ключової точки.

Власне, для визначення функціональних детекторів (ключових точок), найпопулярнішими є наступні методи: SIFT, SURF, FAST, BRIEF та ORB. Розглянемо кожен з них окремо.

Масштабно-інваріантна трансформація ознак (англ. Scale-invariant feature transform, SIFT) [9] – це алгоритм виявлення ознак в комп'ютерному зорі для виявлення і опису локальних ознак в зображеннях. Програми включають розпізнавання об'єктів, роботизовані складання карти і роботизовану навігацію, зшивання зображень, тривимірне моделювання, розпізнавання жестів, трекінг, ідентифікацію диких тварин і позиційний трекінг.

Спочатку в SIFT витягуються ключові точки об'єктів з набору контрольних зображень і запам'ятовуються в базі даних. Об'єкт розпізнається в новому зображенні шляхом порівнювання кожної ознаки з нового зображення з ознаками з бази даних і знаходження ознак-кандидатів на основі евклідової відстані між векторами ознак. З повного набору відповідностей в новому зображенні відбираються піднабори ключових точок, які найбільш добре узгоджуються з об'єктом по його місцю розташування, масштабу і орієнтації. Визначення відповідних блоків ознак здійснюється швидко за допомогою ефективної реалізації хеш-таблиці узагальненого перетворення Хафа. Кожен блок з трьох або більше ознак, узгоджується з об'єктом і його положенням, підлягає подальшій ретельній перевірці відповідності моделі, і блоки, що різко відхиляються, відкидаються. Нарешті, обчислюється ймовірність, що певний набір ознак свідчить про присутність об'єкта, що дає інформацію про точність збігу і серед можливих промахів. Об'єкти, які проходять всі ці тести, можуть вважатися правильними з високим ступенем впевненості.

Метод прискорених надійних функцій (Speeded Up Robust Features, SURF) [10] – це швидкий і надійний алгоритм локального, інваріантного подання подібності та порівняння зображень. Основна зацікавленість підходу SURF полягає в його швидкому обчисленні операторів, що використовують box-фільтри, таким чином даючи змогу застосувати додатки в реальному часі, такі як відстеження та розпізнавання об'єктів.

Для виявлення точок інтересу, SURF використовує ціле наближення детермінантного детектора блокування Гессіана, яке можна обчислити за допомогою 3 цілих операцій за допомогою попередньо обчисленого інтегрального зображення. Дескриптор його функцій базується на сумі хвилевідповіді Хаара навколо точки, що цікавить. Вони також можуть бути обчислені за допомогою цілісного зображення.

Дескриптори SURF використовувались для пошуку та розпізнавання об'єктів, людей чи облич, для реконструкції 3D-сцен, для відстеження об'єктів та вилучення цікавих місць.

Зображення перетворюється в координати за допомогою піраміди з багатороздільною здатністю, щоб скопіювати оригінальне зображення в пірамідальній гауссовій або лапласівській піраміді, щоб отримати зображення однакового розміру, але зі зменшеною пропускною здатністю. Це сприяє досягненню особливого ефекту розмивання на оригінальному зображенні, який називається масштаб-простір, і забезпечує те, що точки, що цікавлять, є інваріантними масштабами.

Функції тестування прискореного сегменту (Features from Accelerated Segment Test, FAST) [11] – це кутовий метод виявлення, який може бути використаний для вилучення точок функцій і пізніше використовується для відстеження та картографування об'єктів у багатьох завданнях комп'ютерного зору. Найперспективнішою перевагою кутового детектора FAST є його обчислювальна ефективність. Більше того, якщо застосовуються методи машинного навчання, можна досягти чудової продуктивності з точки зору часу на обчислення та ресурсів. Кутовий детектор FAST підходить для додатків, що обробляють відео в режимі реального часу, через його швидкісну продуктивність.

У функції FAST немає компонента орієнтації та багатомасштабних функцій. Таким чином, алгоритм кулі використовує багатомасштабну піраміду зображень. Піраміда зображень – це багатомасштабне подання єдиного зображення, яке складається з послідовностей зображень, всі з яких є версіями зображення з різною роздільною здатністю. Кожен наступний рівень у піраміді містить більш зменшену версію зображення, ніж попередній.

Бінарні надійні незалежні елементарні функції (Binary Robust Independent Elementary Features, BRIEF) [12] – це дескриптор точок загального призначення, який може поєднуватися з довільними детекторами. Це стійкий до типових класів фотометричних та геометричних перетворень зображень. BRIEF орієнтований на додатки в режимі реального часу, залишаючи їм велику частину доступної потужності процесора для подальших завдань, але також дозволяє запускати алгоритми відповідності точок функцій на обчислювально слабких пристроях, таких як мобільні телефони. BRIEF приймає всі ключові точки, знайдені за FAST алгоритмом, і перетворює їх у бінарні векторні ознаки, щоб разом вони могли представляти об'єкт. Вектор бінарних функцій також відомий як двійковий дескриптор ознак – це функціональний вектор, який містить лише 1 і 0.

Щоб запобігти чутливості дескриптора до високочастотного шуму, BRIEF починає згладжування зображення за допомогою ядра Гаусса. BRIEF вибирає випадкову пару пікселів у визначеному сусідстві навколо цієї ключової точки. Визначене сусідство навколо пікселя відоме як патч, який є квадратною шириною та висотою пікселів. Перший піксель у випадковій парі виводиться з розподілу Гаусса, зосередженого навколо ключової точки, з нитковим відхиленням або поширенням сигми. Другий піксель у випадковій парі виводиться з розподілу Гаусса, зосередженого навколо першого пікселя, зі стандартним відхиленням або розворотом сигми на два. Тепер, якщо перший піксель яскравіший за другий, він присвоює відповідному біту значення 1, інакше 0.

Орієнтований FAST та обернутий BRIEF (Oriented FAST and rotated BRIEF, ORB) [13] – це злиття детектора ключових точок FAST та дескриптора BRIEF з деякими модифікаціями. Обидва ці методи привабливі завдяки високій продуктивності, низькій складності та ресурсозатратності. Основні переваги ORB такі:

- додавання швидкої та точної орієнтаційної складової до FAST;
- ефективне обчислення орієнтованих функцій BRIEF;
- аналіз дисперсії та кореляції орієнтованих ознак BRIEF;
- метод навчання декореляції функцій BRIEF в умовах обертальної інваріантності, що призводить до кращих показників роботи в найближчих сусідніх програмах.

Спочатку, щоб визначити ключові моменти, він використовує метод FAST. Потім застосовується кутовий показник Харріса, щоб знайти топ N точок. FAST не обчислює орієнтацію і є варіантом обертання. Він обчислює середньозважену по інтенсивності центрального патчу з розташованим кутом в центрі. Напрямок вектора від цієї кутової точки до центроїда дає орієнтацію. Моменти обчислюються для поліпшення інваріантності обертання. Дескриптор BRIEF погано формується, якщо є внутрішнє обертання. В ORB матриця обертання обчислюється за допомогою орієнтації патча, а потім дескриптори BRIEF керуються відповідно до орієнтації.

Порівнюючи SIFT, SURF, FAST, BRIEF та ORB, визначено, що алгоритм ORB працює швидше, отримує більш ефективні ключові точки, ніж інші та має можливість задання очікуваної кількості функціональних детекторів, що дозволить варіювати швидкість та точність алгоритму [14]. Тому для визначення ключових точок було обрано саме алгоритм ORB.

Після визначення усіх ключових точок, наступним кроком потрібно порізати картинку таким чином, щоб збіглися схожі області. Для цього використаємо кластеризацію методом k-середніх ключових точок. Якщо картинка сильно не змінювалася в змісті і знайдені локальні ознаки залишаються майже незмінними, то центроїди після кластеризації цих локальних точок теж приблизно збігаються.

Мета методу — розділити n спостережень на k кластерів, так щоб кожне спостереження належало до кластера з найближчим до нього середнім значенням. Метод базується на мінімізації суми квадратів відстаней між кожним спостереженням та центром його кластера, тобто функції.

k-засоби кластеризації – це метод квантування векторів, спочатку від обробки сигналів, який популярний для кластерного аналізу при обробці даних. k-означає кластеризація має на меті розділити n спостережень на k кластери, в яких кожне спостереження належить кластеру з найближчим середнім значенням, слугуючи прототипом кластеру. Це призводить до розподілу простору даних на комірки Вороного [15].

Головні переваги методу k-середніх – його простота та швидкість виконання. Метод k-середніх більш зручний для кластеризації великої кількості спостережень, ніж метод ієрархічного кластерного аналізу (у якому дендограми стають перевантаженими і втрачають наочність).

Принцип алгоритму полягає в пошуку таких центрів кластерів та наборів елементів кожного кластера при наявності деякої функції, що виражає якість поточного розбиття множини на k кластерів, коли сумарне квадратичне відхилення елементів кластерів від центрів цих кластерів буде найменшим:

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2 \quad (1).$$

В початковий момент роботи алгоритму довільним чином обираються центри кластерів, далі для кожного елемента множини ітеративно обраховується відстань від центрів з приєднанням кожного елемента до кластера з найближчим центром. Для кожного з отриманих кластерів обчислюються нові значення центрів, намагаючись при цьому мінімізувати функцію, після чого повторюється процедура перерозподілу елементів між кластерами.

Алгоритм методу «Кластеризація за схемою к-середніх»:

1. Вибрати k інформаційних точок як центри кластерів поки не завершиться процес зміни центрів кластерів;
2. Зіставити кожну інформаційну точку з кластером, відстань до центра якого мінімальна;
3. Переконатися, що в кожному кластері міститься хоча б одна точка. Для цього кожний порожній кластер потрібно доповнити довільною точкою, що розташована «далеко» від центра кластера;
4. Центр кожного кластера замінити середнім від елементів кластера;
5. Кінець.

Після генерування усіх хешів, необхідно буде порівняти хеші між собою. Для порівняння використаємо відстань Геммінга. Відстань Геммінга [16] — число позицій, у яких відповідні цифри двох двійкових слів однакової довжини різні. У загальнішому випадку відстань Геммінга застосовується для рядків однакової довжини будь-яких абеток, що складаються з q символів, і служить метрикою відмінності (функцією, що визначає відстань в метричному просторі) об'єктів однакової вимірності.

Висновок

Таким чином, виявлено, що найкращим порівняльним хешем, серед усіх запропонованих варіантів, є перцептивний хеш, який генерується швидше за інші, дає більшу точність порівняння та меншу кількість помилкових спрацьовувань. Однак, для визначення модифікованого дубліката генерування одного хешу не є оптимальним варіантом, було вирішено генерувати декілька хешів – для кожної ключової області зображення. Для визначення функціональних детекторів було обрано метод ORB. Алгоритм ORB працює швидше, отримує більш ефективні ключові точки, ніж інші та має можливість задання очікуваної кількості функціональних детекторів, що дозволить варіювати швидкість та точність алгоритму. Для розділення зображення на підзображення та подальшої генерації перцептивних хешів, вирішено використати кластеризацію методом k-середніх, а подальше порівняння зображень відбувається за допомогою відстані Геммінга.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ніколайчук В.О. Новий метод пошуку модифікованих дублікатів зображень у форматі JPEG XR з використанням перцептивних хешів / В.О. Ніколайчук, О.В. Романюк – Харків, 2020. – С.107-110.
2. Ніколайчук В.О. Ущільнення зображень на основі методу JPEG XR / В.О. Ніколайчук, О.В. Романюк – Вінниця: ВНТУ, 2019.
3. Looks like it [Електронний ресурс] // Режим доступу: http://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html
4. Blocking hashing algorithm [Електронний ресурс] // Режим доступу: https://en.bitcoin.it/wiki/Block_hashing_algorithm
5. Kind of Like That [Електронний ресурс] // Режим доступу: http://www.hackerfactor.com/blog/index.php?/archives/529-Kind-of-Like-That.html
6. Zauner C. Implementation and Benchmarking of Perceptual Image Hash Functions / C.Zauner – Hagenberg, 2010
7. Wavelet image hash in Python [Електронний ресурс] // Режим доступу: https://fullstackml.com/wavelet-image-hash-in-python-3504fdd282b5
8. Testing different image hash functions [Електронний ресурс] // Режим доступу: https://content-blockchain.org/research/testing-different-image-hash-functions/
9. Scale-invariant transform feature [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/Scale-invariant_feature_transform
10. Speeded up robust features [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/Speeded_up_robust_features
11. Introduction to FAST (Features from Accelerated Segment Test) [Електронний ресурс] // Режим доступу: https://medium.com/analytics-vidhya/introduction-to-fast-features-from-accelerated-segment-test-4ed33dde6d65
12. Introduction to BRIEF (Binary Robust Independent Elementary Features) [Електронний ресурс] // Режим доступу: https://medium.com/analytics-vidhya/introduction-to-brief-binary-robust-independent-elementary-features-436f4a31a0e6
13. Introduction to ORB (Oriented FAST and Rotated BRIEF) [Електронний ресурс] // Режим доступу: https://medium.com/analytics-vidhya/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf
14. A Comparison of SIFT, SURF and ORB [Електронний ресурс] // Режим доступу: https://medium.com/@shehan.a.perera/a-comparison-of-sift-surf-and-orb-333d64bcaaea
15. k-means clustering [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/K-means_clustering
16. Hamming Distance [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/Hamming_distance

Ніколайчук Владислав Олександрович, студент групи 2ПІ-19м, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: erengard97@gmail.com

Романюк Оксана Володимирівна, доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м.Вінниця, e-mail: romaniukoksanav@gmail.com

Vladyslav Nikolaichuk, student of group 2PI-19m, Faculty for Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: erengard97@gmail.com.

Oksana Romaniuk, Associate Professor of the Software Chair, Vinnytsia National Technical University, Vinnytsia, e-mail: romaniukoksanav@gmail.com