УДК 519.688:004.92

О. І. Білобородько, Д. Д. Колчин

Дніпропетровський національний університет імені Олеся Гончара

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПОШУКУ ЗОБРАЖЕНЬ У КАТАЛОЗІ

Розроблено програмне забезпечення, яке дозволяє підтримувати каталогізацію зображень та виконувати пошук зображення за шаблоном. Реалізовано методи кодування зображень за допомогою дескриптора ключових точок SIFT та з використанням технік перцептивного хешування. Для порівняння представлень зображень використовуються K-вимірні дерева та LSH. Програмний продукт забезпечує користувача засобами для виконання пошуку різної складності.

Ключові слова: *пошук зображень за вмістом, дескриптори ключових точок, SIFT, перцептивний хеш, LSH, K-вимірні дерева, програмне забезпечення пошуку зображень.*

Разработано программное обеспечение, позволяющее каталогизировать изображения и выполнять поиск изображения по шаблону. Реализованы методы кодирования изображений с помощью дескриптора ключевых точек SIFT и с использованием техник перцептивного хэширования. Для сравнения представлений изображения используются K-мерные деревья и LSH. Программный продукт обеспечивает пользователя средствами для выполнения поиска разной сложности.

Ключевые слова: *поиск изображений по содержимому, дескрипторы ключевых точек, SIFT, перцептивный хэш, LSH, K-мерные деревья, программное обеспечение поиска изображений.*

Images cataloging software with ability to search images by template was developed. Image encoding methods using the SIFT key point descriptor and with use of perceptive hashing techniques were implemented. For image representations comparison K-dimensional trees and LSH are used. The software product provides the user with the means to perform a search of varying complexity.

Keywords: *content-based image retrieval, keypoint descriptors, SIFT, perceptual hash, LSH, K-d trees, image search software.*

Вступ. Велика кількість цифрових зображень, що супроводжують кожну людину, потребує сучасної технології для керування ними.

І пошук потрібного зображення – один з найважливіших аспектів роботи зі збереженими зображеннями – настільки ж важливий, як їх збереження, обробка та захист. Якщо користувач у ході своєї професійної діяльності не має змоги вчасно знайти потрібне йому зображення або схожі зображення, це призводить до економічних втрат.

Усі програмні пакети каталогізації та пошуку зображень можна класифікувати за принципами виконання пошуку та розташуванням сховища. Пошук зображень здійснюють на основі присвоєних тегів, на основі метаданих, за вмістом та за колірною гамою. Розташування сховища зображень (каталогу) може бути локальним або хмарним. Проте використання хмарного розташування може бути неприйнятним у разі нестабільності інтернет-зв'язку або, що більш важливо, небезпека порушення конфіденційності та авторських прав можуть стати причинами, що змушують багатьох відмовитися від онлайн систем збереження.

Аналіз літературних даних. Задача пошуку зображення в каталозі за шаблоном є актуальною та розглядається в багатьох роботах. Для кодування зображень з метою їх подальшого порівняння використовують дескриптори ключових точок та хеш-орієнтовані алгоритми. Серед дескрипторів ключових точок найчастіше використовують такі: SIFT (Scale Invariant Feature Transform) [1], SURF (Speeded Up Robust Features) [2, 3], FAST (Features from Accelerated Segment Test), BRIEF (Binary Robust Independent Elementary Features) ORB (Oriented fast and Rotated BRIEF) [4, 5].

Для побудови хеш-представлення зображення використовують такі алгоритми: aHash (Average Hashing), pHash (Perceptive Hashing), dHash (Difference Hashing), wHash (Wavelet Hashing). Кожен з цих підходів має своє коло застосувань.

Постановка задачі. Метою роботи є розробка програмного продукту каталогізації та пошуку зображень з локально розміщеною базою зображень. Програмний продукт повинен бути оптимізований для роботи з великим обсягом даних, надавати можливість оптимального вибору алгоритмів пошуку з урахуванням співвідношення швидкість/якість залежно від критеріїв пошуку. З цією метою в програмному забезпеченні необхідно реалізувати алгоритм пошуку ключових точок SIFT та побудови їх дескрипторів, алгоритми пошуку найближчих сусідів для дискрипторів на базі K-вимірних дерев та з використанням принципів LSH (Locality Sensitive Hashing), а також пошук з використанням перцептивного хеш-алгоритму на базі дискретного косинусного перетворення.

Процес пошуку зображення в каталозі складається з таких етапів:

– кодування зображень. Усі зображення в каталозі мають бути закодовані таким чином, аби представлення двох схожих зображень були близькі між собою;

– отримання на вхід зображення, що слугує пошуковим запитом;

– для кожного зображення з каталогу має бути розраховано коефіцієнт збіжності з цільовим зображенням;

– результати пошуку надавати користувачу в порядку їх ранжування за коефіцієнтом збіжності.

Для зменшення часу пошуку кожне зображення в каталозі має бути закодовано лише один раз. Після цього результат кодування зберігається в базі разом із зображенням.

Основний матеріал. Результатом роботи є програмний продукт «ImageSearch», розроблений мовою C#, діаграму варіантів використання (ВВ) якого наведено на рис 1.

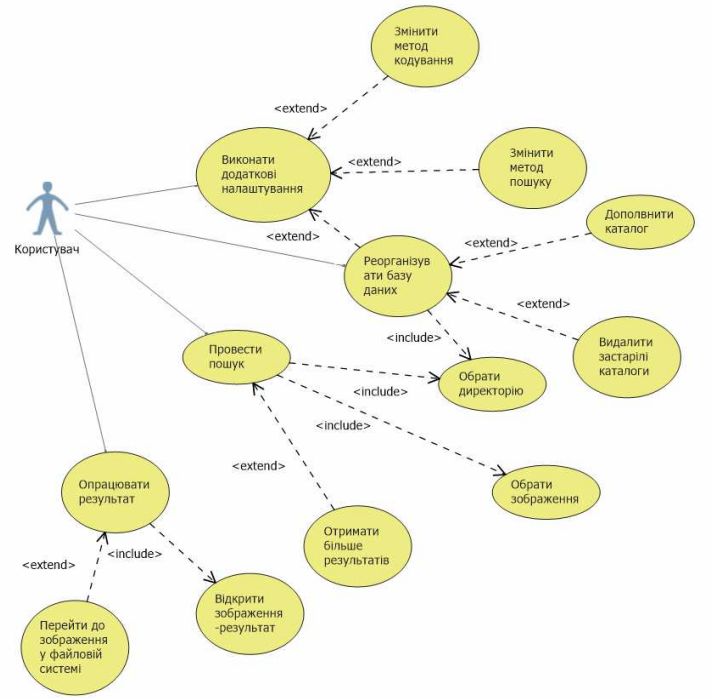

Рисунок 1 – Діаграма ВВ програмного продукту «ImageSearch»

Розглянемо найважливіші варіанти використання більш докладно.

«Обрати директорію» – користувач задає каталог для пошуку зображення, вказуючи локальну директорію.

«Обрати зображення» – користувач вказує зображення на диску, яке буде використано в якості пошукового шаблону.

«Виконати додаткові налаштування» – користувач відкриває панель додаткових налаштувань і конфігурує параметри роботи програми.

«Змінити метод кодування» – цей блок налаштувань дозволяє змінити метод кодування зображення. Користувачеві надається два методи на вибір: дескриптор SIFT або перцептивний хеш.

«Змінити метод пошуку» – цей блок налаштувань дозволяє користувачу змінити метод пошуку найближчих сусідів для дескриптора SIFT. Блок надає можливість обирати між використанням більш точного методу, який використовує K-вимірні дерева, та більш швидкого методу на основі LSH.

«Реорганізувати базу даних» – цей блок налаштувань дозволяє користувачу виконувати дві операції:

– ***«Доповнити каталог»*** – дозволяє додати до індексованого каталогу в базі даних нові зображення;

– ***«Видалити застарілі каталоги»*** – очищує базу від каталогів, що були видалені з диска.

«Провести пошук» – користувач підтверджує, що він зробив усі необхідні налаштування, задав каталог та пошуковий шаблон, і програма виконує пошук.

«Отримати більше результатів» – після проведення пошуку користувач отримує десять найбільш схожих зображень. Якщо він хоче побачити більше – він має натиснути відповідну кнопку.

«Опрацювати результат» – після проведення пошуку користувач може опрацювати результати одним із наведених нижче способів:

– ***«Відкрити зображення»*** – відкриває одне із зображень-результатів програмою, яка асоційована із даним типом файлів на комп’ютері користувача;

– ***«Перейти до зображення»*** – користувач може одразу перейти до папки, у якій знаходиться обране зображення.

Програмне забезпечення спроектовано та реалізовано з використанням об’єктно-орієнтованого підходу. Діаграму класів наведено на рис. 2. Реалізацію класів розміщено в таких модулях:

UserInterface – містить класи для взаємодії з користувачем.

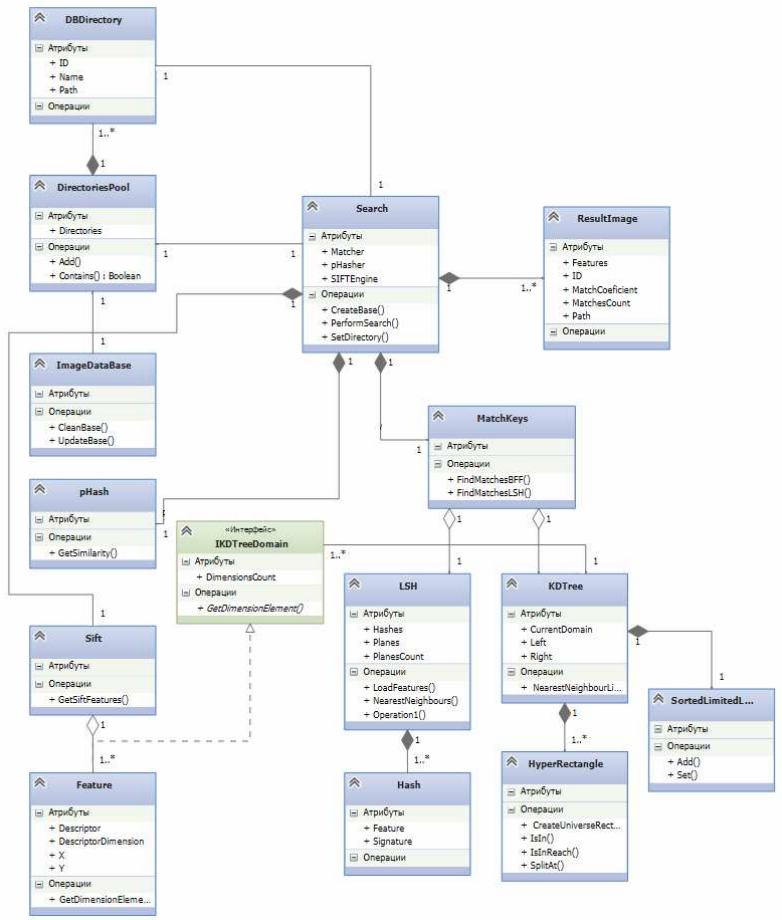

Рисунок 2 – Діаграма класів програмного продукту «ImageSearch»

Infrastructure – містить типи та інтерфейси, потрібні для взаємодії між модулями.

Matcher – містить класи, які виконують пошук найближчих сусідів поміж дескрипторів, та надає функціонал порівняння ключових точок.

PerceptiveHashing – містить логіку створення перцептивного хешу на основі поданого зображення та спеціальний порівняльний функціонал.

SIFT – містить службовий клас Sift, що обчислює ключові точки зображення та створює відповідні дескриптори.

SearchEngine – приймає запити від інтерфейсу користувача, створює пошукові ресурси, керує процесом пошуку та порівняння. Іншими словами, цей модуль об'єднує і керує іншими модулями, виконуючи основний функціонал програми.

Розглянемо фактичні результати роботи алгоритмів та проведемо їх аналіз.

На рис. 3 наведено, які результати знайдено за алгоритмом SIFT, незважаючи на те, що вони відрізняються масштабом, освітленням, чи навіть якщо над ними виконані афінні перетворення. На рис. 4 видно, що далі у списку результатів є навіть фото того самого об'єкта з іншого ракурсу.

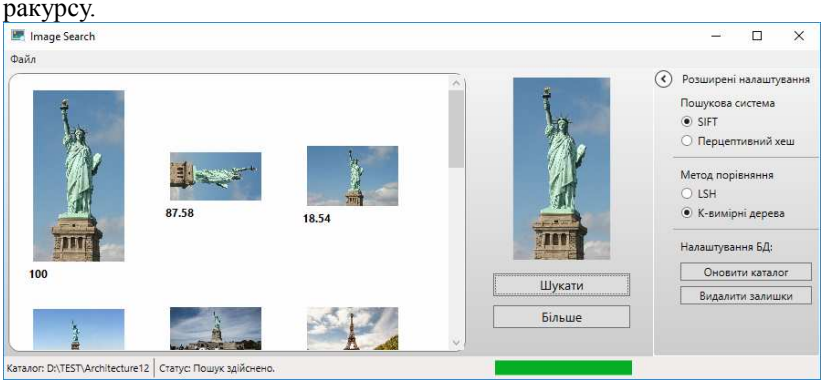

Рисунок 3 – Результати роботи методу SIFT

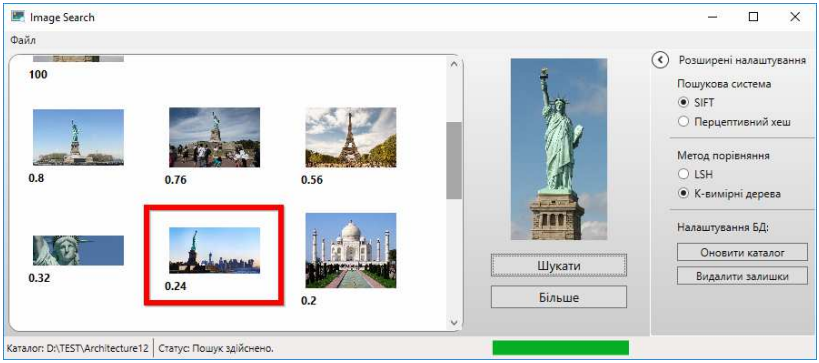

Рисунок 4 – Зображення-інваріант до положення камери

Наведені на рис. 3 та 4 результати отримано з використанням методу знаходження найближчих сусідів за допомогою K-вимірних дерев. При використанні методу LSH зображення пам'ятника з іншого ракурсу не було включено до перших десяти результатів.

Із пошуком на наборі зображень, які є абстрактними сценами (не мають у собі конкретних об'єктів), алгоритм SIFT не впорався (рис. 5). Кодування однієї абстрактної сцени займає близько двох хвилин. Якщо для звичайного зображення високої деталізації (фото Ейфелевої вежі тощо) отримується приблизно 2–3 тисячі ключових точок, то для фотографії бору дескриптор знаходить більше 30-ти тисяч ключових точок.

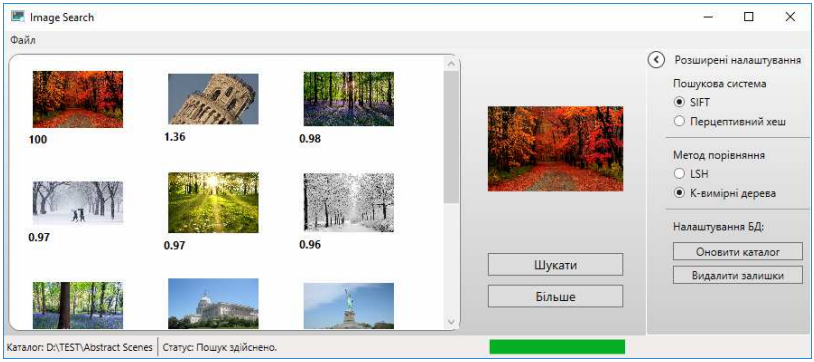

Рисунок 5 – Пошук абстрактної сцени за допомогою дескриптора SIFT

На відміну від алгоритму SIFT, алгоритм перцептивного хешування на базі дискретного косинусного перетворення для зображень, які є абстрактними сценами, показує кращі результати (рис.6).

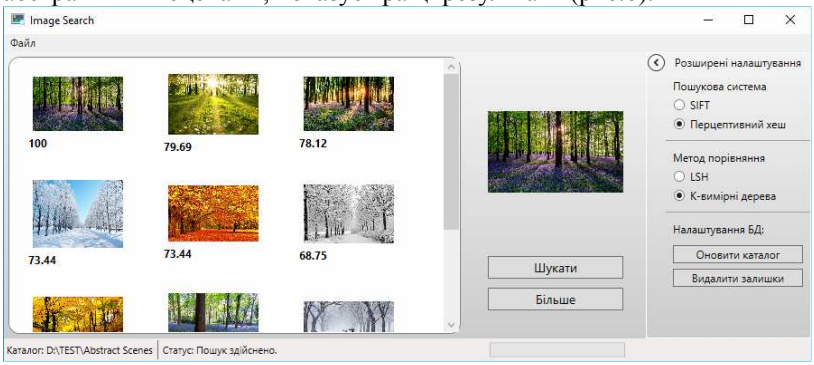

Рисунок 6 – Пошук абстрактної сцени за допомогою перцептивного хешування

На рис. 7 наведено результати пошуку на даних, що є сукупністю об'єктів (наприклад, на тих самих даних, на яких ми випробовували SIFT), при застосуванні перцептивного хешування.

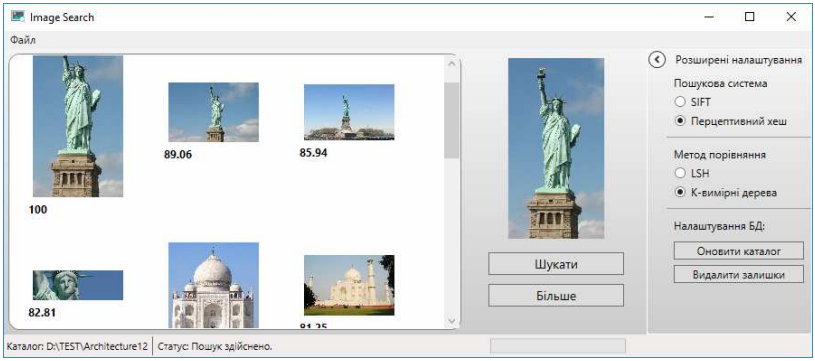

Рисунок 7 – Пошук методом перцептивного хешування на даних із сукупністю об'єктів

Як видно з результатів, перцептивний хеш досить непогано впорався з розпізнаванням, але він пропустив результати, на яких було проведено зміни освітлення або афінні перетворення.

Таким чином, для пошуку зображень, що відповідають пошуковому шаблону та можуть містити зміни освітлення, афінні перетворення, зміну масштабу та положення камери, доцільно використовувати метод SIFT.

Головною перевагою підходу на базі перцептивного хешування є можливість шукати схожі сцени (такий високий рівень абстракції не може бути прийнятно оброблений дескрипторами ключових точок) та його швидкість. Сценою назвемо випадок, коли зображення трактується як одне ціле, як один об'єкт.

Також суттєвим параметром при обранні алгоритму пошуку є час.

При використанні SIFT дескриптор застосовується двічі за час роботи програми. Вперше – при індексації каталогу. На кожному зображенні знаходяться ключові точки, та їх дескриптори записуються в базу. Вдруге – коли виконується пошук. Знаходяться ознаки цільового зображення, а потім порівнюються з ознаками зображень у базі.

Час, потрібний на проведення пошуку ознак та створення дескриптора для зображення середнього рівня деталізації – 20 секунд. Зображення різного рівня деталізації наведено на рис. 8. Зліва направо: високий рівень деталізації, середній, низький.

Пошук найближчих сусідів з використанням K-вимірних дерев займає приблизно 5 секунд для зображення середньої деталізації. LSH проводить порівняння за 2 секунди.

В табл. 1 наведено порівняння швидкості обчислення подібності з використанням K-вимірних дерев та LSH для зображень різного рівня

деталізації.

Рисунок 8 – Зображення різного рівня деталізації

Таблиця 1 – Порівняння швидкості обчислення подібності для SIFT

Деталізація	Алгоритм	
	К-вимірні дерева	LSH
Низька	2 с.	0,27 с.
Середня	5 с.	2 с.
Висока	15 с.	4 с.

Підхід порівняння дескрипторів ключових точок з використанням К-вимірних дерев дає високу точність, але характеризується більшим обчислювальним часом. Підхід з використанням LSH заснований на випадковому виборі планів, а отже, працює із деякою похибкою. Чим більше в базі зображень приблизно одного ступеня подібності, тим більшою є похибка. LSH працює краще на зображеннях високої деталізації, бо він створений спеціально для цього.

В табл. 2 наведено час обчислення перцептивного хешу для зображень різного рівня деталізації.

Таблиця 2 – Швидкість обчислення хешів для зображень різного рівня деталізації

Деталізація	Час обчислення
Низька	0,008 с.
Середня	0,02 с.
Висока	0,048 с.

Використання паралельних потоків дозволяє оброблювати десятки зображень водночас. Для використання перцептивних хешів у якості кодуючого методу немає потреби створювати базу даних. Пошук навіть у неіндексованому каталозі займає секунди.

Висновки. Розроблено програмний продукт, який дозволяє підтримувати каталогізацію зображень з локально розміщеною базою зображень та виконувати в них пошук зображення за шаблоном. У програмному забезпеченні реалізовано методи кодування зображень за допомогою дескриптора ключових точок SIFT та з використанням технік перцептивного хешування. Користувачеві також доступно два методи порівняння представлень зображень: K-вимірні дерева та LSH.

Реалізовані методи та підходи доповнюють один одного. За часом роботи оптимальним є підхід, заснований на перцептивному хешуванні, за точністю більш оптимальним є використання дескриптора ключових точок. Для дескриптора також представлені більш швидка модифікація (LSH) та більш точна (K-вимірні дерева).

Реалізований набір алгоритмів має на меті задовольнити користувачів із найрізноманітнішими потребами й можливостями. Різноманітні поєднання зазначених підходів дійсно дозволяють досягти цієї мети.

Бібліографічні посилання

1. Lowe D. G. Distinctive Image Features from Scale-Invariant Keypoints // Int'l J: Computer Vision. 2004. Vol. 60. No. 2. P. 91–110.

2. Luo J., Oubong G. A comparison of SIFT, PCA-SIFT and SURF // International Journal of Image Processing, 2009. Vol. 3. No. 4. P. 143–152.

3. Bay H., Tuytelaars T., Gool L.V. SURF: Speeded Up Robust Features // Proceedings of the ninth European Conference on Computer Vision. 2006. P. 404–417.

4. Calonder M., Lepetit V., Strecha C., Fua P. BRIEF: Binary Robust Independent Elementary Features // 11th European Conference on Computer Vision (ECCV). 2010. Part IV. P. 778–792.

5. Rublee E., Rabaud V., Konolige K., Bradski G. ORB: an efficient alternative to SIFT or SURF // ICCV. 2011: P. 2564–2571.

Надійшла до редколегії 26.06.17