

TRƯỜNG ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



**BÁO CÁO CUỐI KỲ
MÔN NHẬN DẠNG THỊ GIÁC VÀ ỨNG DỤNG
TRUY VĂN ẢNH ÁP DỤNG MÔ HÌNH GIỎ ĐẶC
TRƯNG**

**GVHD: TS. LÊ ĐÌNH DUY
TS. NGUYỄN TẤN TRẦN MINH KHANG
HV: NGUYỄN THANH PHƯƠNG
MSHV: CH1601042
LỚP: CAO HỌC KHÓA 11 ĐỢT 2**

TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2017

LỜI CẢM ƠN

Đồ án cuối môn học Nhận dạng thị giác và ứng dụng được thực hiện dưới sự hướng dẫn của **TS. Lê Đình Duy, TS. Nguyễn Trần Tân Minh Khang** và đồng thời được kê thừa từ Môn Tìm kiếm thông tin thị giác - **TS. Ngô Đức Thành**. Học viên bày tỏ lòng biết ơn sâu sắc đến các Thầy - người đã truyền cảm hứng, hướng dẫn tận tình, cung cấp những kiến thức căn bản và thiết yếu, những bài học, những góp ý quý báu trong học tập và định hướng nghiên cứu.

Học viên xin được cảm ơn đến các nhà khoa học, các tác giả các công trình công bố đã được trích dẫn trong Báo cáo, đây là những tư liệu có hàm lượng kiến thức liên quan quan trọng giúp học viên hoàn thành báo cáo này.

TP.Hồ Chí Minh, ngày 27 tháng 12 năm 2017

Học viên
Nguyễn Thanh Phương

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN VỀ TRA CỨU ẢNH DỰA TRÊN NỘI DUNG	7
1.1. Giới thiệu truy vấn ảnh dựa trên nội dung	7
1.1.1. Mô hình xử lý	8
1.1.2. Các thành phần chính của một hệ thống CBIR	8
1.1.3. Các chức năng cơ bản của hệ thống CBIR	8
1.1.4. Một số hệ thống tra cứu ảnh dựa trên nội dung	9
1.1.5. Mô hình hệ thống truy vấn ảnh	9
1.2. Thuật toán SIFT (Scale Invariant Feature Transform)	9
1.3. Thuật toán SURF	9
1.4. Thuật toán ORB	10
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT THUẬT TOÁN	12
2.1. Thuật toán SIFT	12
2.1.1. Giới thiệu	12
2.1.2. Các nghiên cứu liên quan	13
2.1.3. Phát hiện cực trị trong không gian tỉ lệ	14
2.1.3.1. Phát hiện cực trị địa phương	16
2.1.3.2. Tần suất lấy mẫu tỉ lệ	17
2.1.3.3. Tần suất lấy mẫu trong miền không gian	18
2.1.4. Định vị chính xác Keypoint	19
2.1.4.1. Loại bỏ phản ứng biên	21
2.1.5. Gán hướng	22
2.1.6. Bộ mô tả hình ảnh cục bộ	23
2.1.6.1. Bộ mô tả	24
2.1.6.2. Kiểm thử Descriptor	24
2.1.6.3. Độ nhạy với biến đổi Affine	25
2.1.6.4. Kết hợp với cơ sở dữ liệu lớn	26
2.1.7. Ứng dụng cho nhận dạng đối tượng	27
2.1.7.1. Khớp Keypoint	27
2.1.7.2. Hiệu quả của việc đánh số các điểm láng giềng gần	28
2.1.7.3. Cụm biến đổi Hough	29

2.1.7.4. Giải pháp cho các thông số Affine	30
2.2. Thuật toán SURF (Speeded Up Robust Features).....	31
2.2.1. Giới thiệu thuật toán SURF.....	31
2.2.2. Nghiên cứu liên quan	32
2.2.3. Bộ dò Fast- Hessian	33
2.2.4. Bộ mô tả SURF	36
2.2.4.1. Gán hướng	36
2.2.4.2. Thành phần của mô tả.....	36
2.3. Thuật toán ORB	38
2.3.1. Nghiên cứu liên quan	39
2.3.2. oFAST: hướng của keypoint FAST	40
2.3.2.1. Bộ dò FAST	40
2.3.2.2. Hướng của cường độ trọng tâm	40
2.3.3. Phép quay: rBRIEF	41
2.3.3.1. Hiệu quả của phép quay BRIEF	41
2.3.4. Biến thể và tương quan	43
2.3.5. Phương pháp học cho các thuộc tính nhị phân.....	44
2.3.6. Đánh giá	45
2.3.7. Ước lượng các đối sánh thuộc tính nhị phân.....	47
2.3.7.1. Hàm băm cục bộ tốt cho rBRIEF	47
2.3.7.2. Tương quan và Cân bằng.....	47
CHƯƠNG 3. MÔ TẢ TRUY VÂN ẢNH SỬ DỤNG MÔ HÌNH GIỎ ĐẶC TRƯNG 49	
3.1. Mô tả tập dữ liệu Oxford Building	49
3.2. Kết quả thực nghiệm.....	52
3.3. Mô hình truy vấn ảnh BoW-SVM.....54	
3.4. Mô hình truy vấn ảnh BoW-IF.....55	
3.4.1. Giai đoạn huấn luyện	56
3.4.1.1. Rút trích đặc trưng.	56
3.4.1.2. Xây dựng tập từ điển ảnh.....	56
3.4.1.3. Lượng từ hóa đặc trưng của các ảnh huấn luyện.	57
3.4.1.4. Xây dựng cấu trúc chỉ mục IF.....	57
3.4.2. Giai đoạn truy vấn ảnh.....	57

3.4.2.1. Tiết xử lý ảnh truy vấn	57
3.4.2.2. Truy vấn ảnh	58
a) Đặc trưng SIFT	58
b) Đặc trưng ROOTSIFT	59
c) Thuật toán APPROXIMATE K-MEANS (AKM)	61
d) Cấu trúc chỉ mục IF	61
3.5. Cấu trúc chỉ mục thông tin không gian phân cấp.....	65
3.6. Kết hợp mô hình BoVW và CNNs	68
3.6.1. Rút trích đặc trưng sử dụng CNNs	68
3.6.2. Kết hợp mô hình BoVW và CNNs	69
3.6.3. Giai đoạn tiền xử lý	71
3.6.4. Cấu hình CNNs.....	71
3.6.5. Phân tích chọn lọc các đặc trưng cuộn xoắn.....	72
3.6.6. Cài đặt thông số	73
3.6.7. Kết luận	73
3.6.8. Hướng nghiên cứu mở rộng.....	73
3.7. Các cách tính độ đo tương đồng	77
3.8. Các độ đo đánh giá hiệu suất	77
CHƯƠNG 4. CÀI ĐẶT CHƯƠNG TRÌNH TRUY VẤN ẢNH THỬ NGHIỆM.....	80
4.1. Thử nghiệm 1	80
4.1.1. Sử dụng bộ datasets Oxford Building như mô tả ở Chương 3	80
4.1.2. Sử dụng bộ datasets flowers.....	90
4.2. Thử nghiệm 2	96
TÀI LIỆU THAM KHẢO.....	98

LỜI MỞ ĐẦU

Trong vấn đề truy vấn dữ liệu, đặc biệt là dữ liệu ảnh, bài toán tìm kiếm ảnh tương tự là một bài toán quan trọng, phù hợp với nhu cầu xã hội hiện đại. Bài toán tìm kiếm ảnh được phân làm 2 lớp chính:

- Tìm kiếm ảnh dựa trên văn bản TBIR.
- Tìm kiếm ảnh dựa trên nội dung CBIR.

Phương pháp CBIR thực hiện tìm kiếm dựa trên đặc trưng thị giác của hình ảnh nên hiệu quả hơn phương pháp TBIR. Thách thức của CBIR: trích xuất tự động các đặc trưng thị giác, tạo ra các chỉ mục đa chiều, đưa ra phương pháp tìm kiếm ảnh tương tự. Do đó, phương pháp tìm kiếm ảnh theo nội dung là sự kết hợp của các lĩnh vực: xử lý ảnh, thị giác máy tính, truy hồi thông tin,...

Mục tiêu của đồ án cuối môn học này của học viên là tìm hiểu các phương pháp dùng BoW.

Nội dung và bô cục của đồ án gồm 4 chương:

- Chương 1: Tổng quan về tra cứu ảnh dựa trên nội dung.
- Chương 2: Cơ sở lý thuyết thuật toán.
- Chương 3: Truy vấn ảnh sử dụng mô hình giỏ đặc trưng. Chương 3 là trọng tâm của đồ án, giới thiệu mô hình truy vấn ảnh **BoW-SVM**, **BoW-IF**, **BoW-SPIF**, **BoW-CNNs**.
- Chương 4: Cài đặt chương trình truy vấn ảnh thử nghiệm.

Địa chỉ lưu source code báo cáo:

(1).Địa chỉ email:

phuongnt.11@grad.uit.edu.vn

nguyenthanhphuongdaisy@gmail.com

(2).Địa chỉ github lưu source code và các báo cáo:

<https://github.com/NguyenThanhPhuongdaisy/VRA1>

(3).Địa chỉ YouTube link đến video minh họa cho việc cài đặt, chạy chương trình và hiển thị kết quả: <https://youtu.be/jEKOkKeDhp4>

CHƯƠNG 1. TỔNG QUAN VỀ TRUY CỨU ẢNH DỰA TRÊN NỘI DUNG

1.1. Giới thiệu truy vấn ảnh dựa trên nội dung

Tra cứu ảnh dựa vào nội dung – Content based image retrieval (CBIR) được giới thiệu bởi các nhà nghiên cứu năm 1980. CBIR được nghiên cứu rộng rãi, nhiều phương pháp và hệ thống đã được phát triển như QBIC, Photobook, MARS, PicHunter, VisualSEEK,... để rút trích nội dung của ảnh bằng cách sử dụng các đặc trưng mức thấp bên trong mỗi ảnh như màu sắc, hình dạng, kết cấu,... Các hệ thống này thường trích rút các biểu diễn trực quan của ảnh và định nghĩa các hàm tìm kiếm, đối sánh mối liên quan khi tra cứu nhằm đáp ứng yêu cầu người dùng. Trước khi rút trích đặc trưng, các ảnh được tiền xử lý: chuyển đổi không gian màu, giảm nhiễu, lượng tử hóa,... Việc lựa chọn đặc trưng tùy thuộc vào cơ sở dữ liệu ảnh, ứng dụng cũng như mong muốn người dùng. Các đặc trưng có thể biểu diễn toàn cục hoặc cục bộ hoặc các tiếp cận không gian.

Để cải thiện hiệu quả tra cứu, các kỹ thuật học máy (SVM, boosting,...) được sử dụng trong phân lớp theo truy vấn. Hạn chế là dữ liệu huấn luyện tương đối ít và dữ liệu kiểm tra bị nhiễu do vấn đề khoảng trống ngữ nghĩa. Mục đích sử dụng các kỹ thuật học máy để thực hiện nhiệm vụ xếp hạng. Đây là nhiệm vụ học giám sát do có các pha huấn luyện và kiểm tra. Dữ liệu bao gồm: bộ datasets và ảnh truy vấn.

Có 3 vấn đề quan trọng trong truy vấn ảnh dựa vào nội dung là: phương pháp trích đặc trưng trên ảnh, cách thức tổ chức cơ sở dữ liệu và phương pháp xử lý truy vấn. Bốn cách thức truy vấn ảnh được sắp xếp từ thấp đến cao:

- (1). Đặc trưng thị giác cấp thấp: được xác định tại mỗi điểm ảnh hoặc toàn ảnh.
- (2). Đặc trưng thị giác cấp cao: được xác định cho vùng ảnh.
- (3). Khái niệm thị giác: khái quát hóa dạng đặc trưng thị giác dưới dạng tên gọi gần với cách nghĩ của con người.
- (4). Ngữ nghĩa: thể hiện sự khái hóa đối tượng đang hiện hữu.

Đặc trưng thị giác là thông tin dạng số của màu sắc, vân, hình dáng.

Hệ thống truy vấn ảnh dựa vào ngữ nghĩa thường gặp phải vấn đề về rút trích và nhận dạng đặc trưng ngữ nghĩa trong ảnh và độ tương đồng tổng thể giữa các đặc trưng ngữ nghĩa với nhau. Phương pháp truy vấn ảnh dựa trên đặc trưng cơ bản thông qua các hệ thống CBIR. Mỗi cách lựa chọn đặc trưng cơ bản để truy vấn thường cũng chỉ tối ưu trong một số loại ảnh trong những trường hợp nhất định. Hiện nay, việc đánh chỉ mục cho cơ sở dữ liệu ảnh có hạn

chế về chi phí tính toán. Bên cạnh đó, kích thước dữ liệu ảnh lớn cũng là một thách thức.

1.1.1. Mô hình xử lý

Để xây dựng hệ thống truy vấn ảnh, cần giải quyết 3 vấn đề chính sau:

- Rút trích các đặc trưng trên ảnh (Feature Extraction).
- Xác định độ đo sự tương đồng giữa hai ảnh (Similarity Measure).
- Lập chỉ mục cho Cơ sở dữ liệu (CSDL) ảnh (Image Indexing). Từ đó xây dựng nên các thành phần cho hệ thống.

1.1.2. Các thành phần chính của một hệ thống CBIR

Hiện nay, trọng tâm chính của CBIR là nghiên cứu chủ yếu trên 3 chủ đề chính:

Trích chọn đặc trưng: các đặc trưng của hình ảnh bao gồm các đặc trưng nguyên thủy và các đặc trưng ngữ nghĩa/đặc trưng logic.

- Đặc trưng nguyên thủy như màu sắc, hình dạng, kết cấu và các mối quan hệ không gian được định lượng trong tự nhiên, chúng có thể được trích xuất tự động hoặc bán tự động.
- Đặc trưng logic cung cấp mô tả trừu tượng của dữ liệu hình ảnh ở các cấp độ khác nhau. Lập chỉ số hiệu quả: Để tạo điều kiện truy vấn hiệu quả và xử lý tìm kiếm, các chỉ số hình ảnh cần thiết được tổ chức thành các cấu trúc dữ liệu hiệu quả. Các cấu trúc như KD-tree, R-tree family, R *- tree, quad-tree, và grid file (tập lưới) thường được sử dụng.

Giao diện người dùng: giao diện người dùng bao gồm một bộ xử lý truy vấn và trình duyệt để cung cấp các công cụ đồ họa tương tác, cơ chế truy vấn và truy cập cơ sở dữ liệu, theo thứ tự định sẵn.

1.1.3. Các chức năng cơ bản của hệ thống CBIR

Những chức năng chính của một hệ thống bao gồm các nội dung sau:

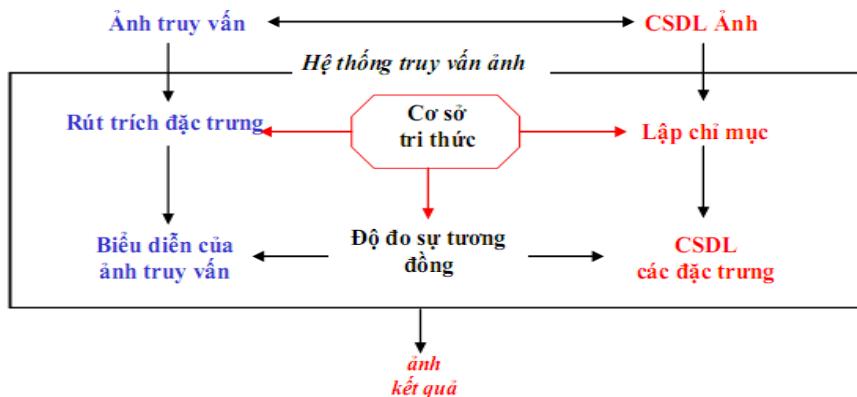
- Phân tích nội dung của nguồn thông tin và biểu diễn nội dung của các nguồn thông tin.
- Phân tích các truy vấn của người dùng và biểu diễn chúng thành các dạng phù hợp với việc đối sánh với cơ sở dữ liệu nguồn.
- Xác định chiến lược để đối sánh tìm kiếm truy vấn với thông tin được lưu trữ trong cơ sở dữ liệu.
- Thực hiện các điều chỉnh cần thiết trong hệ thống dựa trên phản hồi từ người sử

dụng hoặc những hình ảnh được tra cứu.

1.1.4. Một số hệ thống tra cứu ảnh dựa trên nội dung

- Hệ thống QBIC (Query By Image Content).
- Hệ thống Virage.
- Hệ thống RetrievalWare.
- Hệ thống VisualSeek và WebSeek.
- Hệ thống Photobook.

1.1.5. Mô hình hệ thống truy vấn ảnh



Bài toán đối sánh ảnh là bài toán con của bài toán truy vấn ảnh. Đối sánh ảnh là đối sánh các đặc trưng bất biến tỉ lệ của 2 bức ảnh. Một số phương pháp đối sánh ảnh dựa trên các đặc điểm bất biến dùng các thuật toán phổ biến SIFT, SURF, ORB.

1.2. Thuật toán SIFT (Scale Invariant Feature Transform)

SIFT là một phương pháp để chiết xuất các thuộc tính bất biến đặc biệt từ các hình ảnh và được sử dụng để thực hiện đối sánh tin cậy giữa các khung nhìn khác nhau của một đối tượng hay cảnh. Các thuộc tính này là bất biến đối với phép thay đổi tỉ lệ và phép quay ảnh và thể hiện rõ nét trong việc đối sánh một vùng con với phép biến đổi affine và sự thay đổi khung nhìn 3D cộng thêm nhiều và thay đổi trong chiều sáng. Các thuộc tính này rất đặc biệt và là một thuộc tính duy nhất có thể đối sánh chính xác trong một cơ sở dữ liệu lớn các thuộc tính chiết xuất từ nhiều hình ảnh. Ngoài ra thuật toán này cũng được ứng dụng trong cách tiếp cận để nhận dạng đối tượng.

1.3. Thuật toán SURF

SURF là bộ phát hiện và bộ mô tả các điểm quan trọng bất biến với tỷ lệ và góc xoay. Phương pháp này tương đương hoặc thậm chí nhanh hơn so với các phương pháp để xuất trước đây mà liên quan đến tính lặp đi lặp lại, tính riêng biệt và tính vững chắc, nó còn giúp

việc tính toán và so sánh nhanh hơn.

SURF đạt được kết quả này bằng cách dựa trên những hình ảnh tích hợp có nhiều nếp cuộn hình ảnh thông qua việc xây dựng dựa trên các thế mạnh của các bộ phát hiện và bộ mô tả hàng đầu (ở đây sử dụng phương pháp ma trận của Hessian để đo đạc cho bộ phát hiện và dựa trên phương pháp phân phối cho các bộ mô tả). Bằng cách đơn giản hóa các phương pháp này sẽ cho ta các kết quả thiết yếu và dẫn tới việc liên kết các phát hiện và mô tả mới phù hợp. Báo cáo kết quả thực nghiệm dựa trên các đánh giá tiêu chuẩn cũng như dựa trên các hình ảnh thu được trong phạm vi của các ứng dụng nhận dạng đối tượng trong thế giới thực. Cả hai cho thấy hiệu suất mạnh mẽ của SURF.

Việc tìm sự tương đồng giữa 2 bức ảnh trong cùng một khung cảnh hoặc cùng một đối tượng là một phần trong rất nhiều phần của các ứng dụng về thị giác máy tính. Hiệu chỉnh máy ảnh, tái cấu trúc 3D, đăng ký ảnh và nhận dạng ảnh là một vài ví dụ điển hình. Việc tìm kiếm sự tương đồng riêng biệt giữa các bức ảnh là mục đích của luận văn này. Thuật toán SURF có thể được chia thành 3 bước chính như sau: Đầu tiên, các điểm quan tâm được lựa chọn tại những vị trí đặc trưng trên bức ảnh như tại các góc, những đốm màu, các ngã 3. Điểm đáng chú ý nhất của những điểm quan tâm là tính lặp đi lặp lại, tức là việc tìm kiếm các điểm quan tâm dưới nhiều khung nhìn khác nhau là đáng tin cậy. Vùng phụ cận của các điểm quan tâm được miêu tả bằng các vector đặc tính. Bộ mô tả phải có sự riêng biệt, cùng một thời điểm, có dữ liệu thừa, sai số phát hiện, biến dạng hình học và trắc quang. Cuối cùng, các vector mô tả được hòa hợp giữa các bức ảnh. Sự hòa hợp thông thường dựa trên khoảng cách giữa các vector, chẳng hạn khoảng cách Mahalanobis hoặc Euclidean. Kích thước của bộ mô tả có sự tác động trực tiếp tới thời gian thực hiện quá trình, và do vậy kích thước của bộ mô tả nhỏ là điều chúng ta mong muốn.

1.4. Thuật toán ORB

Đối sánh thuộc tính là cơ sở của nhiều vấn đề thị giác máy tính, chẳng hạn như nhận dạng đối tượng hoặc cấu trúc từ sự chuyển động. Phương pháp hiện nay dựa trên việc mô tả gây tổn kém về nhận dạng và đối sánh. Trong báo cáo này ta nghiên cứu một mô tả nhị phân dựa trên BRIEF gọi là ORB, đó là vòng xoay bất biến và có khả năng chống nhiễu. Các thí nghiệm đã chứng minh rằng ORB đứng ở vị trí thứ hai về độ lớn và nhanh hơn SIFT trong khi nó hoạt tốt trong nhiều tình huống. Hiệu quả được thử nghiệm trên một số ứng dụng thế giới thực, bao gồm phát hiện đối tượng và theo dõi trên điện thoại thông minh.

Các bộ dò keypoint và mô tả SIFT đã được chứng minh hiệu quả trong một số ứng dụng sử dụng các thuộc tính trực quan, bao gồm nhận dạng đối tượng, tách biên ảnh, ánh xạ trực

quan, vv... Tuy nhiên, nó cũng gây gánh nặng cho việc tính toán, đặc biệt là cho các hệ thống thời gian thực như xác định vị trí không gian và hướng di chuyển cho rô-bốt người máy hoặc cho các thiết bị năng lượng thấp như điện thoại di động. Điều này đã dẫn đến một cuộc tìm kiếm sâu hơn để thay thế các thuật toán với chi phí tính toán thấp hơn, thuật toán tốt nhất trong số thuật toán này là SURF. Hiện SURF cũng đã được nghiên cứu nhằm mục đích đẩy mạnh khả năng tính toán của SIFT.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT THUẬT TOÁN

2.1. Thuật toán SIFT

2.1.1. Giới thiệu

Bài toán tìm kiếm một hình ảnh phù hợp là một khía cạnh cơ bản của nhiều vấn đề trong thị giác máy tính bao gồm cả nhận dạng đối tượng hay cảnh và xử lý các cấu trúc 3D từ nhiều hình ảnh, âm thanh và theo dõi chuyển động. Trong một hình ảnh thì việc mô tả các thuộc tính mà làm cho chúng được nhận dạng trong các hình ảnh khác nhau của một đối tượng hay cảnh ở các khung nhìn khác nhau là vô cùng quan trọng. Các thuộc tính này là bất biến khi ta co giãn ảnh và xoay ảnh và một phần bất biến khi ta thay đổi trong chiếu sáng và hướng nhìn 3D của camera đa điểm. Chúng được định vị hóa tốt trong cả hai lĩnh vực không gian và miền tần số, giảm sự ảnh hưởng của sự lộn xộn trong hình ảnh hoặc nhiễu. Một số lượng lớn các thuộc tính có thể được chiết xuất từ các hình ảnh tiêu biểu với các thuật toán hiệu quả. Ngoài ra, các thuộc tính này là rất đặc biệt, trong đó cho phép một thuộc tính duy nhất có xác suất truy vấn cao đối với các thuộc tính trong một cơ sở dữ liệu lớn các thuộc tính và cung cấp một cơ sở cho nhận dạng đối tượng và bối cảnh.

Không gian tỉ lệ phát hiện cực trị: Giai đoạn đầu tiên của tìm kiếm được tính trên tất cả các tỉ lệ và vị trí hình ảnh. Nó được thực hiện hiệu quả bằng cách sử dụng hàm DoG (Difference-of-Gaussian) để xác định các điểm quan tâm tiềm năng, đó là những điểm bất biến với các tỉ lệ và hướng.

Cục bộ hóa các Keypoint: Tại mỗi điểm ứng viên địa phương sẽ có một mô hình chi tiết phù hợp để xác định vị trí và tỉ lệ. Keypoint được lựa chọn dựa trên sự ổn định của chúng trong các phép đo.

Gán hướng: Một hoặc nhiều hướng được gán cho mỗi keypoint cục bộ dựa trên hướng gradient hình ảnh cục bộ. Tất cả các hoạt động trong tương lai được thực hiện trên dữ liệu hình ảnh đó đã được chuyển đổi liên quan đến phép gán hướng và tỉ lệ địa phương hóa cho mỗi thuộc tính, qua đó cung cấp các phép biến đổi bất biến

Bộ mô tả keypoint: Các gradient hình ảnh địa phương được chọn lựa trong các vùng xung quanh keypoint. Chúng được chuyển đổi thành đại diện địa phương quan trọng khi làm méo hình dạng và thay đổi trong chiếu sáng. Cách tiếp cận này được đặt tên là các đặc trưng bất biến tỉ lệ (SIFT) vì nó biến đổi dữ liệu hình ảnh vào hệ tọa độ bất biến tỉ lệ liên quan đến các thuộc tính địa phương.

Với đối sánh ảnh và nhận dạng, các thuộc tính SIFT trước tiên được trích xuất từ một

tập các ảnh tham chiếu và lưu trữ trong cơ sở dữ liệu. Một ảnh mới được đối sánh bằng cách so sánh các thuộc tính riêng lẻ từ ảnh mới với cơ sở dữ liệu và tìm thuộc tính đối sánh dựa trên khoảng cách ocolid của các véc tơ thuộc tính. Thuật toán lảng giềng gần được sử dụng để có thể thực hiện các tính toán này nhanh chóng đối với cơ sở dữ liệu lớn.

Mỗi cụm Hough gồm ít nhất 3 thuộc tính giống với đối tượng và cần xác minh. Trước tiên một ước tính tối thiểu bình phương được thực hiện cho một xấp xỉ Affine với mỗi đối tượng. Bất kỳ thuộc tính hình ảnh nào khác phù hợp sẽ được nhận dạng và sự chênh lệch sẽ bị loại bỏ. Cuối cùng, ta sẽ có một tính toán chi tiết để tính xác suất để một tập hợp các thuộc tính chỉ ra sự hiện diện của một đối tượng, đem lại độ chính xác cho phép đối sánh. Đối sánh đối tượng qua các phép kiểm tra này có thể được xác định với độ tin cậy cao.

2.1.2. Các nghiên cứu liên quan

Sự phát triển của bài toán đối sánh ảnh bằng cách sử dụng một tập hợp các điểm quan tâm địa phương có thể được truy ngược trở lại công việc của Moravec (1981) về việc sử dụng một máy dò góc. Các máy dò Moravec được cải thiện bằng cách Harris và Stephens (1988) làm cho nó có thể lắp lại nhiều hơn dưới các phép biến dạng hình ảnh nhỏ và gần biên. Harris cũng cho thấy hiệu quả của nó trong việc theo dõi chuyển động và khôi phục được cấu trúc 3D từ chuyển động (Harris, 1992), các góc dò Harris đã được sử dụng rộng rãi từ đó cho nhiều công việc đối sánh với hình ảnh khác. Các thiết bị dò thuộc tính này thường được gọi là máy dò góc, họ không chỉ chọn góc mà hơn nữa là định vị bất kỳ hình ảnh có độ dốc lớn trong tất cả các hướng cùng ở cùng một tỉ lệ xác định.

Các máy dò góc Harris rất nhạy cảm với những thay đổi trong tỉ lệ ảnh, vì vậy nó không cung cấp một nền tảng tốt phù hợp với hình ảnh với kích cỡ khác nhau. Trước đó công trình của các tác giả (Lowe, 1999) cũng mở rộng cách tiếp cận thuộc tính cục bộ để đạt được tỉ lệ bất biến. Công việc này cũng mô tả một bộ mô tả địa phương mới cung cấp các thuộc tính đặc biệt hơn và ít nhạy cảm với biến dạng hình ảnh cục bộ như thay đổi khung nhìn 3D. Điều này cung cấp một nghiên cứu sâu hơn trong việc phân tích và trình bày một số cải tiến trong việc ổn định các thuộc tính bất biến.

Khung Affine cũng nhạy cảm với nhiều hơn so với các đặc điểm bất biến, vì vậy trong thực tế các thuộc tính Affine lắp lại ít hơn so với các đặc điểm bất biến trong biến dạng Affine với độ nghiêng 40 độ so với một bề mặt phẳng (Mikolajczyk, 2002). Hơn nữa bất biến Affine có thể không quan trọng đối với nhiều ứng dụng, ví dụ như thay đổi hướng nhìn là tốt nhất với vòng quay 30 độ trong khung nhìn (nghĩa là công nhận trong vòng 15 độ của điểm huấn luyện gần nhất) để nắm bắt những thay đổi không phẳng và các hiệu ứng tác động lên các đối

tượng 3D.

2.1.3. Phát hiện cực trị trong không gian tỉ lệ

Giai đoạn đầu tiên là phát hiện keypoint để tìm các khu vực và các tỉ lệ lặp đi lặp lại dưới các hướng nhìn khác nhau của cùng một đối tượng. Phát hiện địa điểm đó là bắt biến với tỉ lệ thay đổi của hình ảnh và có thể thực hiện bằng cách tìm kiếm các thuộc tính ổn định trên tất cả các tỉ lệ, có thể dùng một hàm liên tục của tỉ lệ được gọi là không gian tỉ lệ (Witkin, 1983). Nó đã được chứng minh bởi Koenderink (1984) và Lindeberg (1994) mà theo một loạt các giả định hợp lý thì chỉ có thể nhân rộng không gian là hàm Gaussian. Vì thế nên không gian tỉ lệ của một hình ảnh được định nghĩa như một hàm $L(x, y, \sigma)$ được tạo ra từ phép nhân chập một biến tỉ lệ Gaussian $G(x, y, \sigma)$ với một hình ảnh đầu vào $I(x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (2.1)$$

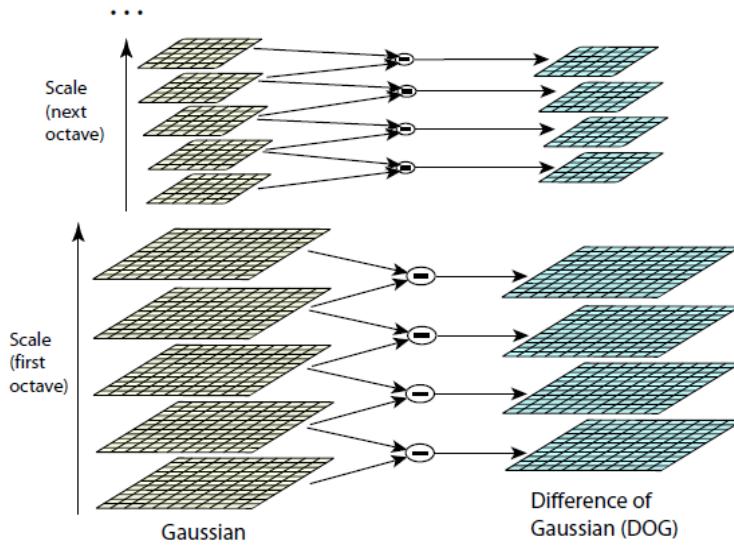
Trong đó $*$ là phép toán nhân chập giữa x, y và:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \quad (2.2)$$

Để phát hiện địa điểm Keypoint ổn định và hiệu quả trong không gian tỉ lệ, Lowe đã đề xuất sử dụng không gian cực trị dùng các hàm Gaussian khác nhau với các hình ảnh $D(x, y, \sigma)$, chúng có thể được tính toán từ sự khác biệt của hai tỉ lệ lân cận cách nhau bởi một số hằng số k không đổi:

$$\begin{aligned} D(x, y, \sigma) &= G(x, y, k\sigma) - G(x, y, k\sigma) * I(x, y) \quad (2.3) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned}$$

Có một số lý do cho việc lựa chọn hàm này. Đầu tiên nó là một hàm đặc biệt về hiệu suất để tính toán như những hình ảnh mịn L cần phải được tính toán trong bất kỳ bộ mô tả thuộc tính không gian tỉ lệ nào và D có thể được tính bằng cách đơn giản là trừ hình ảnh.



Hình 2.1: Mô tả hàm Gaussian và hàm Difference-of-Gaussian (DoG)

Ngoài ra, các hàm Gaussian khác nhau cung cấp một xấp xỉ gần Laplacian tỉ lệ. Bình thường Laplacian của Gaussian là $\sigma^2 \nabla^2 G$ như nghiên cứu bởi Lindeberg (1994). Lindeberg cho thấy rằng Laplacian bình thường với các yếu tố σ^2 là thực sự cần thiết cho tỉ lệ bất biến. Trong so sánh thử nghiệm chi tiết Mikolajczyk (2002) thấy rằng các cực đại và cực tiêu của $\sigma^2 \nabla^2 G$ tạo nên các thuộc tính hình ảnh ổn định nhất so với một các hàm hình ảnh khác chẳng hạn như gradient, Hessian hoặc hàm của góc Harris.

Mối quan hệ giữa D và $\sigma^2 \nabla^2 G$ như sau:

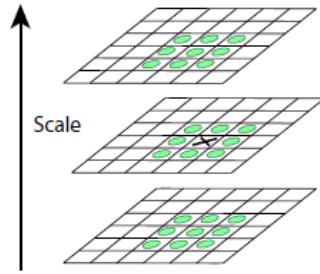
$$\frac{dG}{d\sigma} = \sigma \nabla^2 G \quad (2.4)$$

Từ đây, chúng ta thấy rằng $\nabla^2 G$ có thể được tính xấp xỉ để $\partial G / \partial \sigma$ đạt sự khác biệt gần nhất về tỉ lệ tại $k\sigma$ và σ :

$$\sigma \nabla^2 G = \frac{dG}{d\sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma} \quad (2.5)$$

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G \quad (2.6)$$

Điều này cho thấy rằng khi các hàm khác của hàm Gaussian có tỉ lệ khác nhau bởi một hằng số có quan hệ chặt chẽ với tỉ lệ σ^2 cho tỉ lệ bất biến Laplacian. Các yếu tố ($k - 1$) trong phương trình là một hằng số trên tất cả tỉ lệ và do đó không ảnh hưởng đến vị trí cực trị. Các lõi xấp xỉ sẽ trả về 0 khi k tiến đến 1, nhưng trong thực tế, người ta đã tìm thấy rằng xấp xỉ gần như không có tác động đến sự ổn định của việc phát hiện cực trị hoặc địa phương hóa đối với sự khác biệt quan trọng về tỉ lệ, như $k = \sqrt{2}$.

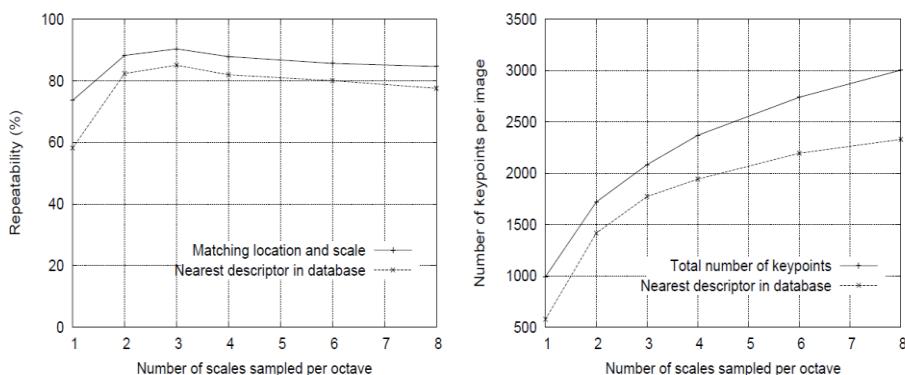


Hình 2.2: Phát hiện cực trị của hàm DoG

Một cách tiếp cận hiệu quả để xây dựng $D(x,y,\sigma)$ được thể hiện trong Hình 2.1. Hình ảnh ban đầu là từng bước kết hợp với Gaussian để tạo ra hình ảnh riêng biệt bởi hằng số k trong không gian tỉ lệ hiện xếp chồng lên nhau trong cột bên trái. Ở đây ta chọn cách phân chia từng octave của không gian tỉ lệ (tức là gấp đôi σ) thành một số nguyên s , vì vậy $k = 2^{m/s}$. Chúng ta phải tạo ra $s + 3$ ảnh trong chồng hình ảnh mờ cho mỗi octave, vì thế cuối cùng việc phát hiện cực trị bao phủ một octave hoàn chỉnh. Tỉ lệ ảnh liền kề được trừ cho nhau để tạo sự khác biệt của ảnh Gaussian hiển thị bên phải. Khi một octave hoàn chỉnh đã được xử lý, chúng ta đổi mẫu hình Gaussian có giá trị khởi tạo gấp đôi σ (nó sẽ có 2 hình ảnh từ phía trên cùng của ngăn xếp) bằng cách lấy mỗi điểm ảnh thứ hai trong mỗi hàng và cột. Độ chính xác của mẫu so với σ là không có khác biệt so với thời điểm khởi tạo octave trước đó, trong khi các phép tính toán được giảm đi rất nhiều.

2.1.3.1. Phát hiện cực trị địa phương

Để phát hiện cực đại và cực tiểu địa phương của $D(x, y, \sigma)$, mỗi điểm mẫu được so sánh với tám điểm láng giềng của bức ảnh hiện tại và chín điểm láng giềng ở tỉ lệ trên và dưới. Nó được chọn khi và chỉ khi nó lớn hơn tất cả các nước láng giềng hoặc nhỏ hơn tất cả. Chi phí của việc kiểm tra này là khá thấp do thực tế hầu hết các điểm láng giềng sẽ được loại bỏ sau lần đầu kiểm tra.



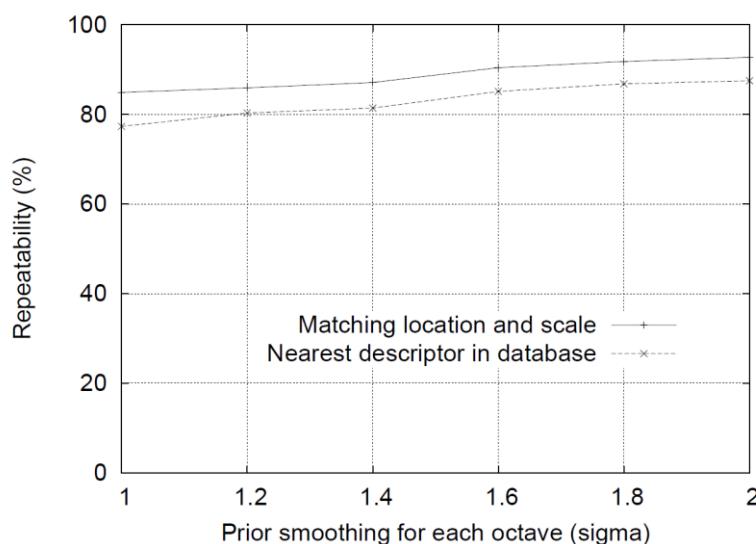
Hình 2.3: Số lượng mẫu tỷ lệ trên mỗi Octave

Vì vậy, chúng ta phải giải quyết một giải pháp chuyển đổi về hiệu năng. Trong thực tế, điều này có thể được minh chứng bằng các thí nghiệm. Các cực trị đó gần nhau là khá ổn định với những nhiễu loạn nhỏ của hình ảnh. Ta có thể xác định những thực nghiệm tốt nhất bằng cách nghiên cứu một loạt các tần số lấy mẫu và sử dụng các kết quả đáng tin cậy nhất trong một mô phỏng thực tế.

2.1.3.2. Tần suất lấy mẫu tỉ lệ

Ta thực hiện việc đổi sánh dùng một bộ sưu tập 32 hình ảnh thực tế rất đa dạng, bao gồm cả ngoại cảnh, khuôn mặt người, hình ảnh trên không và hình ảnh công nghiệp (miền hình ảnh đã được tìm thấy hầu như không có ảnh hưởng đến bất kỳ kết quả nào). Mỗi hình ảnh sau đó đã phải chịu một loạt các biến đổi, bao gồm quay, thay đổi tỉ lệ, Affine, sự thay đổi về độ sáng và độ tương phản và bổ sung các nhiễu hình ảnh. Bởi vì những thay đổi này là tổng hợp, nó đã có thể dự đoán chính xác nơi mỗi thuộc tính trong một hình ảnh ban đầu sẽ xuất hiện trong hình ảnh chuyển đổi, cho phép đo lặp lại chính xác và độ chính xác vị trí cho mỗi thuộc tính.

Hình 2.4 cho thấy các kết quả mô phỏng được sử dụng để kiểm tra tác động của thay đổi số lượng tỉ lệ mỗi octave mà tại đó các chức năng chụp ảnh được lấy mẫu trước khi phát hiện cực trị. Trong trường hợp này, mỗi hình ảnh được lấy mẫu lại xoay sau bằng một góc ngẫu nhiên và nhân rộng bởi một số lượng ngẫu nhiên giữa 0,2 và 0,9 lần kích thước ban đầu. Keypoint từ các hình ảnh có độ phân giải giảm được đổi sánh với những điểm đó từ các hình ảnh gốc vì thế tỉ lệ cho tất cả các keypoint được thể hiện trong ảnh đổi sánh. Ngoài ra, 1% nhiễu hình ảnh đã được bổ sung, nghĩa là mỗi điểm ảnh đã thêm vào một số ngẫu nhiên từ khoảng thống nhất [-0,01,0,01] nơi các giá trị điểm ảnh nằm trong khoảng [0,1]



Hình 2.4: Thí nghiệm làm mịn cho mỗi Octave

Dòng trên cùng trong đồ thị đầu tiên của Hình 3 cho thấy số phần trăm keypoint được phát hiện tại địa điểm đối sánh và tỉ lệ trong hình ảnh chuyển đổi. Đối với tất cả các ví dụ này, tỉ lệ đối sánh là $\sqrt{2}$ của tỉ lệ chính xác và vị trí đối sánh là 0 trong pixels σ , σ là tỉ lệ của các keypoint (định nghĩa phương trình (1) là độ lệch chuẩn của Gaussian nhỏ nhất được sử dụng trong hàm DOG). Các dòng thấp hơn trên biểu đồ này cho thấy số lượng các keypoint được đối sánh một cách chính xác đến một cơ sở dữ liệu gồm 40.000 keypoint sử dụng thủ tục đối sánh lảng giềng gần để mô tả trong phần 6 (điều này cho thấy rằng một khi các keypoint được lặp đi lặp lại, nó có khả năng là hữu ích cho nhận dạng và phù hợp với nhiệm vụ đối sánh). Như biểu đồ này cho thấy, độ lặp lại cao nhất thu được khi lấy mẫu 3 thang mỗi octave.

Số keypoint tăng lên với việc tăng tỉ lệ mẫu và tổng số các đối sánh đúng cũng tăng. Từ thành công trong nhận dạng đối tượng thường phụ thuộc nhiều vào số lượng keypoint đối sánh đúng, và phần trăm đối sánh đúng cũng tăng, nhiều ứng dụng sẽ được tối ưu để sử dụng một số lượng lớn các mẫu tỉ lệ. Tuy nhiên, chi phí của việc tính toán cũng tăng lên với con số này, vì vậy mà ta lựa chọn sử dụng chỉ 3 mẫu tỉ lệ mỗi octave.

Các thí nghiệm cho thấy rằng hàm không gian tỉ lệ hàm DOG có một số lượng lớn các cực trị và nó sẽ rất tốn kém để phát hiện tất cả. Và điều may mắn là ta có thể phát hiện các tập con ổn định nhất và hữu ích ngay cả với một mẫu thô của tỉ lệ.

2.1.3.3. Tần suất lấy mẫu trong miền không gian

Để xác định tần số lấy mẫu cho mỗi octave của không gian tỉ lệ thì phải xác định tần số lấy mẫu trong hình ảnh liên quan đến tỉ lệ của độ mịn. Giả sử rằng cực trị có thể được tự ý gần nhau, sẽ có một sự hoán đổi tương tự giữa tần số lấy mẫu và tỷ lệ phát hiện. Hình 2.4 cho thấy thực nghiệm của lượng làm mịn trước khi σ được áp dụng cho từng cấp hình ảnh trước khi xây dựng các không gian biểu diễn tỉ lệ cho một octave. Dòng trên cùng là lặp lại của phát hiện keypoint và kết quả cho thấy rằng khả năng lặp lại tiếp tục tăng với σ . Tuy nhiên, nếu chọn σ quá lớn thì lại mất nhiều thời gian, để tăng hiệu quả ta lựa chọn $\sigma = 1.6$ cung cấp gần lặp lại tối ưu. Giá trị này được sử dụng trong suốt báo cáo này và đã được sử dụng cho các kết quả trong hình 2.3.

Tất nhiên, nếu ta làm mịn hình ảnh trước khi phát hiện cực trị, ta đang loại bỏ hiệu quả của các tần số không gian cao nhất. Vì vậy, để sử dụng đầy đủ các đầu vào, các hình ảnh có thể được mở rộng để tạo thêm nhiều điểm hơn mẫu đã có mặt trong bản gốc. Ta tiến hành nhân đôi kích thước của hình ảnh đầu vào sử dụng nội suy tuyến tính trước khi xây dựng các mức đầu tiên của kim tự tháp. Trong khi các hoạt động tương đương có thể có hiệu quả đã được thực hiện bởi việc dùng bộ lọc bù tập con điểm ảnh trên ảnh gốc, tăng gấp đôi hình ảnh dẫn đến việc

thực hiện hiệu quả hơn. Ta giả định rằng các hình ảnh ban đầu có một vệt mờ tối thiểu $\sigma = 0,5$ (mức tối thiểu cần thiết để ngăn chặn hiện tượng rỗng cưa tại đường biên ảnh), và do đó để tăng các điểm ảnh ta cần tăng gấp đôi giá trị $\sigma = 1,0$. Điều này có nghĩa rằng việc làm mịn bổ sung là cần thiết trước khi tạo ra các octave đầu tiên của không gian tỉ lệ. Việc tăng gấp đôi hình ảnh làm tăng số lượng các keypoint ổn định gần gấp 4.

2.1.4. Định vị chính xác Keypoint

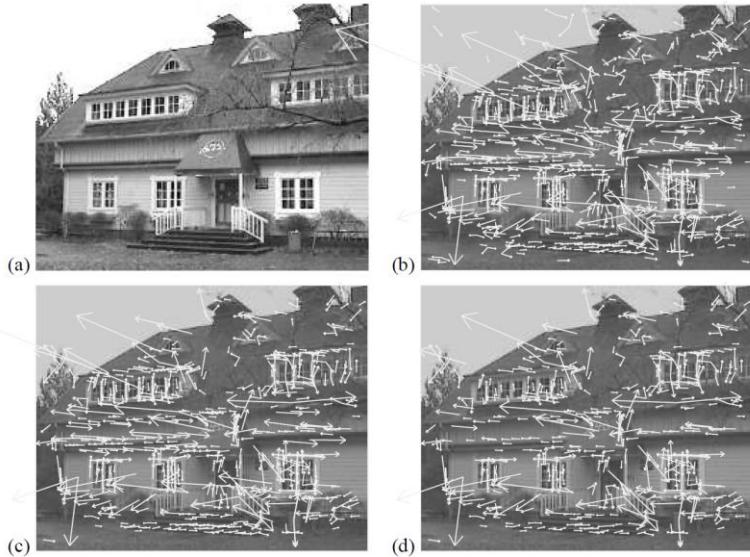
Khi một ứng viên keypoint đã được tìm thấy bằng cách so sánh một pixel với hàng xóm của mình, bước tiếp theo là để thực hiện một cách chi tiết để các dữ liệu trong khu vực với vị trí, tỉ lệ và tỉ lệ của độ cong chính. Điều này cho phép các điểm được loại bỏ khi có độ tương phản thấp (và do đó nhạy cảm với nhiễu) hoặc ít được địa phương hóa dọc theo một cạnh.

Việc thực hiện ban đầu của phương pháp này (Lowe, 1999) chỉ đơn giản là định vị keypoint vào vị trí và tỉ lệ của các điểm mẫu trung tâm. Tuy nhiên, thời gian gần đây Brown đã phát triển một phương pháp (Brown và Lowe, 2002) cho một hàm bậc hai 3D vừa khít với các điểm lấy địa phương để xác định vị trí nội suy tối đa, và thí nghiệm của ông cho thấy rằng việc này cung cấp một sự cải thiện đáng kể phù hợp và ổn định. Cách tiếp cận của ông sử dụng các mở rộng Taylor (lên đến các phương trình bậc hai) của hàm tỉ lệ không gian, $D(x, y, \sigma)$, dịch chuyển sao mà nguồn gốc là ở vị trí mẫu:

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 y}{\partial x^2} x \quad (2.7)$$

Trong đó D và các dẫn xuất của nó được đánh giá ở vị trí mẫu và $x = (x, y, \sigma)$. T là phần bù đắp từ vị trí này. Các vị trí của các cực trị x được xác định bằng cách lấy đạo hàm của hàm này đối với x và gán nó bằng 0, cho

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2} \frac{\partial D}{\partial x} \quad (2.8)$$



Hình 2.5: Các giai đoạn lựa chọn các điểm Keypoint

Theo đề xuất của Brown, Hessian và dẫn xuất của D được tính xấp xỉ bằng cách sử dụng những khác biệt của các điểm mẫu lân cận. Kết quả là hệ thống tuyến tính 3×3 có thể được giải quyết với chi phí tối thiểu. Nếu phần bù lớn hơn 0,5 lần kích thước bất kỳ, điều đó có nghĩa là nó gần hơn với một mẫu khác. Trong trường hợp này, các điểm mẫu được thay đổi và suy diễn thay vì về điểm đó. Cuối cùng phần bù x được thêm vào vị trí của điểm mẫu của nó để có được các ước tính nội suy cho vị trí của các cực trị.

Các giá trị hàm tại cực trị $D(x)$ rất hữu ích cho việc loại bỏ cực trị không ổn định với độ tương phản thấp. Điều này có thể thu được bằng cách thay thế phương trình (3) vào (2), cho

$$\hat{D(x)} = D + \frac{1}{2} \frac{\partial D^T}{\partial x} \hat{x} \quad (2.9)$$

Đối với các thí nghiệm này, tất cả các cực trị với một giá trị của $|D(x)|$ ít hơn 0.03 sẽ bị loại bỏ (ta giả định các giá trị điểm ảnh trong khoảng $[0,1]$).

Hình 2.5 cho thấy những ảnh hưởng của lựa chọn keypoint trên một hình ảnh tự nhiên. Để tránh quá nhiều lỗ xộn, một độ phân giải điểm ảnh thấp 233×189 được sử dụng và keypoint được hiển thị như là vectơ cho vị trí, tỉ lệ và hướng của mỗi keypoint (phân hướng được mô tả dưới đây). Hình 5(a) cho thấy những hình ảnh ban đầu được hiển thị ở độ tương phản giảm sau hình tiếp theo. Hình(b) hiển thị 832 keypoint trên tất cả các cực đại và cực tiêu tìm được của hàm DOG, trong đó hình (c) hiển thị 729 keypoint còn lại sau khi loại bỏ các giá trị $d(x)$ nhỏ hơn 0.03.

2.1.4.1. Loại bỏ phản ứng biên

Sự ổn định không đủ để loại keypoint với độ tương phản thấp. Các hàm DOG sẽ có một đáp ứng mạnh mẽ dọc theo các biên, ngay cả khi các vị trí dọc theo các biên là khó xác định và do đó không ổn định với một lượng nhỏ của nhiễu.

Một điều khó định nghĩa trong hàm DOG sẽ có một độ cong chính lớn trên biên nhưng một lượng nhỏ theo hướng vuông góc. Các đường cong chính có thể được tính toán từ một ma trận Hessian $H(2x2)$ tính theo vị trí và tỉ lệ của các Keypoint:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \quad (2.10)$$

Các dẫn xuất được ước tính bằng cách lấy sự khác biệt của các điểm mẫu lân cận.

Các giá trị riêng của H là tỷ lệ thuận với độ cong chính của D . Từ cách tiếp cận được sử dụng bởi Harris và Stephens (1988), ta có thể tránh được việc tính toán các giá trị đặc trưng, ta chỉ quan tâm đến tỷ lệ của chúng.

Cho α là eigenvalue với cường độ lớn nhất và β là nhỏ hơn. Sau đó, ta có thể tính tổng các giá trị đặc trưng từ các dấu 0.03, vết của H và kết quả từ việc xác định là:

$$\text{Tr}(H) = D_{xx} + D_{yy} = \alpha + \beta, \quad (2.11)$$

$$\text{Det}(H) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta \quad (2.12)$$

Trong trường hợp không chắc các yếu tố xác định là không tốt, độ cong có những dấu hiệu khác nhau thì điểm đó bị bỏ đi vì không có một cự trị. Cho r là tỷ số giữa độ lớn eigenvalue lớn nhất và nhỏ hơn, do đó $\alpha = r\beta$. Vì vậy,

$$\frac{\text{Tr}(H)^2}{\text{Det}(H)} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(r\beta + \beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \quad (2.13)$$

Chỉ phụ thuộc vào tỷ lệ của các giá trị đặc trưng hơn là giá trị riêng lẻ của nó. Số lượng $(r+1)^2/r$ là ở mức tối thiểu khi hai giá trị riêng là bằng nhau và nó tăng theo r . Vì vậy, để kiểm tra tỷ lệ của độ cong chính là một ngưỡng r dưới đây chúng ta chỉ cần kiểm tra:

$$\frac{\text{Tr}(H)^2}{\text{Det}(H)} < \frac{(r+1)^2}{r} \quad (2.14)$$

Đây là tính toán rất hiệu quả với chưa đến 20 điểm nổi bật cần phải kiểm tra từng keypoint. Ở đây ta sử dụng một giá trị của $r = 10$ trong đó loại bỏ keypoint có tỷ lệ giữa đường

cong lớn hơn 10. Việc chuyển đổi từ hình 5 (c) và (d) cho thấy ảnh hưởng của hoạt động này.

2.1.5. Gán hướng

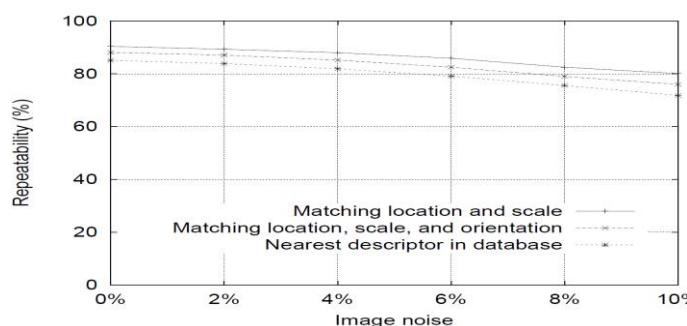
Bằng cách gán một hướng phù hợp với từng keypoint dựa trên các thuộc tính hình ảnh cục bộ, các bộ mô tả keypoint có thể liên quan đến hướng và do đó đạt được sự ổn định khi xoay hình ảnh. Tỉ lệ của các keypoint được sử dụng để chọn hình ảnh Gaussian mịn L với tỉ lệ gần nhất, vì thế tất cả các tính toán được thực hiện một cách bất biến tỉ lệ. Đối với mỗi hình ảnh mẫu $L(x, y)$ ở tỉ lệ này, độ lớn gradient $m(x, y)$ và hướng $\theta(x, y)$ được tính toán trước do sự khác biệt điểm ảnh:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \quad (2.15)$$

$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y))) \quad (2.16)$$

Một biểu đồ hướng được hình thành từ những hướng dốc của điểm lấy mẫu trong khu vực xung quanh các keypoint. Hướng biểu đồ tần số có 36 ngăn (bin) bao phủ 360 độ của hướng. Mỗi mẫu thêm vào biểu đồ được gán trọng số bằng độ lớn Gradient của nó và bởi một hình tròn trọng số Gaussian với σ gấp 1,5 lần so với tỉ lệ của các keypoint.

Hình 2.6 cho thấy sự ổn định vị trí, tỉ lệ, hướng và được gán hướng khác nhau với nhiễu ảnh. Trước những hình ảnh được quay và thu nhỏ lại bởi một lượng ngẫu nhiên, dòng đầu cho thấy sự ổn định của vị trí keypoint và gán tỉ lệ. Dòng thứ hai cho thấy sự ổn định phù hợp khi gán hướng (yêu cầu trong khoảng 15 độ). Khoảng cách giữa hai dòng trên cùng thể hiện việc gán hướng vẫn chính xác 95% ngay cả sau khi bổ sung $\pm 10\%$ nhiễu ảnh (tương đương với một camera cung cấp ít hơn 3 bit chính xác). Các cách đo biến đổi hướng cho các đối sánh chính xác là khoảng 2,5 độ, tăng lên 3,9 độ cho 10% nhiễu. Điểm mấu chốt trong hình 6 cho thấy đối sánh đúng một mô tả chính xác keypoint đến một cơ sở dữ liệu của 40.000. Biểu đồ sau cho thấy các thuộc tính SIFT làm việc tốt ngay cả một lượng lớn các nhiễu pixel và các nguyên nhân chính gây lỗi là vị trí và tỉ lệ phát hiện ban đầu.

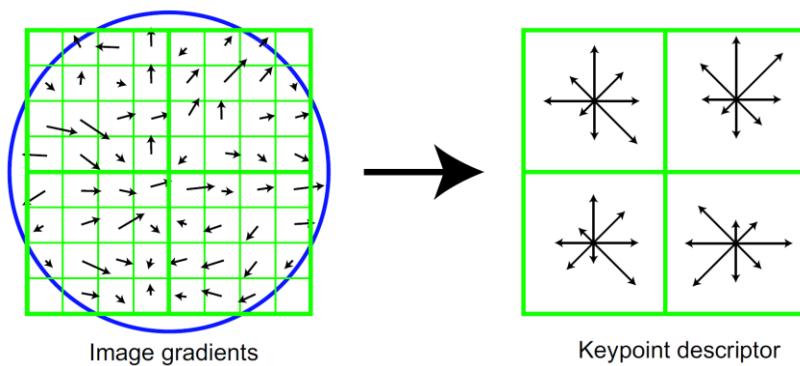


Hình 2.6: Đồ thị độ nhiễu (%) của ảnh

2.1.6. Bộ mô tả hình ảnh cục bộ

Các thao tác trước đó đã được gán một vị trí ảnh, tỉ lệ và hướng đến mỗi điểm Keypoint. Những thông số ám chỉ sự lặp lại vị trí hệ tọa độ 2D trong đó mô tả các vùng ảnh cục bộ và do đó bắt biến các thông số này. Bước tiếp theo là tính toán mô tả cho các khu vực hình ảnh cục bộ mà đặc biệt là chưa bắt biến với các biến thể còn lại, chẳng hạn như thay đổi độ sáng hoặc hướng nhìn 3D.

Một cách tiếp cận là một mẫu cường độ ảnh cục bộ xung quanh keypoint ở tỉ lệ thích hợp, và để đối sánh chúng với các cách sử dụng biện pháp tương quan bình thường. Tuy nhiên, tương quan đơn giản của các bản vá lỗi hình ảnh rất nhạy cảm với những thay đổi, chẳng hạn như Affine hoặc thay đổi hướng nhìn 3D hay biến dạng mềm. Cách tiếp cận tốt hơn đã được chứng minh bởi Edelman, Intrator, và Poggio (1997). Họ đề xuất dựa trên một mô hình thị giác sinh học, đặc biệt là các tế bào thần kinh phức tạp trong vỏ não thị giác chính. Những tế bào thần kinh phức tạp đáp ứng với một gradient ở một hướng cụ thể và tần số không gian, nhưng vị trí của gradient trên võng mạc được phép thay đổi theo một lĩnh vực nhỏ hơn được cục bộ hóa một cách chính xác. Edelman et al. giả thuyết rằng chức năng của các tế bào thần kinh phức tạp này là cho phép đối sánh và nhận dạng của đối tượng 3D từ một vùng của hướng nhìn. Họ đã thực hiện thí nghiệm chi tiết sử dụng mô hình máy tính 3D của hình dạng đối tượng và động vật mà thấy phù hợp với gradients trong khi cho phép thay đổi vị trí của chúng tốt hơn khi xoay 3D. Ví dụ, nhận dạng chính xác cho các đối tượng 3D xoay theo chiều sâu bằng 20 độ tăng từ 35% cho mối tương quan của gradient đến 94% bằng cách sử dụng mô hình tế bào phức tạp. Thực hiện của chúng tôi mô tả dưới đây được lấy cảm hứng từ ý tưởng này, nhưng cho phép thay đổi vị trí bằng cách sử dụng một cơ chế tính toán khác nhau.



Hình 2.7: Hướng phân bố trên ảnh và bộ mô tả các điểm Keypoint

2.1.6.1. Bộ mô tả

Hình 2.7 minh họa các tính toán của các bộ mô tả keypoint. Đầu tiên là độ lớn gradient và hướng được lấy mẫu xung quanh vị trí keypoint sử dụng tỉ lệ của các keypoint để lựa chọn cấp độ mờ Gaussian cho hình ảnh. Để đạt được hướng bất biến, tọa độ của các mô tả và độ dốc được xoay tương đối với hướng keypoint. Để đạt hiệu quả, gradient được tính toán trước ở tất cả các mức của các kim tự tháp như mô tả trong phần 5. Những minh họa bằng các mũi tên nhỏ ở mỗi vị trí lấy mẫu bên trái của Hình 2.7.

Bộ mô tả được hình thành từ một vector chứa các giá trị của tất cả các thực thể histogram tương ứng với chiều dài của mũi tên bên phải của Hình 2.7. Hình vẽ cho thấy một mảng 2×2 biểu đồ hướng, trong khi các thí nghiệm dưới đây cho thấy rằng kết quả tốt nhất đạt được với một mảng 4×4 biểu đồ với 8 hướng trong từng vùng. Do đó, các thí nghiệm này sử dụng một vector đặc trưng $4 \times 4 \times 8 = 128$ phần tử cho mỗi Keypoint.

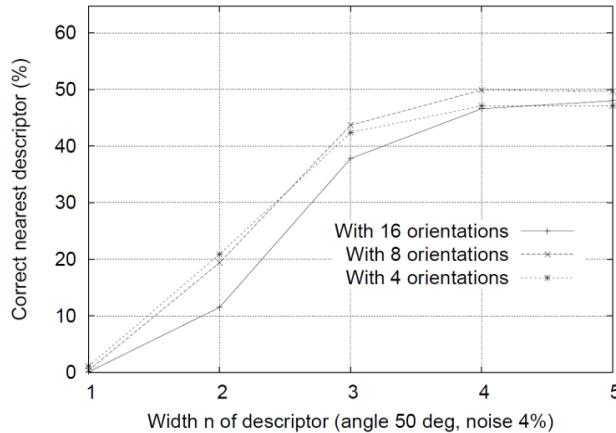
Khi thay đổi độ sáng trong đó một hằng số được thêm vào mỗi điểm ảnh hình ảnh thì sẽ không ảnh hưởng đến giá trị gradient khi chúng được tính từ sự khác biệt pixel. Do đó, các mô tả là bất biến để thay đổi Affine trong chiều sáng. Tuy nhiên, những thay đổi ánh sáng phi tuyến tính cũng có thể xảy ra do độ bão hòa của máy ảnh hoặc do sự thay đổi ánh sáng có ảnh hưởng đến bề mặt 3D với hướng khác nhau. Các hiệu ứng này có thể gây ra một sự thay đổi tương đối lớn cho một gradient, nhưng ít có khả năng ảnh hưởng đến hướng gradient. Do đó, ta sẽ làm giảm ảnh hưởng của độ dốc lớn bởi các giá trị ngưỡng trong các vector đặc trưng cho mỗi đơn vị, ngưỡng này không được lớn hơn 0.2 và sau đó đưa về giá trị bình thường cho mỗi đơn vị chiều dài. Điều này có nghĩa là sự phù hợp với độ lớn cho gradient không còn là quan trọng và sự phân bố các hướng có trọng tâm hơn. Giá trị của 0.2 được xác định bằng thực nghiệm bằng cách sử dụng các hình ảnh có chứa sự chiều sáng khác nhau đối với các đối tượng 3D.

2.1.6.2. Kiểm thử Descriptor

Có hai tham số có thể được sử dụng để thay đổi độ phức tạp của mô tả: số lượng hướng r trong biểu đồ và chiều rộng n của mảng $n \times n$ các hướng của biểu đồ. Kích thước của vector mô tả kết quả là rn^2 . Như sự phức tạp của mô tả phát triển, nó có thể phân biệt rõ hơn trong một cơ sở dữ liệu lớn, nhưng nó cũng sẽ nhạy cảm hơn với biến dạng hình và làm bế tắc công việc.

Hình 2.8 cho thấy kết quả thực nghiệm trong đó số các hướng và kích thước của các mô tả đã được thay đổi. Các đồ thị đã được tạo ra cho một chuyển đổi khung nhìn trong đó một mặt phẳng nghiêng 50 độ so với hướng nhìn và 4% nhiễu hình ảnh được thêm vào. Điều này là giới hạn gần của đối sánh đáng tin cậy, vì đây là những trường hợp khó hơn và trong các

trường hợp này thì thực hiện mô tả là quan trọng nhất. Kết quả thể hiện số phần trăm keypoint được đối sánh đúng so với láng giềng gần nhất trong cơ sở dữ liệu của 40.000 keypoint. Đồ thị cho thấy một xu hướng biểu đồ duy nhất ($n = 1$) là rất ít tại các điểm khác biệt, nhưng kết quả tiếp tục cải thiện lên đến một mảng 4×4 của biểu đồ với 8 hướng. Khi số hướng tăng lên hoặc một mô tả lớn hơn có thể thực sự làm sai lệch việc đối sánh bằng cách làm cho các mô tả nhạy cảm hơn với sự biến dạng. Những kết quả này là tương tự nhau với thay đổi điểm nhìn và nhiễu, mặc dù trong một số trường hợp đơn giản sự khác biệt tiếp tục cải thiện (từ mức cao) với 5×5 và kích thước bộ mô tả lớn. Ở đây ta sử dụng một mô tả 4×4 với 8 hướng, dẫn đến các vector với 128 chiều. Trong khi số chiều của mô tả có vẻ nhiều và ta đã tìm thấy rằng nó luôn thực hiện tốt hơn so với mô tả dưới chiều trên một loạt các đối sánh phù hợp và các chi phí tính toán của so khớp vẫn thấp khi sử dụng các phương pháp láng giềng gần nhất.



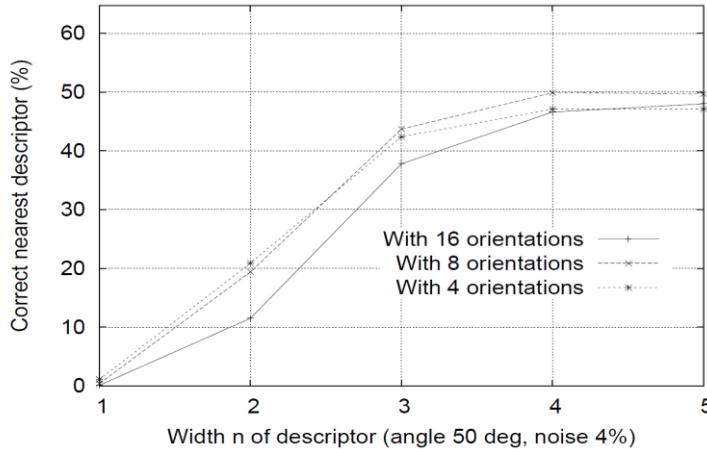
Hình 2.8: Độ rộng của bộ mô tả (góc 50 độ, độ nhiễu ảnh 4%)

2.1.6.3. Độ nhạy với biến đổi Affine

Độ nhạy của các mô tả trong thay đổi Affine được kiểm tra trong Hình 2.9. Biểu đồ thể hiện độ tin cậy của điểm keypoint và lựa chọn tỉ lệ, phân hướng, đối sánh láng giềng gần nhất với một cơ sở dữ liệu như là một hàm số của phép quay theo chiều sâu so với hướng nhìn. Có thể thấy rằng mỗi giai đoạn tính toán đã làm giảm khả năng lặp lại với việc tăng biến dạng Affine nhưng các so khớp chính xác vẫn ở trên mức 50% với sự thay đổi 50 độ của hướng nhìn.

Để đạt được độ tin cậy khi đối sánh trên một khung nhìn rộng hơn, một trong các máy dò bất biến Affine có thể được dùng để chọn và lấy mẫu các khu vực ảnh như trong phần 2. Như đã đề cập ở trên, không cách tiếp cận nào trong số những phương pháp biến đổi Affine bất biến thực sự, tất cả đều bắt đầu từ việc xác định thuộc tính ban đầu khi không bất biến affine. Điều đó thể hiện Phương pháp tốt nhất về bất biến Affine. Mikolajczyk (2002) đã đề xuất và chạy thử nghiệm chi tiết với các máy dò Harris-ne Affine. Ông thấy rằng các keypoint lặp lại dưới dưới một góc nhìn 50 độ và nó vẫn đạt gần 40% dưới góc nhìn 70 độ, nó cung cấp

hiệu suất tốt hơn cho những thay đổi Affine lớn. Nhưng nhược điểm là chi phí tính toán cao hơn nhiều, giảm số lượng các keypoint, và tính ổn định kém hơn cho những thay đổi Affine nhỏ do sai sót trong việc gán một khung Affine phù hợp dưới nhiễu. Trong thực tế, phạm vi cho phép quay cho các đổi tượng 3D là ít hơn đáng kể hơn so với bề mặt phẳng, vì vậy Affine bất biến thường không phải là yếu tố hạn chế trong khả năng để phù hợp với sự thay đổi quan điểm trên. Nếu một phạm vi rộng của Affine bất biến là mong muốn, chẳng hạn như đối với một bề mặt được biết đến là phẳng, sau đó là một giải pháp đơn giản là áp dụng phương pháp tiếp cận của Pritchard và Heidrich (2003), trong đó thuộc tính SIFT bổ sung được tạo ra từ biến đổi Affine phiên bản 4 của hình ảnh huấn luyện tương ứng với thay đổi 60 độ của hướng nhìn, cho phép việc sử dụng các thuộc tính chuẩn SIFT và không phát sinh thêm chi phí khi các bức ảnh được nhận dạng, nhưng kết quả là tăng kích thước của cơ sở dữ liệu thuộc tính theo hệ số 3.



Hình 2.9: Sự ổn định của việc phát hiện vị trí các Keypoint

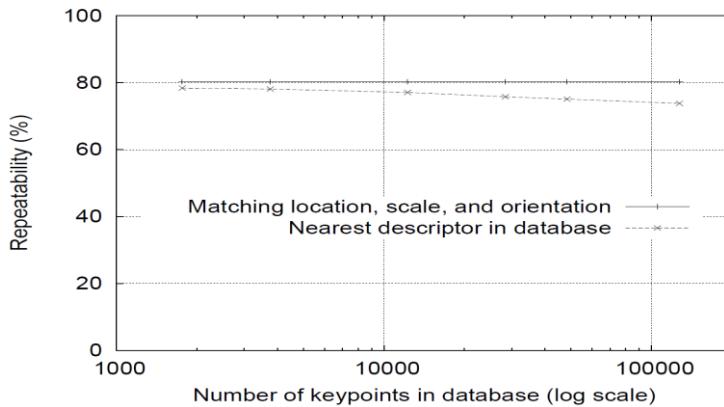
2.1.6.4. Kết hợp với cơ sở dữ liệu lớn

Một vấn đề còn quan trọng để đo sự khác biệt của thuộc tính là độ tin cậy của các biến đổi sánh như là một hàm như thế nào với số lượng các thuộc tính trong cơ sở dữ liệu đổi sánh. Với cách sử dụng một cơ sở dữ liệu 32 ảnh với khoảng 40.000 keypoint, hình 10 cho thấy độ tin cậy của các đổi sánh như một hàm của độ lớn cơ sở dữ liệu. Hình vẽ này đã được tạo ra bằng cách sử dụng một cơ sở dữ liệu lớn hơn 112 ảnh, với hướng nhìn xoay 30 độ và 2% nhiễu ảnh và lấy ảnh xoay ngẫu nhiên và thay đổi tỉ lệ.

Các đường nét đứt hiển thị một phần của thuộc tính ảnh mà những hàng xóm gần nhất trong cơ sở dữ liệu đổi sánh đúng như là một hàm của kích thước cơ sở dữ liệu hiển thị trên một tỉ lệ lôgarít. Các điểm tận cùng bên trái là phù hợp với các thuộc tính từ một hình ảnh duy nhất, trong khi các điểm ngoài cùng bên phải là lựa chọn phù hợp từ một cơ sở dữ liệu của tất

cả các thuộc tính từ 112 hình ảnh. Có thể thấy rằng độ tin cậy của đối sánh giảm như là một hàm của số lượng các sai số, nhưng tất cả các dấu hiệu cho thấy nhiều kết quả đúng sẽ tiếp tục được phát hiện ra khi kích thước cơ sở dữ liệu rất lớn.

Các dòng nét liền là tỷ lệ phần trăm của keypoint được nhận dạng tại vị trí đối sánh đúng và hướng trong hình ảnh chuyển đổi. Mỗi quan tâm của ta là khi khoảng cách giữa hai đường là nhỏ nghĩa là các đối sánh bị sai do việc khởi tạo các thuộc tính ban đầu và gán hướng chứ không phải do sự tính khác biệt về thuộc tính, thậm chí với kích thước cơ sở dữ liệu lớn.



Hình 2.10: Số lượng Keypoint trong cơ sở dữ liệu

2.1.7. Ứng dụng cho nhận dạng đối tượng

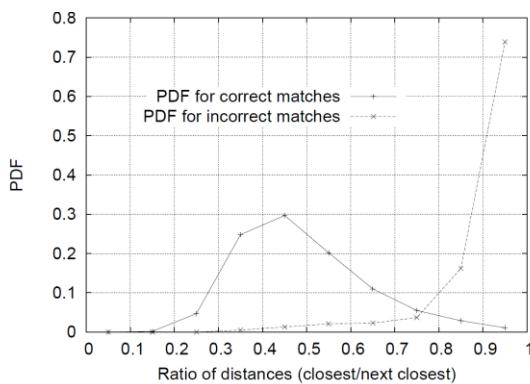
Nhận dạng đối tượng được thực hiện trước tiên bởi việc đối sánh từng keypoint độc lập với cơ sở dữ liệu của keypoint chiết xuất từ các hình ảnh huấn luyện. Nhiều đối sánh trong số những đối sánh đầu tiên sẽ là không chính xác, do thuộc tính không rõ ràng hoặc các thuộc tính phát sinh từ một nền lõn xộn. Do đó, các cụm ít nhất 3 thuộc tính đầu tiên được nhận dạng đúng về một đối tượng và tư thế của nó, việc đối sánh theo những cụm thuộc tính có xác suất cao hơn nhiều so với các đối sánh đặc điểm riêng biệt. Sau đó, mỗi cụm được kiểm tra bằng cách thực hiện một mô hình hình học chi tiết và kết quả được sử dụng để xem xét xem đối sánh trên đúng hay sai.

2.1.7.1. Khớp Keypoint

Đối sánh các keypoint tốt nhất được tìm thấy bằng cách xác định điểm hàng xóm gần nhất với nó trong cơ sở dữ liệu của keypoint từ hình ảnh huấn luyện. Điểm hàng xóm gần nhất được định nghĩa là các keypoint với khoảng cách Euclidean tối thiểu đối với các vector mô tả bất biến như đã được mô tả trong phần sau.

Tuy nhiên, nhiều thuộc tính từ một hình ảnh sẽ không có bất kỳ đối sánh nào chính xác trong cơ sở dữ liệu chi phí đào tạo bởi vì nó phát sinh từ nền lõn hoặc không được phát hiện

trong những hình ảnh huấn luyện. Đó là một cách hữu ích để loại bỏ dễ dàng các thuộc tính mà không có bất kỳ đối sánh nào với cơ sở dữ liệu. Một ngưỡng toàn cục về khoảng cách đến các thuộc tính gần nhất là không hiệu quả vì có nhiều bộ mô tả khác nhau về một đối tượng. Biện pháp hiệu quả hơn thu được bằng cách so sánh khoảng cách của những điểm hàng xóm gần nhất đó với điểm hàng xóm gần nhất thứ hai. Nếu có nhiều hình ảnh huấn luyện của cùng một đối tượng, ta sẽ định nghĩa điểm hàng xóm thứ hai từ hàng xóm gần nhất được biết đến từ một đối tượng khác so với đối tượng đầu, chẳng hạn như bằng cách chỉ sử dụng các hình ảnh có chứa nhiều đối tượng khác nhau. Biện pháp này hoạt động tốt vì các đối sánh chính xác cần phải có số lượng đáng kể những điểm hàng xóm gần nhất hơn so với đối sánh không chính xác để đạt được đối sánh đáng tin cậy. Đối với đối sánh sai, có thể sẽ có một số lượng đối sánh sai khác trong khoảng cách tương tự do chiều cao của không gian đặc trưng.



Hình 2.11: Tỷ lệ khoảng cách từ điểm điểm lân cận tới điểm kế tiếp

Hình 2.11 cho thấy giá trị của biện pháp này đối với dữ liệu hình ảnh thực tế. Hàm mật độ xác suất cho các đối sánh chính xác và không chính xác được thể hiện trong trực tỷ lệ gần nhất với điểm láng giềng gần nhất thứ hai của mỗi keypoint. Đối sánh hàng xóm gần nhất là một kết hợp chính xác có một PDF (probability of distance from) mà tập trung tại một tỷ lệ thấp hơn nhiều so với các đối sánh không chính xác. Để thực hiện nhận dạng đối tượng, ta lược bỏ tất cả các đối sánh trong đó tỷ lệ khoảng cách lớn hơn 0,8, trong đó loại bỏ 90% trong những đối sánh sai và loại bỏ ít hơn 5% trong những đối sánh chính xác. Hình vẽ này được tạo ra bằng cách kết hợp các hình ảnh với tỉ lệ ngẫu nhiên và thay đổi hướng, xoay chiều sâu 30 độ và thêm 2% nhiễu hình ảnh đối với một cơ sở dữ liệu của 40.000 Keypoint.

2.1.7.2. Hiệu quả của việc đánh số các điểm láng giềng gần

Không có thuật toán nổi tiếng nào có thể xác định chính xác những điểm hàng xóm gần nhất của các điểm trong không gian mà hiệu hơn so với tìm kiếm vét cạn. Mô tả keypoint ta sử dụng một vector đặc trưng 128 chiều và các thuật toán tốt nhất chẳng hạn như cây kd (Friedman et al., 1977) sẽ nhanh hơn so với tìm kiếm vét cạn trong không gian khoảng 10 chiều (hoặc

hơn). Do đó, ta sử dụng một thuật toán gần đúng, gọi là thuật toán Best-Bin-First (BBF) (Beis và Lowe, 1997). Thuật toán trả về điểm láng giềng gần nhất với xác suất cao.

Các thuật toán BBF sử dụng thứ tự tìm kiếm đã được chỉnh sửa cho thuật toán cây kd vì thế các vùng không gian đặc trưng được tìm trong các trật tự khoảng cách gần nhất của nó từ vị trí truy vấn, tìm kiếm ưu tiên này yêu cầu sử dụng đầu tiên được kiểm tra bởi Arya và Mount (1993), họ cung cấp nghiên cứu sâu về việc tính toán các thuộc tính (Arya et al., 1998). Việc tìm kiếm theo trật tự đòi hỏi việc sử dụng một hàng đợi ưu tiên dựa trên heap để xác định về hiệu quả của lệnh tìm kiếm. Một câu trả lời gần đúng có thể thực hiện với chi phí thấp bằng cách cắt đứt tìm kiếm sâu hơn nữa sau khi một số khu vực gần đó đã được tìm rồi. Trong việc thực hiện này, ta cắt đứt tìm kiếm sau khi kiểm tra lần đầu với 200 điểm láng giềng gần. Đối với một cơ sở dữ liệu của 100.000 keypoint, ta sẽ tăng tốc thuật toán tìm kiếm láng giềng gần nhất bằng cách tăng độ lớn gấp đôi và kết quả cho thấy sai số không quá 5% các đối sánh đúng.

2.1.7.3. Cụm biến đổi Hough

Để tối đa hóa hiệu suất của nhận dạng đối tượng cho các đối tượng nhỏ hoặc khả năng bết tắc cao, ta xác định các đối tượng với số lượng ít nhất có thể các đối sánh thuộc tính. Ta đã biết rằng việc nhận dạng là đáng tin cậy khi có 3 thuộc tính. Một hình ảnh chuẩn chứa 2.000 hoặc nhiều thuộc tính có thể đến từ nhiều đối tượng khác nhau và có sự lộn xộn nền. Trong khi kiểm tra tỷ lệ khoảng cách được mô tả trong Phần 7.1 đã cho phép chúng ta loại bỏ nhiều đối sánh sai phát sinh từ một nền lộn xộn, điều này không loại bỏ các đối sánh từ các đối tượng có giá trị khác, và chúng ta thường vẫn cần phải xác định tập con các đối sánh đúng có chứa ít hơn 1% inliers trong số 99%. Nhiều phương pháp nổi tiếng như RANSAC hoặc phương pháp tính trung bình nhỏ nhất của Squares hoạt động kém khi số phần trăm inliers rơi xuống thấp hơn 50%. May mắn thay, có thể thu được hiệu năng tốt hơn bằng cách phân nhóm các thuộc tính trong không gian bằng cách sử dụng biến đổi Hough (Hough, 1962; Ballard, 1981; Grimson 1990).

Mỗi keypoint ta sẽ đặc tả bằng 4 thông số: vị trí 2D, tỉ lệ và hướng và đối sánh mỗi keypoint trong cơ sở dữ liệu có một đặc tả về các thông số của keypoint liên quan tới hình ảnh huấn luyện đã được tìm thấy. Do đó, chúng ta có thể tạo ra một biến đổi Hough để dự đoán vị trí, hướng, và tỉ lệ từ giả thuyết đối sánh. Dự đoán này có thể bị sai sót nhiều do sự biến đổi tương đối bởi 4 thông số xấp xỉ 6 độ trong không gian tự do choi mỗi đối tượng 3D và cũng không lý giải cho bất kỳ sự biến dạng nào. Do đó, ta sử dụng kích cỡ mỗi vùng rộng 30 độ để gán hướng, hệ số 2 cho tỉ lệ, và tối đa gấp 0,25 lần kích thước ảnh huấn luyện (bằng cách sử dụng tỉ lệ dự đoán) cho vị trí. Để tránh những vấn đề về ranh giới phân chia vùng, mỗi đối sánh

keypoint dùng cho 2 vùng gần nhất ở mỗi hướng, tổng cộng 16 mục cho mỗi giả thuyết và tiếp tục mở rộng phạm vi tư thế.

2.1.7.4. Giải pháp cho các thông số Affine

Các biến đổi Hough được sử dụng để xác định tất cả các cụm có ít nhất 3 mục trong một bin. Mỗi cụm như vậy sau đó tùy thuộc vào một thủ tục xác định hình học trong đó một giải pháp bình phương nhỏ nhất được thực hiện đối với các thông số Affine tốt nhất liên quan đến hình ảnh huấn luyện cho hình ảnh mới. Một biến đổi Affine chính xác cho vòng quay 3D của một bề mặt phẳng dưới phép chiếu trực giao, nhưng sự thiếu chính xác có thể xảy ra khi quay 3D của đối tượng không phẳng. Tuy nhiên, một giải pháp ma trận cơ bản đòi hỏi ít nhất 7 điểm phù hợp so với chỉ cần 3 cho các giải pháp Affine và trong thực tế đòi hỏi nhiều hơn các đối sánh mới ổn định tốt. Ta muốn thực hiện nhận dạng với ít nhất là 3 đối sánh thuộc tính, vì vậy giải pháp Affine cung cấp một điểm khởi đầu tốt hơn và ta có thể khoanh vùng cho các lỗi trong xấp xỉ Affine bằng cách cho phép các lỗi còn sót lớn. Đối với các ví dụ điển hình của các đối tượng 3D được sử dụng trong bài báo này, một giải pháp Affine hoạt động tốt vì ta cho phép các lỗi còn sót lại lên đến 0,25 lần so với dự kiến. Một biến đổi Affine của một điểm mô hình $[x \ y]$ T đến một điểm ảnh $[uv]$ T có thể được viết như:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad (2.17)$$

Khi mô hình biến đổi là $[t_x \ t_y]^T$ và affine luân chuyển, tỉ lệ và độ căng được thể hiện bởi các thông số m_i . Muốn giải quyết cho các thông số chuyển đổi, phương trình trên có thể được viết lại để chèn các ẩn số vào một vector cột:

$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ \dots & & \dots & & & \end{bmatrix} \begin{bmatrix} m \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ \vdots \\ \vdots \end{bmatrix} \quad (2.18)$$

Phương trình này cho thấy một đối sánh duy nhất, tuy nhiên có nhiều đối sánh hơn nữa có thể được thêm vào, với mỗi đối sánh có ít nhất 2 hàng cho các ma trận đầu tiên và cuối cùng. Ít nhất 3 đối sánh là cần thiết cho một giải pháp.

Ta có thể viết hệ thống tuyến tính này như.

$$Ax = b \quad (2.19)$$



Hình 2.12: Ví dụ minh họa về thuật toán SIFT

Các giải pháp bình phương nhỏ nhất cho các tham số x có thể được xác định bằng cách giải quyết các phương trình tương đương:

$$x = [A^T A]^{-1} A^T b \quad (2.20)$$

Công thức trên làm tối thiểu tổng bình phương của khoảng cách từ vị trí mô hình dự đoán đến các địa điểm tương ứng trong hình ảnh. Trong cách tiếp cận này bình phương nhỏ nhất có thể dễ dàng được mở rộng để giải quyết cho kiểu 3D (Lowe, 1991).

Các giá trị ngoại lai bây giờ có thể được loại bỏ bằng cách kiểm tra mối liên hệ giữa các thuộc tính hình ảnh và mô hình. Để thêm phần chính xác thì áp dụng giải pháp bình phương nhỏ nhất, bây giờ ta yêu cầu mỗi đối sánh phải ràng buộc trong vòng nửa phạm vi lõi đã được sử dụng cho các thông số trong biến đổi Hough. Nếu ít hơn 3 điểm còn lại sau khi loại bỏ giá trị ngoại lai, sau đó đối sánh bị loại bỏ. Sau đó các giải pháp bình phương nhỏ nhất được giải quyết với các điểm còn lại, và quá trình lặp đi lặp lại.

Phương pháp này lần đầu tiên tính toán số lượng dự kiến của các đối sánh sai với mô hình đặt ra. Ta chấp nhận một mô hình nếu xác suất cuối cùng cho một luận điểm đúng là lớn hơn 0,98. Đối với các đối tượng mà ở các khu vực nhỏ của hình ảnh, 3 thuộc tính có thể là đủ để nhận dạng chính xác. Đối với các đối tượng lớn bao trùm một hình ảnh rất nhiều kết cấu, số lượng dự kiến của các đối sánh giả là cao hơn, và có thể cần đến đối sánh 10 thuộc tính.

2.2. Thuật toán SURF (Speeded Up Robust Features)

2.2.1. Giới thiệu thuật toán SURF

Khi làm việc với các đặc tính cục bộ, vấn đề đầu tiên cần phải được giải quyết là mức độ yêu cầu của bất biến. Điều này dựa trên sự biến dạng hình học và trắc quang mà bản thân nó

được xác định bằng khả năng bất biến trong điều kiện hướng nhìn thay đổi. Luận văn tập trung vào bộ phát hiện và bộ mô tả bất biến về tỷ lệ và góc xoay. Điều này dường như mang đến một sự kết hợp tốt giữa các đặc tính phức tạp và rộng lớn và thường xảy ra sự biến dạng. Lowe đã viết sự phức tạp của các thuộc tính bất biến thường có một tác động tiêu cực đến tính bền vững của ảnh trừ khi thay đổi hướng nhìn thực sự lớn. Trong một số trường hợp, thậm chí bất biến quay có thể được bỏ qua, kết quả là chỉ một phiên bản mô tả bất biến mà ta gọi là U-SURF.

2.2.2. Nghiên cứu liên quan

Bộ phát hiện điểm quan tâm được sử dụng rộng rãi nhất có lẽ là bộ dò góc Harris được đề xuất lại năm 1988 dựa trên các giá trị riêng của ma trận cấp 2. Tuy nhiên, góc Harris không bất biến tỉ lệ. Lindeberg giới thiệu một khái niệm về việc lựa chọn tỉ lệ tự động, điều này cho phép phát hiện các điểm quan tâm trong một hình ảnh, mỗi điểm đều có đặc trưng riêng của nó. Ông đã thử nghiệm với cả hai định thức của ma trận Hessian cũng như Laplacian (tương ứng với các dấu vết của ma trận Hessian) để phát hiện cấu trúc blob-like. Mikolajczyk và Schmid đã cải tiến phương pháp này và tạo ra một bộ dò mạnh với độ lặp lại cao và họ đặt tên là Harris-Laplace và Hessian-Laplace.

Bằng cách nghiên cứu các máy dò hiện tại và các so sánh được công bố ta có thể kết luận rằng dò dựa trên cơ sở Hessian là ổn định hơn nhiều hơn so với dò dựa trên Harris. Sử dụng cách xác định của ma trận Hessian có vẻ tốt hơn vết của nó (Laplacian) vì nó khả năng lọc kéo dài kém, cấu trúc cục bộ kém. Ngoài ra xấp xỉ như DoG thể mang lại tốc độ tốt với một chi phí thấp cho những phần dò kém chính xác.

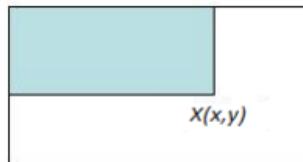
Bộ mô tả thuộc tính có số lượng lớn các mô tả đã được đề xuất như dẫn xuất Gaussian, bất biến thời điểm và các thuộc tính phức tạp, điều chỉnh bộ lọc, các thuộc tính cục bộ dựa trên từng giai đoạn, phân phối mô tả đại diện của thuộc tính tỉ lệ nhỏ trong khu vực điểm quan tâm. Sau này Lowe đã chứng minh một cách lọc tốt hơn những cách khác. Thực tế đã chứng minh rằng họ nắm bắt một số lượng đáng kể các thông tin về mô hình cường độ không gian trong khi tại cùng một thời điểm có biến dạng nhỏ hoặc lỗi cục bộ hóa. Bộ mô tả SIFT tính toán một biểu đồ của hướng gradient địa phương xung quanh các điểm quan tâm và lưu trữ trong một vector 128 chiều và được chia thành các ngăn (8 ngăn cho hướng và mỗi 4×4 ngăn vị trí).

Nhiều cách lọc khác nhau trên lược đồ cơ bản đã được đề xuất. Ke và Sukthankar áp dụng PCA vào hình ảnh gradient. Ở đây PCA-SIFT là một mô tả 36 chiều và nó đối sánh rất nhanh, nhưng cách này cũng ít đặc biệt hơn SIFT trong một nghiên cứu so sánh bởi Mikolajczyk et al. Và khả năng tính toán chậm làm giảm hiệu quả so khớp. Các tác giả đã đề xuất một biến thể của SIFT gọi là GLOH. Tuy nhiên GLOH có chi phí cao hơn.

Bộ mô tả SIFT vẫn có vẻ là mô tả hấp dẫn nhất đối với sử dụng thực tế và do đó cũng được sử dụng rộng rãi nhất hiện nay. Nó đặc biệt và tương đối nhanh vì thế SIFT rất quan trọng cho các ứng dụng trên mạng.

Máy dò SURF là máy dò được dựa vào ma trận Hessian nhưng sử dụng một xấp xỉ cơ bản giống như DoG là một máy dò dựa trên cơ sở Laplacian. Nó dựa trên ảnh tích hợp để làm giảm thời gian tính toán và do đó chúng ta gọi nó là phát hiện 'Fast-Hessian'. Mặt khác, phương pháp này còn mô tả sự phân bố của các đặc trưng Haar-wavelet trong khu vực điểm quan tâm. Phương pháp này khai thác những hình ảnh thích hợp để cải tiến tốc độ. Hơn nữa, việc chỉ sử dụng kích thước 64 chiều nên giảm thời gian cho tính toán và đối sánh và tăng tính ổn định. Luận văn cũng trình bày một bước đánh chỉ số mới dựa trên các ký hiệu của Laplacian, làm tăng tốc độ đối sánh và sự vững mạnh của các mô tả.

Ảnh tích hợp được đưa ra bởi Viola và Jones là một mảng 2 chiều với kích thước bằng kích thước của ảnh cần tính các đặc trưng, mỗi phần tử của mảng này được tính bằng cách tính tổng của điểm ảnh phía trên (dòng-1) và bên trái (cột-1) của nó. Bắt đầu từ vị trí trên, bên trái đến vị trí dưới, phải của ảnh, việc tính toán này đơn thuần chỉ dựa trên phép cộng số nguyên đơn giản, do đó tốc độ thực hiện rất nhanh.



$$I_{\Sigma}(x) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j)$$

Với I_{Σ} được tính toán thì chỉ mất bốn phép cộng để tính toán tổng các cường độ bất kỳ của một hình chữ nhật độc lập với kích thước của nó.

Ảnh tích hợp cho phép thực hiện nhanh chóng các loại bộ lọc nếp cuộn. Sự tham gia của một hình ảnh tích hợp $I_{\Sigma}(x)$ tại một vị trí $x = (x, y)$ đại diện cho tổng của tất cả các điểm ảnh trong hình ảnh đầu vào I của một khu vực hình chữ nhật được hình thành bởi các điểm x với $I_{\Sigma}(x) = \sum_{i=0}^{i < x} \sum_{j=0}^{j < y} I(i, j)$. Khi tính toán I_{Σ} chỉ phải mất bốn phép tính để tính tổng các cường độ phía trên bên phải khu vực hình chữ nhật độc lập với kích thước của nó.

2.2.3. Bộ dò Fast- Hessian

Bộ dò này căn cứ trên ma trận Hessian vì nó có hiệu suất tốt trong tính toán thời gian và độ chính xác. Tuy nhiên thay vì sử dụng một phát hiện Hessian cho việc lựa chọn vị trí và tỉ lệ (như đã được thực hiện trong các máy dò Laplace -Hessian), ở đây ta dựa vào các yếu tố quyết

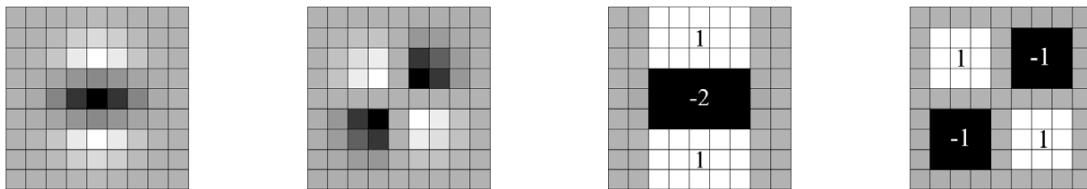
định của Hessian. Cho một điểm $x=(x,y)$ trong một hình I, các ma trận Hessian $H(x,\sigma)$ tại x ở tỉ lệ σ là được định nghĩa như sau.

$$H(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix} \quad (2.21)$$

Trong đó $L_{xx}(x, \sigma)$ là nếp cuộn của hàm Gaussian và đạo hàm cấp 2 $\frac{\partial^2}{\partial x^2} g(\sigma)$ của ảnh I tại điểm x , và tương tự cho $L_{xy}(x, \sigma)$ và $L_{yy}(x, \sigma)$.

Gaussian là tối ưu cho việc phân tích không gian tỉ lệ, như trong [24], tuy nhiên trong thực tế, các Gaussian cần được được phân đoạn và cắt (hình 1 nửa trái) và ngay cả với lọc Gaussian rằng cưa vẫn xảy ra ngay sau khi các hình ảnh kết quả được lấy từ mẫu. Ngoài ra, các thuộc tính không có cấu trúc mới có thể xuất hiện trong khi độ phân giải giảm đã được chứng minh trong trường hợp 1D nhưng không áp dụng trong trường hợp có liên quan 2D. Do đó tầm quan trọng của Gaussian dường như đã phần nào đánh giá quá cao trong vấn đề này. Vì lọc Gaussian là không lý tưởng trong mọi trường hợp nên Lowe đã tính xấp xỉ LoG với các hộp lọc sâu hơn (hình 1 nửa bên phải). Hình thứ 2 là dẫn xuất Gaussian gần đúng và có thể được đánh giá rất nhanh do sử dụng hình ảnh tích hợp độc lập với kích thước. Kết quả là hai phương pháp có hiệu suất tương đương với cách phân đoạn và cắt Gaussian.

Hộp lọc 9×9 trong hình 1 là gần đúng cho các xấp xỉ Gaussian với $\sigma = 1.2$ và đại diện cho tỉ lệ thấp nhất (tức là độ phân giải không gian cao nhất). Ta biểu diễn xấp xỉ bằng D_{xx} , D_{yy} , và D_{xy} .



Hình 2.15: Sự biến đổi của hàm Gaussian theo từng giai đoạn

Các trọng số áp dụng cho các khu vực hình chữ nhật được giữ đơn giản để tính toán hiệu suất nhưng chúng ta cần phải cân bằng trọng lượng của biểu thức xác định của Hessian với:

$$\frac{|L_{xy}(1.2)|_F |D_{xx}(9)|_F}{|L_{xx}(1.2)|_F |D_{xy}(9)|_F} = 0.912... \square 0.9 \quad (2.22)$$

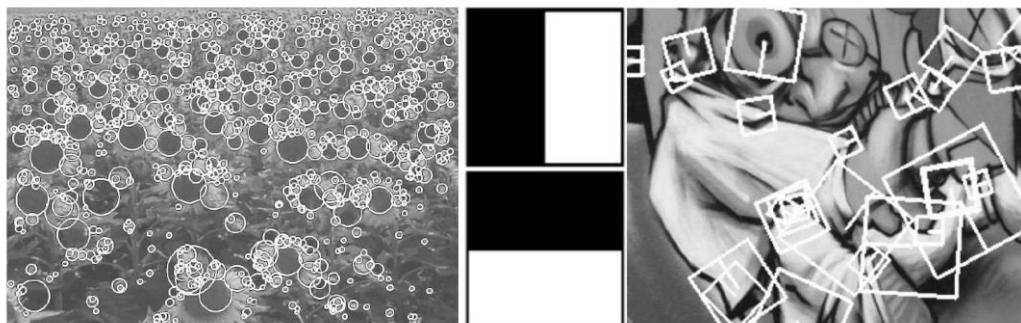
Trong đó $|x|_F$ là định mức Frobenius và

$$\det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \quad (2.23)$$

Hơn nữa những bộ lọc phản hồi là bình thường đối với các kích thước mặt nạ. Điều này đảm bảo một hằng số Frobenius liên tục cho bất kỳ kích thước bộ lọc nào. Không gian tỉ lệ thường được thực hiện như hình ảnh kim tự tháp. Các hình ảnh liên tục được làm mịn với một Gaussian và sau đó các mẫu con đạt được một mức độ cao hơn của kim tự tháp. Sau đó ta dùng các bộ lọc và hình ảnh tích hợp, ta không cần phải lặp đi lặp lại cùng bộ lọc với đầu ra của lớp bộ lọc trước nhưng thay vào đó có thể áp dụng các bộ lọc có kích thước bất kỳ trên ảnh gốc và thậm chí thực hiện song song bộ lọc. Vì vậy không gian tỉ lệ được phân tích bằng tỉ lệ trên kích thước bộ lọc hơn là việc lặp đi lặp lại làm giảm kích thước ảnh. Đầu ra của bộ lọc 9×9 trên được coi là lớp tỉ lệ ban đầu mà ta đề cập như tỉ lệ $s = 1,2$ (tương ứng với các xấp xỉ Gaussian với $\sigma = 1,2$). Các lớp sau đây thu được bằng lọc hình ảnh với mặt nạ dần dần lớn hơn có tính đến tính chất rời rạc của ảnh tích hợp và các cấu trúc đặc biệt của bộ lọc. Đặc biệt kết quả này trong bộ lọc kích thước $9 \times 9, 15 \times 15, 21 \times 21, 27 \times 27$, vv Ở tỉ lệ lớn hơn, bước giữa các kích thước bộ lọc liên tiếp cũng nên tỉ lệ phù hợp. Do đó, đối với mỗi octave mới, ta nên tăng kích thước bộ lọc gấp đôi (từ 6-12 đến 24). Đồng thời, khoảng lấy mẫu để tách các điểm quan tâm có thể được tăng lên gấp đôi là phù hợp.

Tỉ lệ bộ lọc vẫn là hằng số khi thay đổi tỉ lệ, các dãy xuất xấp xỉ Gaussian vẫn phù hợp. Vì vậy, ví dụ lọc 27×27 của ta tương ứng với $\sigma = 3 \times 1,2 = 3,6 = s$. Hơn nữa, các định mức Frobenius vẫn không đổi cho bộ lọc, chúng đã được tỉ lệ một cách bình thường.

Để khoanh vùng các điểm quan tâm trong ảnh và trên những tỉ lệ, vùng lảng giềng $3 \times 3 \times 3$ được áp dụng. Cực đại của định thức ma trận Hessian sau đó nội suy về tỉ lệ và không gian ảnh với các phương pháp của Brown et al. Tỉ lệ không gian nội suy là đặc biệt quan trọng trong trường hợp này và sự khác biệt của tỉ lệ giữa các lớp đầu tiên của mỗi octave là tương đối lớn. Hình 2.16 (trái) cho thấy một ví dụ về những điểm quan tâm phát hiện bằng cách sử dụng bộ đồ 'Fast-Hessian'.



Hình 2.16: Phát hiện điểm Keypoint của thuật toán SURF

2.2.4. Bộ mô tả SURF

Hiệu quả của SIFT tốt hơn nhiều so với các mô tả khác. Sự pha trộn của các thông tin sơ sài cục bộ hóa và sự phân bố của gradient liên quan đến các thuộc tính đường như mang lại sức mạnh đặc biệt tốt tránh ảnh hưởng của lỗi cục bộ hóa về tỉ lệ hoặc không gian. Việc sử dụng thế mạnh và hướng của gradient làm giảm ảnh hưởng của thay đổi trắc quang.

Bộ mô tả SURF được đề xuất dựa trên tính chất tương tự với độ phức tạp giảm dần. Bước đầu tiên là gán hướng dựa trên các thông tin từ một khu vực hình tròn xung quanh điểm quan tâm. Sau đó, chúng ta xây dựng một khu vực hình vuông phù hợp với hướng lựa chọn và trích xuất các mô tả SURF từ nó. Ngoài ra một đề xuất phiên bản của mô tả (U-SURF) không bát biến khi xoay hình ảnh nhưng tính toán nhanh hơn và phù hợp hơn cho các ứng dụng máy ảnh.

2.2.4.1. Gán hướng

Để bắt biến khi xoay ta xác định một hướng để gán cho các điểm quan tâm. Với mục đích đó, đầu tiên ta tính toán phản ứng Haar-wavelet trong hướng x và y thể hiện trong hình 2 và trong một vùng lảng giềng hình tròn bán kính $6s$ xung quanh các điểm quan tâm với s là tỉ lệ mà tại đó các điểm quan tâm đã được phát hiện. Ngoài ra các bước lấy mẫu là tỉ lệ phụ thuộc vào việc chọn s . Để phù hợp với phần còn lại cũng có những phản ứng Wavelet được tính theo tỉ lệ hiện tại s . Theo đó, ở tỉ lệ cao kích thước của wavelets là lớn. Do đó, ta sử dụng lại hình ảnh tích hợp để lọc nhanh. Chỉ có sáu thao tác cần thiết để tính toán các phản ứng trong hướng x hoặc y ở tỉ lệ bất kỳ. Chiều dài cạnh của Wavelets là $4s$. Sau khi phản ứng Wavelet được tính toán và trọng số Gaussian ($\sigma = 2.5s$) tập trung tại các điểm quan tâm, các phản ứng được biểu diễn như là vectơ trong một không gian và đáp ứng mạnh với chiều ngang dọc theo trực hoành và đáp ứng mạnh theo trực thẳng đứng dọc theo hệ tọa độ. Các hướng tọa độ được ước tính bằng cách tính tổng của tất cả các phản ứng trong một cửa sổ định hướng trượt một góc $\pi/3$. Các phản ứng theo chiều ngang và thẳng đứng bên trong cửa sổ được cộng lại. Hai phản ứng tổng hợp sau đó tạo ra một vector mới. Vector nào dài nhất thì hướng nó là hướng đến các điểm quan tâm. Kích thước của cửa sổ trượt là một tham số đã được lựa chọn thí điểm. Kích thước nhỏ trên vùng đáp ứng Wavelet chiếm ưu thế, kích thước lớn mang lại vector độ dài lớn nhất. Cả hai kết quả là một hướng không ổn định của khu vực quan tâm. Các U-SURF bỏ qua bước này.

2.2.4.2. Thành phần của mô tả

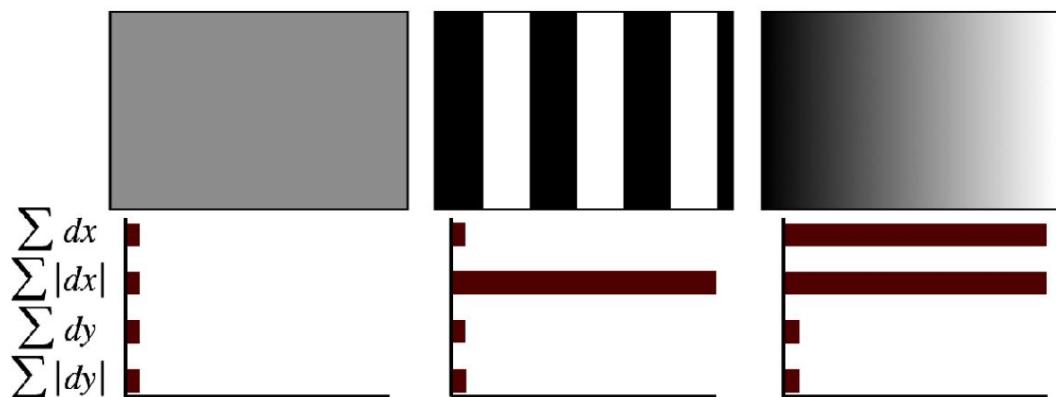
Đối với việc khai thác các mô tả, bước đầu tiên bao gồm xây dựng một cấu trúc hình vuông tập trung vào các điểm quan tâm và định hướng theo hướng lựa chọn trong phần trước. Các chuyển đổi này là không cần thiết. Kích thước của cửa sổ này là $20s$. Ví dụ về các khu vực hình vuông như

được minh họa trong Hình 2.17.

Khu vực này thường được chia ra nhỏ hơn 4×4 tiêu vùng vuông. Đối với từng tiêu vùng ta tính toán một vài thuộc tính đơn giản với không gian 5×5 điểm mẫu đồng đều. Vì lý do đơn giản, ta gọi dx là phản ứng Wavelet Haar theo hướng ngang và dy là phản ứng Wavelet Haar theo hướng thẳng đứng ($2s$ kích thước lọc). "Ngang" và "thẳng đứng" ở đây là khái niệm liên quan đến hướng điểm quan tâm được lựa chọn. Để tăng độ vững mạnh theo hướng biến dạng hình học và các lỗi cục bộ hóa, các phản ứng dx và dy trước hết được gán trọng số Gaussian ($\sigma=3.3s$) tập trung tại các điểm quan tâm.

Sau đó, các phản ứng dx và dy Wavelet được tổng hợp qua từng tiêu vùng và tạo thành một thực thể đầu tiên của tập hợp các mục cho vector đặc trưng. Để có được thông tin về sự phân cực của việc thay đổi cường độ ta cũng trích tổng các giá trị tuyệt đối của các phản ứng $|dx|$ và $|dy|$. Do đó, mỗi tiêu vùng có bốn chiều vector mô tả v , cơ cấu cường độ của nó $v = (\sum dx, \sum dy, \sum |dx|, \sum |dy|)$. Điều này dẫn đến một vector mô tả cho tất cả 4×4 tiêu vùng có độ dài 64. Các phản ứng Wavelet là bất biến đối với một sự ưu tiên trong vùng chiếu sáng (offset). Sự bất biến trong tương phản (một yếu tố tỉ lệ) đạt được bằng cách chuyển các mô tả vào một vector đơn vị.

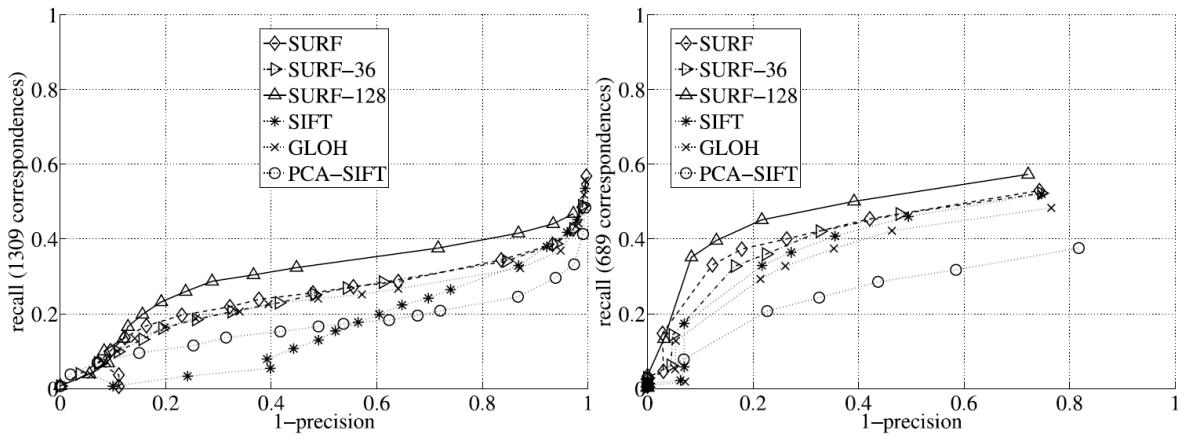
Hình 2.17 cho thấy các thuộc tính của các mô tả cho ba đặc trưng của các ảnh khác nhau về cường độ trong tiêu vùng. Để có được những mô tả SURF, ta đã thử nghiệm với ít thuộc tính hơn và nhiều hơn nữa Wavelet, và sử dụng Wavelets bậc cao PCA giá trị trung bình, vv.



Hình 2.17: Mô tả các đặc trưng của ảnh về cường độ

Khi đánh giá toàn diện để tìm cách thực hiện tốt nhất khi thay đổi số lượng các điểm mẫu và các tiêu vùng, các giải pháp phân chia 4×4 tiêu vùng cung cấp các kết quả tốt nhất. Mặt khác, mô tả ngắn với 3×3 tiêu vùng (SURF-36) thực hiện rất kém nhưng cho phép kết hợp rất nhanh và vẫn còn chấp nhận được so với các mô tả khác. Hình 2.18 cho thấy chỉ có một vài

trường hợp đúng trong số những kết quả so sánh.



Hình 2.18: So sánh các điểm Keypoint tìm được với phép dò nhanh Hessian

Khi thử nghiệm một phiên bản khác của bộ mô tả SURF khi có thêm một vài thuộc tính tương tự (SURF-128) với chi phí không đổi nhưng chia ra nhiều giá trị hơn nữa. Các giá trị d_x và $|d_x|$ được tính riêng cho $d_y < 0$ và $d_y \geq 0$. Tương tự như vậy, các giá trị của d_y và $|d_y|$ được chia ra theo các dấu hiệu của d_x , qua đó tăng gấp đôi số lượng các thuộc tính. Bộ mô tả này đặc biệt hơn và không chậm hơn trong tính toán.

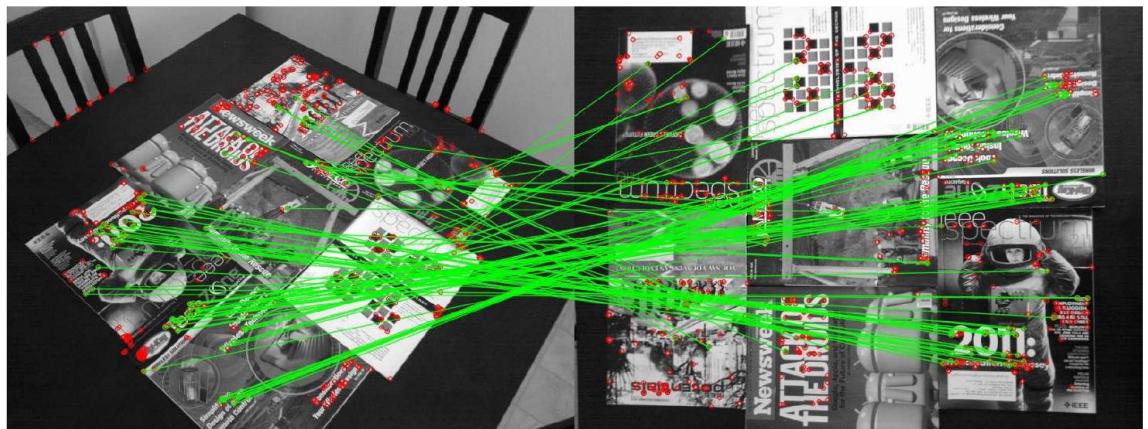
Trong hình 2.18, các lựa chọn tham số được so sánh với các tiêu chuẩn 'Graffiti', đó là thách thức lớn nhất khi ảnh bị xoay như trong mặt phẳng vòng xoay cũng như thay đổi độ sáng. Bộ mô tả mở rộng cho 4×4 tiêu vùng (SURF-128) là thực hiện tốt nhất. Ngoài ra, SURF hoạt động tốt và nhanh hơn trong xử lý.

Việc đổi sánh chỉ mục thực hiện rất nhanh bao gồm các dấu hiệu của Laplacian (tức là các dấu vết của ma trận Hessian) cho các điểm quan tâm cơ bản. Thông thường những điểm quan tâm được tìm thấy ở các cấu trúc blob-type. Các dấu hiệu của Laplacian phân biệt các đốm màu sáng trên nền tối từ hình ngược lại. Thuộc tính này có sẵn mà không mất thêm chi phí tính toán vì nó đã được tính toán trong giai đoạn phát hiện. Trong giai đoạn tiếp ta chỉ so sánh thuộc tính nếu chúng có cùng một độ tương phản. Do đó, thông tin tối thiểu này cho phép đổi sánh nhanh hơn và tăng hiệu suất.

2.3. Thuật toán ORB

Thuộc tính ORB đề xuất được xây dựng trên các bộ dò keypoint nổi tiếng FAST và gần đây là bộ mô tả BRIEF và gọi là ORB (Oriented Fast and Rotated BRIEF). Cả hai công nghệ này rất hấp dẫn vì cách thực hiện tốt của nó và chi phí thấp. Trong luận văn này, ta giải quyết một số hạn chế của những công cụ liên quan đến SIFT, đáng kể nhất là thiếu bát biến trong phép quay BRIEF. Cụ thể là:

- Thêm vào FAST và gán hướng chính xác cho các thành phần của FAST
- Các tính toán hiệu quả thuộc tính hướng BRIEF.
- Phân tích phương sai và tương quan của thuộc tính hướng BRIEF.
- Một phương pháp học giảm tương ứng thuộc tính BRIEF dưới bất biến quay, dẫn đến thực hiện tốt hơn các ứng dụng láng giềng gần.



Hình 2.21: Ví dụ về kết quả đối sánh ảnh sử dụng thuật toán ORB

Để hiểu hơn về ORB ta thực hiện các thí nghiệm kiểm tra các thuộc tính của ORB liên quan đến SIFT và SURF, cả khả năng đối sánh thô và hiệu suất trong các ứng dụng đối sánh hình ảnh. Kiểm nghiệm hiệu quả của ORB bằng cách thực hiện một ứng dụng theo dõi trên bản vá trên điện thoại thông minh. Một lợi ích nữa của ORB là nó là miễn phí trong khi SIFT và SURF thì không.

2.3.1. Nghiên cứu liên quan

Keypoint FAST và các biến thể của nó là phương pháp lựa chọn việc tìm kiếm keypoint trong các hệ thống thời gian thực phù hợp với thuộc tính thị giác, ví dụ theo dõi vết trên bản đồ. Để tìm góc hợp lý cho keypoint cần tăng cường tỉ lệ của lược đồ kim tự tháp và trong trường hợp này bộ lọc góc Harris lọc loại bỏ đường biên.

Nhiều phép dò keypoint bao gồm một thao tác gán hướng (SIFT và SURF là hai ví dụ nổi bật) nhưng FAST thì không. Có nhiều cách khác nhau để mô tả hướng của một keypoint và có nhiều cách liên quan đến biểu đồ của các tính toán gradient, ví dụ như trong SIFT và sự tương đối của mô hình khối trong SURF. Những phương pháp này là một trong hai yêu cầu tính toán hoặc trong trường hợp của SURF thì tính toán xấp xỉ kém. Các tài liệu tham khảo của Rosin đưa ra một phân tích khác nhau để đo hướng của các góc và sử dụng kỹ thuật trọng tâm. Không giống như các thao tác gán hướng trong SIFT, có thể có nhiều giá trị trên một keypoint

duy nhất, các thao tác trọng tâm của Rosin cho một kết quả duy nhất.

Mô tả BRIEF là một mô tả thuộc tính mới có sử dụng các kiểm tra nhị phân đơn giản giữa các điểm ảnh trong hình ảnh của một bản vá nhẫn. Hiệu quả của nó tương tự như SIFT ở nhiều khía cạnh, trong đó rất tốt với ánh sáng, làm mờ và hướng nhìn méo. Tuy nhiên, nó rất nhạy cảm với chuyển động quay trong mặt phẳng.

BRIEF phát triển từ nghiên cứu có sử dụng các kiểm tra nhị phân để huấn luyện một tập hợp các cây phân loại. Một tập huấn luyện 500 hoặc nhiều keypoint tiêu biểu, cây có thể được dùng trả về dấu hiệu cho bất kỳ keypoint đặc biệt nào. Bằng cách tương tự ta tìm kiếm các kiểm tra nhị phân ít nhạy với hướng. Lớp phương thức kinh điển để tìm kiếm tra không tương ứng là PCA, ví dụ như nó hiển thị PCA cho SIFT có thể giúp bỏ một lượng lớn thông tin dư thừa. Tuy nhiên không gian kiểm tra nhị phân là quá lớn cho thực hiện PCA và tìm kiếm vét cạn được sử dụng thay thế.

2.3.2. oFAST: hướng của keypoint FAST

2.3.2.1. Bộ dò FAST

Ta bắt đầu dò điểm FAST trong bức ảnh. FAST lấy một tham số là cường độ giữa điểm tâm và những điểm trong hình tròn của tâm.

Ta sử dụng FAST-9 (bán kính tròn là 9) có kết quả thực hiện tốt nhất.

FAST không tạo ra phép đo góc mà ta phải tìm góc mà nó đáp ứng mạnh dọc theo các biên. Ta tiến hành nhúng góc đo Harris để đặt các keypoint FAST. Với đích là N keypoint đầu tiên chúng ta thiết lập các ngưỡng thấp, đủ để nhận được nhiều hơn N keypoint, sau đó đặt chúng theo phép đo Harris và chọn điểm đầu trong N điểm.

FAST không sinh ra các thuộc tính đa tỉ lệ mà sử dụng một kim tự tháp tỉ lệ của hình ảnh và tạo ra các thuộc tính FAST (được lọc bởi Harris) ở mỗi cấp của kim tự tháp.

2.3.2.2. Hướng của cường độ trọng tâm

Cường độ trọng tâm được giả định là cường độ của một góc được tương đương từ trung tâm của nó và vector này có thể được sử dụng để quy cho một hướng. Rosin định nghĩa một bản vá của phép quay như sau:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y) \quad (2.24)$$

và với những phép quay này chúng ta có thể tìm thấy các trọng tâm:

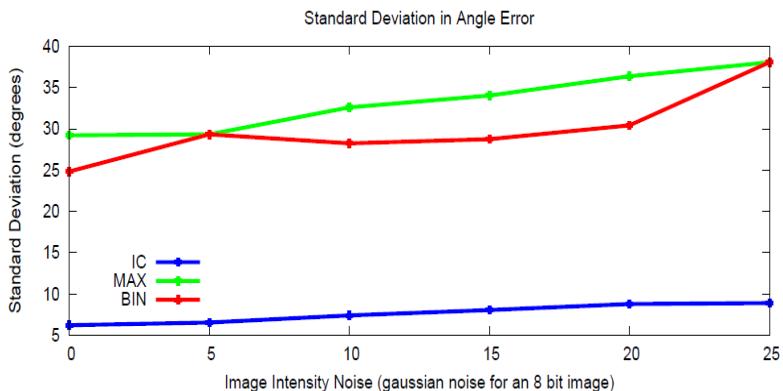
$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \quad (2.5)$$

Ta cấu trúc 1 vector từ tâm O của góc tới các trọng tâm O_c . Hướng của các bản vá đơn giản là:

$$\theta = a \tan 2(m_{01}, m_{10}) \quad (2.26)$$

Ở đây atan2 là phiên bản góc phần tư của arctan. Rosin đề cập lấy loại góc cho dù góc tối hoặc có ánh sáng. Tuy nhiên, mục đích của ta là có thể bỏ qua điều này miễn là biện pháp đo góc không liên quan đến loại góc.

Để cải tiến phép quay bất biến phép đo này ta chắc chắn phép quay được tính toán với X và Y còn lại trong phạm vi hình tròn bán kính r. Ta quan sát thử nghiệm và chọn r là kích thước bản vá, vì thế x và y chạy từ $[-r, r]$. Vì thế C tiếp cận O, phép đo trở thành không ổn định. Với góc Fast FAST ta thấy rằng nó hiếm khi gặp trường hợp này.



Hình 2.22: Độ thị cường độ nhiễu của ảnh.

2.3.3. Phép quay: rBRIEF

Phản báo cáo này tôi giới thiệu về một mô tả hướng BRIEF, sự tính toán hiệu quả của nó và nhược điểm của nó là kém hiệu quả với phép xoay. Để giải quyết vấn đề này ta phải tìm các kiểm tra nhị phân ít tương quan đẽ rBRIEF mô tả tốt hơn, sau đó so sánh với SIFT và SURF.

2.3.3.1. Hiệu quả của phép quay BRIEF

Tổng quan về BRIEF

Bộ mô tả BRIEF là một mô tả chuỗi bit của một bản vá hình ảnh được xây dựng từ một tập hợp các kiểm tra cường độ nhị phân. Với một bản vá hình ảnh được làm mịn p. Một thử nghiệm nhị phân t được xác định bởi:

$$\tau(p; x, y) := \begin{cases} 1: p(x) < p(y) \\ 0: p(x) \geq p(y) \end{cases} \quad (2.27)$$

Trong đó $p(x)$ là cường độ của p tại điểm x . Thuộc tính được định nghĩa như một vector của n kiểm tra nhị phân

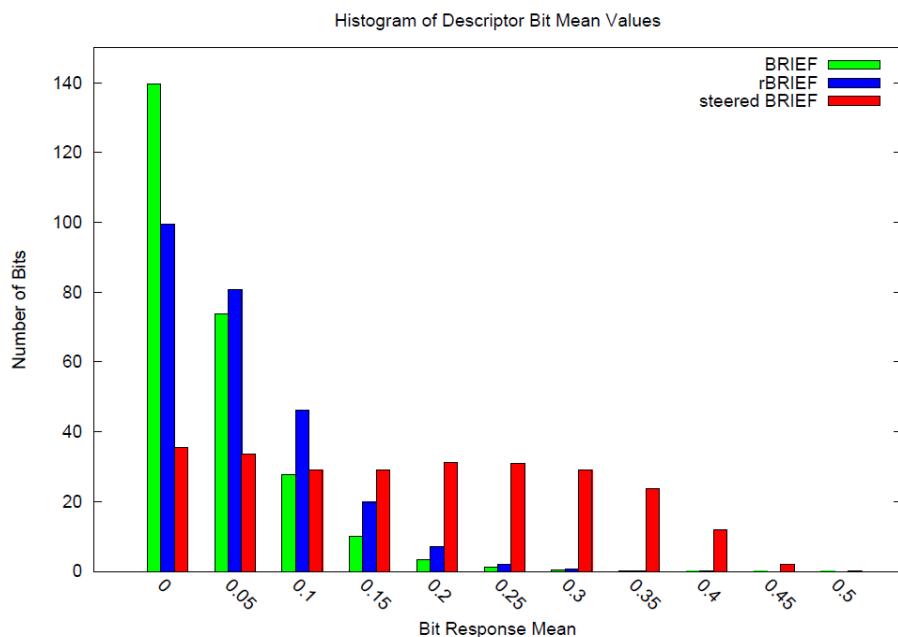
$$f_n(p) := \sum_{1 \leq i \leq n} 2^{i-1} \tau(p; x_i; y_i) \quad (2.28)$$

Ở đây tôi sử dụng một trong những cách biểu diễn tốt nhất là phân phối Gaussian quanh trung tâm của các bản vá và chọn chiều dài vector $n = 256$.

Điều quan trọng là làm mịn hình ảnh trước khi thực hiện các kiểm tra. Trong việc thực hiện này, việc làm mịn dùng một ảnh tích hợp và mỗi điểm kiểm tra là một cửa sổ con 5×5 của bản vá 31×31 .

Hướng BRIEF

Ta giả sử rằng BRIEF là bất biến trong chuyển động quay mặt phẳng. Việc thực hiện đối sánh của BRIEF giảm mạnh trong mặt phẳng quay một vài độ (xem Hình 2.23). Calonder đề nghị tính toán một mô tả BRIEF cho một tập hợp các phép quay và biến dạng cong của mỗi miếng vá



Hình 2.23.: Sự phân phối cân bằng các vector thuộc tính

Nhưng giải pháp này rõ ràng là tồn kém. Một phương pháp hiệu quả hơn là hướng BRIEF theo hướng của keypoint. Đối với bất kỳ thuộc tính nào của bộ n kiểm tra nhị phân ở vị trí $(x_i,$

y_i), định nghĩa các ma trận $2 \times n$

$$S = \begin{pmatrix} x_1, \dots, x_n \\ y_1, \dots, y_n \end{pmatrix} \quad (2.29)$$

Sử dụng bản vá hướng θ và ma trận xoay tương ứng R_θ ta xây dựng một "hướng" phiên bản S_θ của S :

$$S_\theta = R_\theta S \quad (2.30)$$

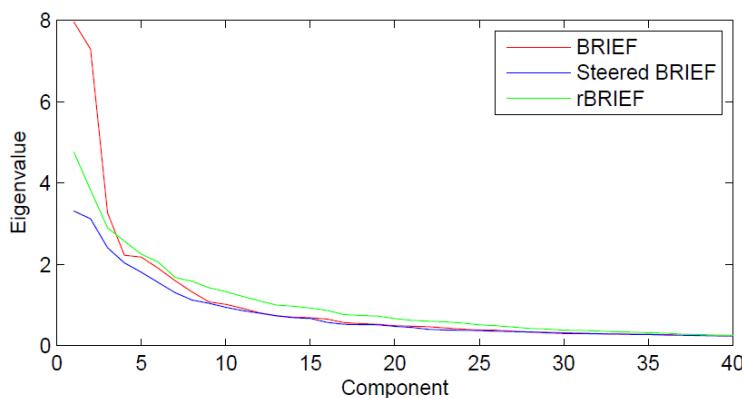
Bây giờ các thao tác hướng BRIEF trở thành

$$g_n(p, \theta) := f_n(p) | (x_i, y_i) \in S_\theta \quad (2.31)$$

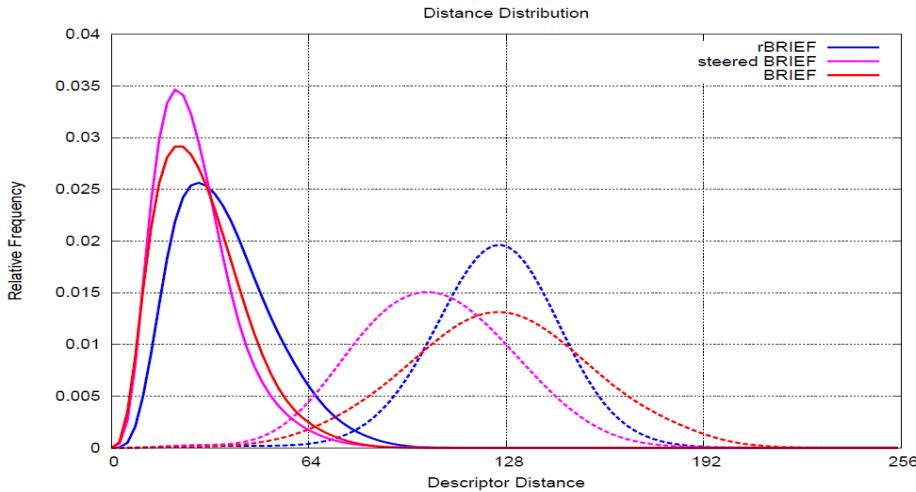
Sau đó rời rạc góc để có số gia $2\pi/30$ (12 độ) và xây dựng một bảng tra cứu của BRIEF. Hướng keypoint θ là nhất quán trên các hướng nhìn, các tập điểm chính xác của điểm S_θ sẽ được dùng để tính toán bộ mô tả của nó.

2.3.4. Biến thể và tương quan

Một trong những đặc tính hay của BRIEF là mỗi bit thuộc tính có một biến thể (phương sai) lớn và trung bình gần 0,5. Hình 2.23 cho thấy sự ảnh hưởng của giá trị trung bình cho bảng Gaussian BRIEF điển hình là 256 bít trên 100k keypoint mẫu. Một giá trị trung bình là 0,5 cho các mẫu sai số tối đa 0,25 cho một bit thuộc tính. Mặt khác, một BRIEF được gán hướng theo hướng keypoint, giá trị trung bình được thay đổi tới bảng phân phối đều hơn (Hình 2.24). Hướng góc keypoint xuất hiện đồng đều hơn để kiểm tra nhị phân.



Hình 2.24: Phân phối giá trị riêng trong phân ly PCA hơn 100k keypoint của ba vectơ thuộc tính: BRIEF, hướng BRIEF và rBRIEF



Hình 2.25: Khoảng cách phát hiện của các vector thuộc tính

Phương sai lớn làm cho một thuộc tính khác biệt hơn từ đó nó đáp ứng khác nhau với các đầu vào. Một mong muốn khác là phải có các kiểm tra không tương quan và kể từ đó mỗi lần kiểm tra sẽ kiểm tra được kết quả.

Để phân tích tương quan và phương sai của các kiểm tra trong các vector BRIEF chúng ta nhìn vào đáp ứng của 100k keypoint cho BRIEF và hướng BRIEF. Các kết quả được thể hiện trong hình 4. Sử dụng PCA trên các dữ liệu, chúng ta vẽ 40 giá trị riêng cao nhất (sau khi chuyển đổi về 2 bộ mô tả). Cả BRIEF và rBRIEF đều khởi tạo giá trị riêng ban đầu cao cho thấy mối tương quan giữa các kiểm tra nhị phân, về cơ bản tất cả các thông tin được chứa trong 10 hoặc 15 thành phần đầu tiên. rBRIEF có tầm quan trọng là biến thể thấp hơn, tuy nhiên khi các giá trị riêng là thấp hơn nên nó không khác biệt. Rõ ràng BRIEF với hướng ngẫu nhiên của các keypoint cho hiệu suất hoạt động tốt.

2.3.5. Phương pháp học cho các thuộc tính nhị phân

Để phục hồi những mất mát do biến thể trong BRIEF và để giảm sự tương quan giữa các kiểm tra nhị phân, chúng ta phát triển một phương pháp học cho việc lựa chọn một tập hợp con các kiểm tra nhị phân. Một chiến lược là sử dụng phương pháp PCA hoặc các kỹ thuật giảm chiều khác và bắt đầu từ một tập lớn các kiểm tra nhị phân, xác định 256 thuộc tính mới mà có phương sai cao không tương quan trên tập huấn luyện lớn. Tuy nhiên, khi các thuộc tính mới được tạo thành từ một số lượng lớn các kiểm tra nhị phân, chúng sẽ ít hiệu quả trong tính toán hướng BRIEF. Thay vào đó ta tiến hành tìm kiếm trong số tất cả các kiểm tra nhị phân có phương sai cao (có giá trị gần 0,5) và không tương quan.

Phương pháp này cụ thể như sau: Ta thiết lập một bộ huấn luyện 300k keypoint rút ra từ tập hình ảnh PASCAL2006. Sau đó liệt kê tất cả các kiểm tra nhị phân có thể được rút ra từ

bản vá 31×31 . Mỗi kiểm tra là một cặp 5×5 các cửa sổ con của các bản vá. Ta lưu ý chiều rộng của miếng vá $wp = 31$ và chiều rộng của các cửa sổ con kiểm tra là $wt = 5$, sau đó ta có $N = (wp - wt)2$ các cửa sổ phụ. Vì vậy ta có $N/2$ kiểm tra nhị phân. Sau đó ta loại bỏ các kiểm tra chồng nhau và ta kết thúc với $M = 205.590$ kiểm tra. Các bước thuật toán là:

1) Chạy mỗi kiểm tra đối với tất cả các bản vá huấn luyện.

2) Sắp thứ tự các kiểm tra theo khoảng cách của chúng từ một giá trị trung bình là 0,5 và tạo thành các vector T.

3) Tìm kiếm Greedy:

a) Đặt các kiểm tra đầu tiên vào vector kết quả R và lại xóa nó khỏi T.

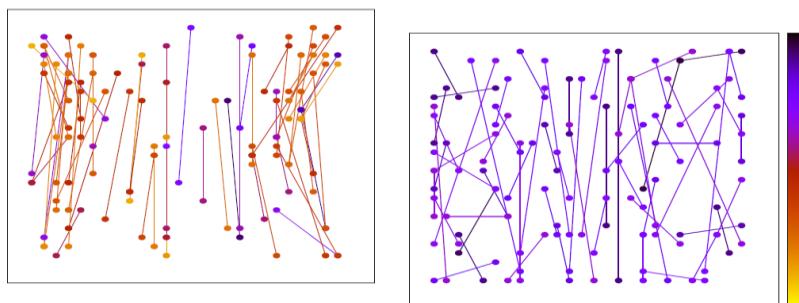
b) Thực hiện kiểm tra tiếp theo từ T, và so sánh nó với tất cả các kiểm tra trong R. Nếu tương quan tuyệt đối của nó lớn hơn một ngưỡng thì loại bỏ nó, ngược lại bổ sung nó vào R.

c) Lặp lại các bước trước đó cho đến khi có 256 kiểm tra trong R. Nếu có ít hơn 256, tăng ngưỡng cũ và thử lại.

Thuật toán này là một tìm kiếm tham lam cho một loạt các kiểm tra không tương quan với các giá trị trung bình gần 0,5. Kết quả được gọi là rBRIEF. rBRIEF có cải thiện đáng kể trong các phương sai và tương quan so với hướng. Các giá trị riêng của PCA cao hơn và chúng giảm rất nhanh.

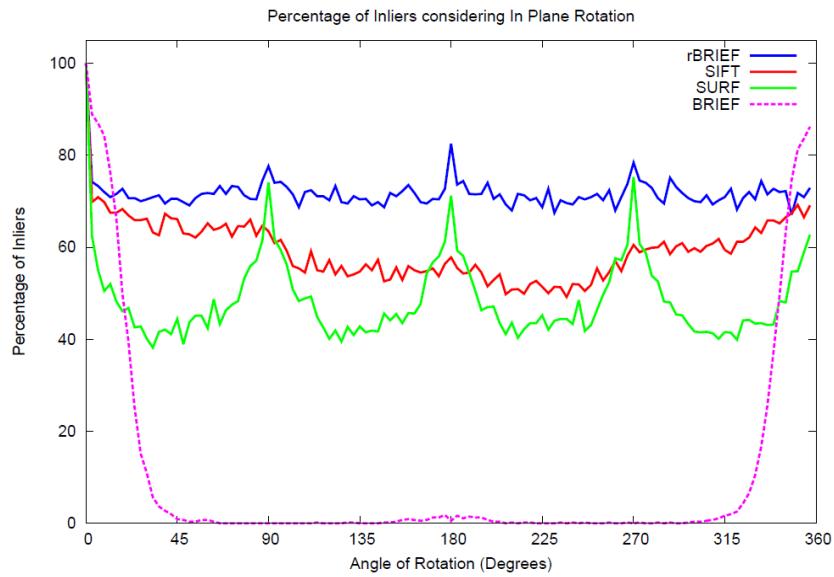
2.3.6. Đánh giá

Ta đánh giá sự kết hợp của oFAST và rBRIEF gọi là ORB sử dụng hai bộ dữ liệu: hình ảnh bị xoay trong mặt phẳng và thêm nhiều Gaussian và một tập dữ liệu thật của hình ảnh phẳng với vân ảnh được chụp từ nhiều hướng nhìn khác nhau. Đối với mỗi hình ảnh tham khảo, chúng tôi tính toán các keypoint oFAST và các thuộc tính rBRIEF nhằm mục tiêu 500 keypoint cho mỗi hình ảnh. Đối với mỗi hình ảnh thử nghiệm (xoay hoặc thay đổi hướng nhìn), sau đó thực hiện đối sánh để tìm điểm tương đồng tốt nhất.



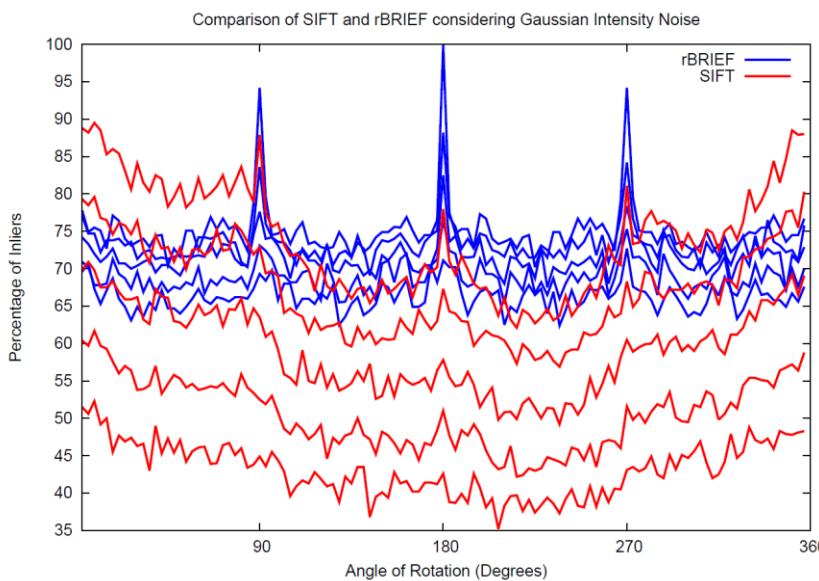
Hình 2.26: Xác định tập con các điểm kiểm tra nhị phân

Các kết quả được đưa ra trong giới hạn tỷ lệ phần trăm các đối sánh chính xác và khắc phục được những góc quay.

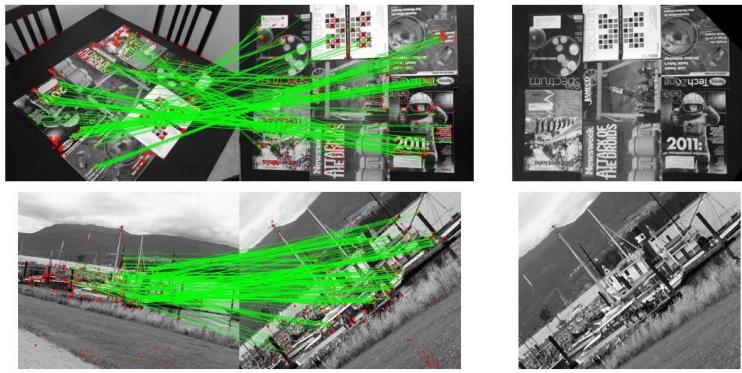


Hình 2.27: Hiệu suất đối sánh của SIFT, SURF, BRIEF với FAST và ORB

ORB chống lại nhiễu ảnh Gaussian, không như SIFT. Nếu chúng ta xét tính hiệu quả với nhiễu, SIFT là 10% với mỗi lần tăng nhiễu thêm 5. ORB cũng giảm nhưng với tốc độ thấp hơn nhiều (Hình 2.28).



Hình 2.28: Thao tác đối sánh có nhiễu cho SIFT và rBRIEF



Hình 2.29: Ví dụ thực tế về đối sánh ảnh ORB

Sau đây ta đo lường hiệu suất của ORB liên quan đến SIFT và SURF. Thực nghiệm này được thực hiện theo cách sau đây:

1. Chọn một điểm tham chiếu V_0 .
2. Đối với tất cả các V_i , tìm một điểm giống nó H_{i0} qua ánh xạ $V_i \rightarrow V_0$.
3. Nay giờ, sử dụng H_{i0} như là nền tảng cho đối sánh mô tả từ SIFT, SURF, và ORB.

2.3.7. Ước lượng các đối sánh thuộc tính nhị phân

Ta thấy rằng ORB làm tốt hơn so với SIFT/SURF trong đối sánh ở khu vực láng giềng gần với cơ sở dữ liệu hình ảnh lớn. Một phần quan trọng của ORB là sự khôi phục các biến thể, làm cho NN(nearest neibou..) tìm kiếm hiệu quả hơn.

2.3.7.1. Hàm băm cục bộ tốt cho rBRIEF

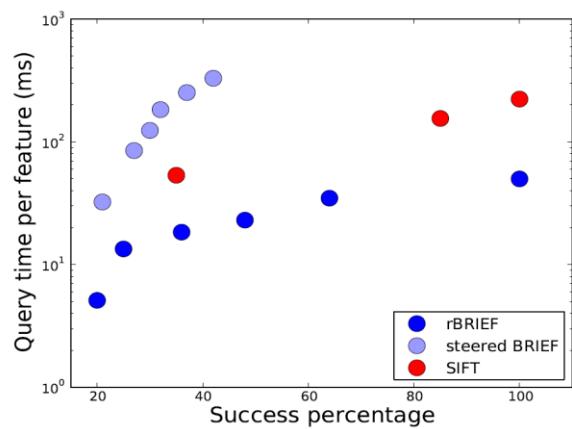
rBRIEF là một mẫu nhị phân vì thế ta chọn phép băm cục bộ tốt như tìm kiếm láng giềng gần nhất. Trong LSH, các điểm được lưu trữ trong một số bảng băm và được băm trong các ngăn khác nhau. Với một mô tả truy vấn, sự đối sánh giữa các ngăn của nó được lấy ra và các phần tử được so sánh sử dụng kết hợp sức mạnh đối sánh. Điểm mạnh của kỹ thuật nằm trong khả năng tìm láng giềng gần nhất với một xác suất cao cho đủ bảng băm.

Đối với thuộc tính nhị phân, hàm băm đơn giản chỉ là một tập hợp con của các bit chữ ký: các ngăn trong các bảng băm chứa các mô tả với một chữ ký con chung. Khoảng cách là khoảng cách Hamming.

2.3.7.2. Tương quan và Cân bằng

rBRIEF cải thiện tốc độ của LSH bằng cách làm cho các ngăn của bảng băm nhiều hơn: như các bit ít tương quan, các hàm băm làm một công việc tốt hơn trong việc phân vùng các dữ liệu. Như thể hiện trong hình 10, ngăn nhỏ hơn ở mức trung bình so với rBRIEF hoặc BRIEF

bình thường



Hình 2.31: Thử nghiệm trên phiên bản biến dạng của hình ảnh đã huấn luyện

CHƯƠNG 3. MÔ TẢ TRUY VẤN ẢNH SỬ DỤNG MÔ HÌNH GIỎ ĐẶC TRƯNG

Căn cứ vào Code tham khảo: <https://github.com/nvtiep/Instance-Search>;

Tham khảo demo tại <http://www.robots.ox.ac.uk/~vgg/research/oxbuildings/index.html> ;

Sử dụng bộ datasets <http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/> ;

Học viên mô tả lại chương trình truy vấn ảnh landmark sử dụng visual instance search như sau:

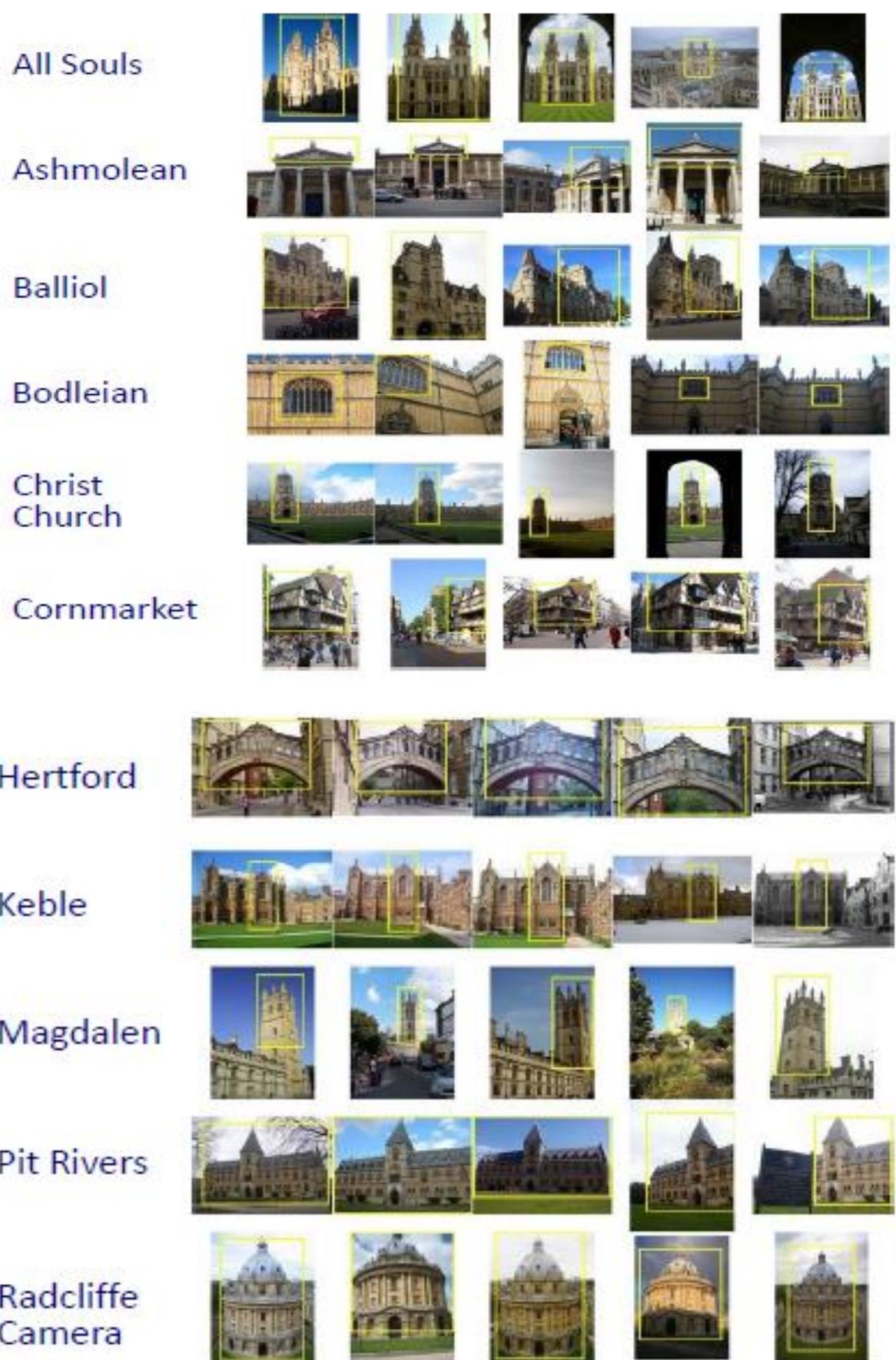
3.1. Mô tả tập dữ liệu Oxford Building

Cơ sở dữ liệu Oxford Building gồm 5062 ảnh độ phân giải cao (1024x768) từ Flickr bằng cách tìm kiếm các địa danh cụ thể của Oxford. Bộ sưu tập này đã được chủ thích thủ công tạo ra một ground truth khái quát cho 11 địa danh khác nhau. Mỗi địa danh có 5 truy vấn khác nhau được chọn, nghĩa là một tập hợp của 55 truy vấn, các ảnh được gán nhãn 1 trong 4 khả năng như sau:

- (1). Good - ảnh đẹp, rõ ràng các đối tượng/tòa nhà.
- (2). OK – hơn 25% của đối tượng là nhìn thấy được.
- (3). Junk – ít hơn 25% của đối tượng được nhìn thấy, hoặc mức độ bị che khuất, méo mó cao.
- (4). Absent – đối tượng không được biểu diễn.

Số lần xuất hiện của các địa danh khác nhau trong phạm vi 7 và 220 các ảnh Good và OK. Cơ sở dữ liệu ảnh sử dụng 55 truy vấn ($k=55$) để đánh giá hiệu năng trên ground truth. Các ảnh Good và OK được xem như là các mẫu dương, các ảnh Absent xem như các mẫu âm và các ảnh Junk xem như các mẫu trống. Những mẫu trống này được xem như không tồn tại trong cơ sở dữ liệu. Do đó, hiệu năng không bị ảnh hưởng khi chúng được trả về hoặc không.

Tập dữ liệu chứa 5062 ảnh, gồm 11 loại landmarks:

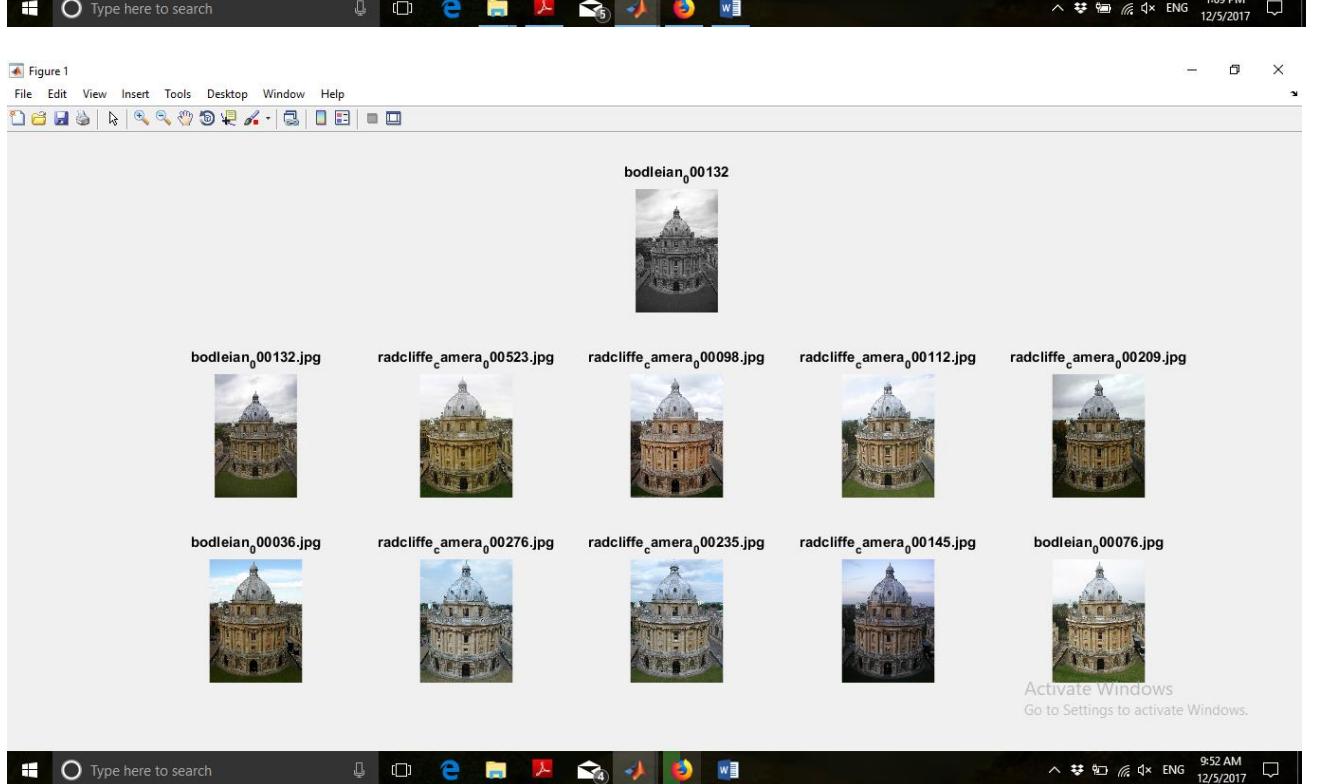
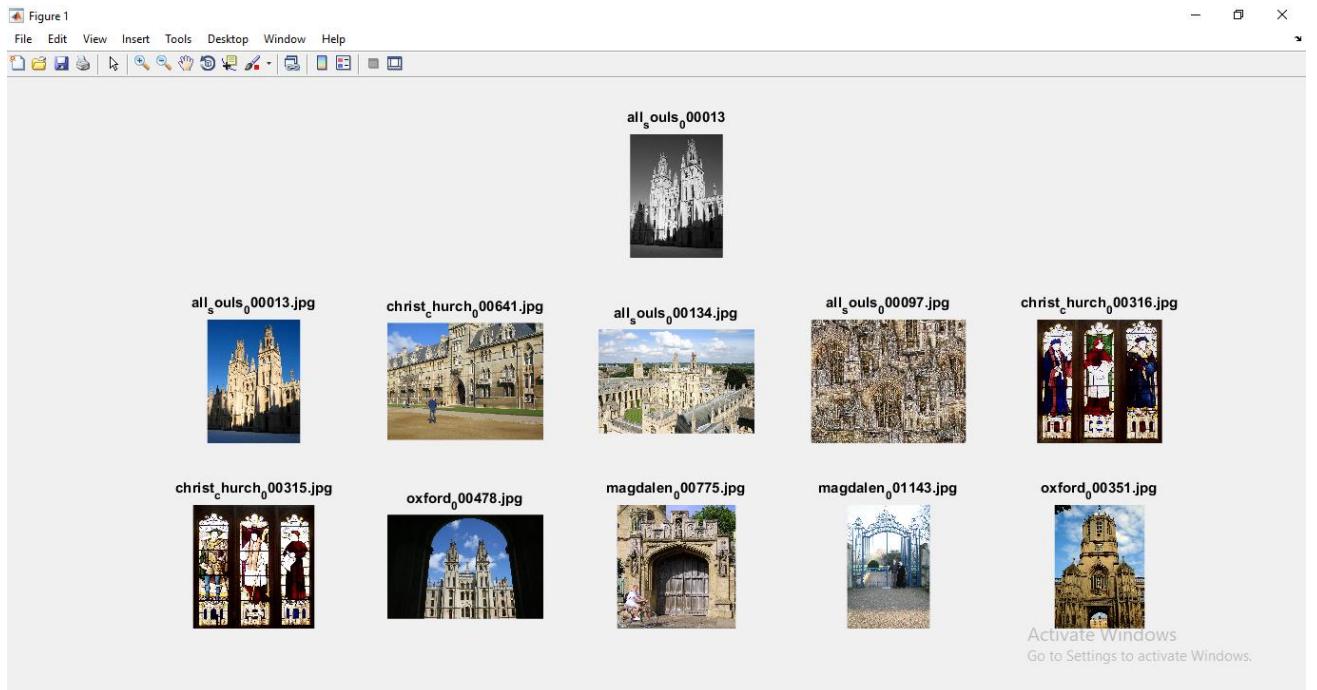


Toàn bộ 55 ảnh truy vấn được sử dụng đánh giá ground truth: mỗi dòng cho biết các truy vấn khác nhau cùng cảnh địa danh, lưu ý sự thay đổi lớn về phạm vi của các vùng truy vấn và thay đổi vị trí, ánh sáng,... của các tấm ảnh.





3.2. Kết quả thực nghiệm



3.3. Mô hình truy vấn ảnh BoW-SVM

Trong vòng 10 năm gần đây, mô hình BoW trở nên phổ biến trong lĩnh vực phân loại và truy vấn ảnh dựa trên nội dung (Content Based Image Retrieval – CIBR) vì sự đơn giản và hiệu quả. CIBR được ứng dụng rộng rãi trong thực tiễn như y học, nhà giải trí, tìm kiếm trên web, báo chí, truyền hình và quảng cáo, kinh tế, thể thao và an ninh,...

Mô hình truy vấn hình ảnh BoW là một công cụ tìm kiếm mạnh các đối tượng đích từ bộ sưu tập ảnh lớn. Tuy nhiên, độ chính xác của truy vấn bị ảnh hưởng bởi mỗi bước của mô hình BoW, cụ thể:

- Hình ảnh đại diện.
- Độ đo tương tự.
- Quy trình xếp hạng lại.

Một trong những kỹ thuật trong lĩnh vực thị giác máy tính là truy vấn đối tượng ảnh – tập trung giải quyết bài toán tìm kiếm và trả về tập ảnh liên quan đến đối tượng ảnh truy vấn có trong tập dữ liệu lớn. Mô hình này thay thế công cụ phân lớp bằng cấu trúc truy vấn được đánh chỉ mục ngược (Inverted File - IF) – kết quả trả về là một tập các ảnh được sắp xếp có độ tương đồng từ cao đến thấp giống với ảnh truy vấn nhất. Nhận thấy kỹ thuật phân lớp IF gần giống với kỹ thuật phân lớp người láng giềng gần nhất (K Nearest Neighbour- KNN).

Vai trò chính của Bag of Words (BoW) là biểu diễn các ảnh bởi các histogram dựa trên các đặc trưng không phụ thuộc, là công cụ ánh xạ các ảnh thành các vecto (histogram) biểu diễn cho ảnh và sau đó sử dụng công cụ phân lớp cho ảnh dựa trên tập các vector này. Công cụ phân lớp được sử dụng phổ biến hiện nay là Support Vector Machine (SVM). BoW kết hợp với SVM là mô hình cơ bản trong bài toán phân loại ảnh. **Mô hình BoW-SVM thực hiện qua 3 bước chính:**

(1). **Rút trích đặc trưng:** tất cả ảnh huấn luyện được tiến hành rút trích đặc trưng. Có nhiều loại đặc trưng cục bộ có thể được sử dụng như SIFT, SUFT,... Thông dụng nhất là SIFT vì bất biến theo tỷ lệ, góc quay, ánh sáng và phép chiếu.

(2). **Xây dựng tập từ điển:** áp dụng thuật toán phân cụm để nhóm các đặc trưng cùng đặc điểm lại với nhau trong tập các đặc trưng cục bộ của tập ảnh được rút trích. Sử dụng thuật toán K-Means là phổ biến nhất. Để giới hạn số lượng đặc trưng trong tập từ điển, mỗi trung tâm của phân cụm sẽ được chọn và làm đại diện cho một từ trong từ điển ảnh.

(3). Phân lớp tập dữ liệu ảnh huấn luyện: các ảnh trong tập huấn luyện lần lượt được lượng tử hóa cho các đặc trưng cục bộ dựa trên tập từ điển để ánh xạ thành các

tập các từ có trong từ điển mà ảnh đó chưa. Sau đó, xây dựng tập vector Histogram biển diễn mức độ xuất hiện của các từ trong ảnh. Tập này làm dữ liệu đầu vào trong việc phân lớp ảnh sử dụng công cụ SVM.

Trong giai đoạn thử nghiệm phân lớp, một ảnh mới thêm vào để huấn luyện cũng sẽ được thực hiện các bước tiền xử lý: rút trích đặc trưng, lượng tử hóa các đặc trưng, biểu diễn vector histogram đại diện. Sau đó, sử dụng SVM để ước tính ảnh mới này thuộc vào phân lớp nào trong tập huấn luyện.

Năm 2007, A.Bosch và cộng sự giới thiệu “Image Classification using random forests and ferns”, hiệu quả hơn SVM về hiệu xuất và thời gian thực thi trong giai đoạn huấn luyện và nhận dạng. Năm 2011, T.Kawanishi và cộng sự giới thiệu “Instance search task”, với mục đích trả về tập ảnh chưa đối tượng trong kho dữ liệu lớn, có thể nhận dạng nhiều kiểu đối tượng khác nhau như mặt người, vật, cảnh,... Mô hình này kết hợp 3 phương pháp: so khớp cục bộ, vùng, đặc trưng toàn cục. Năm 2012, R.Arandjelovic và cộng sự giới thiệu “There things everyone should know to improve object retrieval”, cải tiến SIFT bằng RootSIFT – giảm thiểu độ lớn của các phần tử trong vector đặc trưng SIFT, tăng độ chính xác hơn trong quá trình lượng tử hóa đối với kích thước của từ điển ảnh so với sử dụng Euclidean. Năm 2013, P.Bhattacharya và cộng sự giới thiệu “Spatial consistency of dense features within interest regions for efficient landmark recognition”, tác giả đưa ra phương pháp nhóm các đặc trưng SIFT được lấy mẫu trong vùng cần quan tâm, khám phá, lưu trữ thông tin hình học của các đặc trưng bên trong nhóm, tiến hành so khớp 2 nhóm đặc trưng với nhau.

Trong mô hình tìm kiếm ảnh, kích thước của tập từ điển ảnh hưởng đến độ chính xác của mô hình tìm kiếm. Vì vậy, đòi hỏi tập từ điển phải lớn, có độ phủ rộng, đa dạng. Tuy nhiên, điều này làm tăng mức độ sai số trong việc phân cụm của quá trình lượng tử hóa. Năm 2012, A.Babenko và cộng sự giới thiệu “The inverted multi-index”, không gian tìm kiếm sẽ được chia thành các vùng nhỏ dày đặc hơn làm cho thời gian truy vấn nhanh hơn, danh sách ứng viên trả về ít hơn nhưng độ chính xác cao hơn.

3.4. Mô hình truy vấn ảnh BoW-IF

Bài toán tập trung giải quyết trong mô hình BoW-IF là việc tìm kiếm nhanh và chính xác các ảnh liên quan đến ảnh người dùng truy vấn trong tập dữ liệu.

Ý tưởng chính: ánh xạ các đặc trưng cục bộ của ảnh thành các từ vựng thông qua tập từ điển ảnh, mỗi đặc trưng cục bộ của ảnh được xem tương đương như là một từ vựng, mỗi ảnh được xem như là một văn bản. Một cấu trúc chỉ mục IF được xây dựng dựa trên tập văn bản đã được ánh xạ, cấu trúc chỉ mục này là cơ sở cho việc truy vấn ảnh đối tượng.

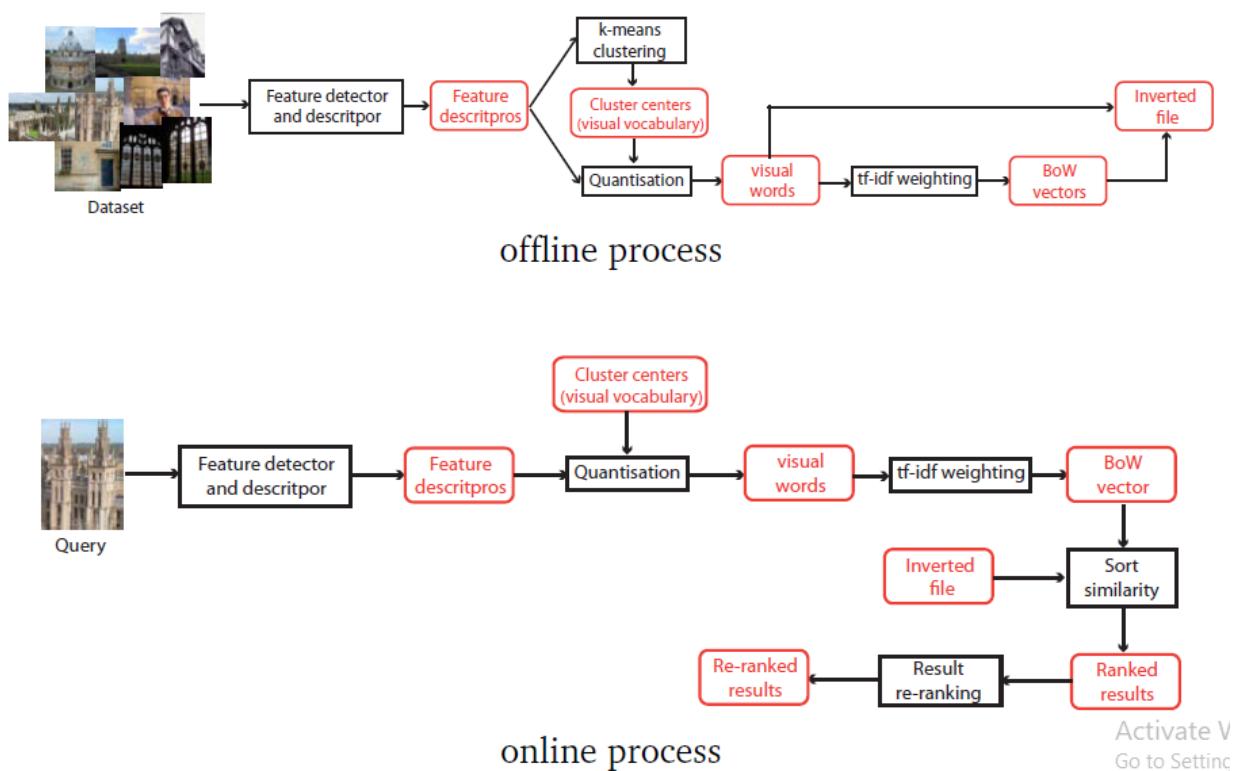
Mô hình truy vấn ảnh được chia làm 2 giai đoạn:

1) Huấn luyện: bao gồm 4 bước:

- Rút trích đặc trưng.
- Xây dựng tập từ điển ảnh.
- Lượng tử hóa đặc trưng của các ảnh huấn luyện.
- Xây dựng cấu trúc chỉ mục IF.

2) Truy vấn ảnh: bao gồm 2 bước:

- Tiền xử lý ảnh truy vấn.
- Truy vấn ảnh.



Cụ thể như sau:

3.4.1. Giai đoạn huấn luyện

3.4.1.1. Rút trích đặc trưng.

Mỗi ảnh trong tập data training sẽ được rút trích các đặc trưng SIFT, sau đó, tất cả các ảnh này được chuyển đổi sang RootSIFT (RootSIFT là cải tiến của SIFT, tăng cường độ chính xác, chi phí của sự chuyển đổi từ SIFT sang RootSIFT không đáng kể).

3.4.1.2. Xây dựng tập từ điển ảnh.

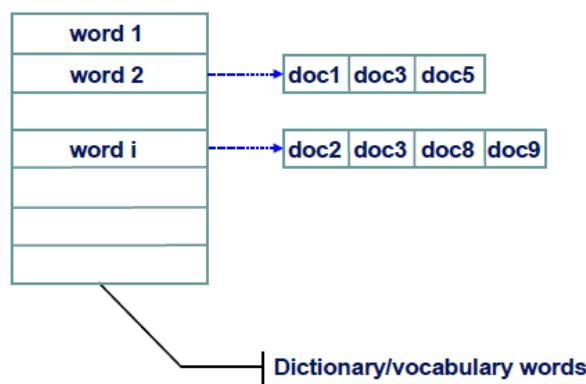
Xây dựng tập từ điển ảnh bằng cách gom tất cả các đặc trưng RootSIFT lại và áp dụng thuật toán phân cụm Approximate Kmean (AKM). Hoàn thành bước này, ta thu được tập từ điển ảnh và cây KD-Tree (công dụng của KD-Tree là giúp ước lượng các đặc trưng của ảnh, lưu trữ cây KD-Tree để giảm thời gian tính toán).

3.4.1.3. Lượng từ hóa đặc trưng của các ảnh huấn luyện.

Sử dụng cây KD-Tree để lượng từ hóa đặc trưng của các ảnh trong tập huấn luyện, xác định các đặc trưng RootSIFT của ảnh thuộc vào cụm nào, nghĩa là ảnh xạ đặc trưng của ảnh tương ứng với từ vựng trong bộ từ điển. Các từ vựng có trong bộ từ điển có đánh thứ tự chỉ mục (Word-ID), nghĩa là tính toán thứ tự của từ vựng được ánh xạ cho các đặc trưng RootSIFT của từng ảnh.

3.4.1.4. Xây dựng cấu trúc chỉ mục IF.

Mỗi ảnh được biểu diễn như là một văn bản, các đặc trưng được xem là một từ và mỗi từ có thứ tự chỉ mục Word-ID tương ứng với thứ tự từ vựng ánh xạ của nó trong tập từ điển. Sau đó, xây dựng cấu trúc tìm kiếm IF, lập chỉ mục cho từng ảnh trong tập huấn luyện. Cấu trúc chỉ mục này sẽ được lưu trữ trong hệ thống và là cơ sở cho việc tìm kiếm ảnh trong giai đoạn truy vấn.



3.4.2. Giai đoạn truy vấn ảnh

Ở giai đoạn này, một ảnh chứa đối tượng cần truy vấn được đưa vào hệ thống. Hệ thống sẽ thực hiện tiền xử lý ảnh truy vấn và truy vấn ảnh thông qua cấu trúc chỉ mục IF đã xây dựng ở giai đoạn huấn luyện.

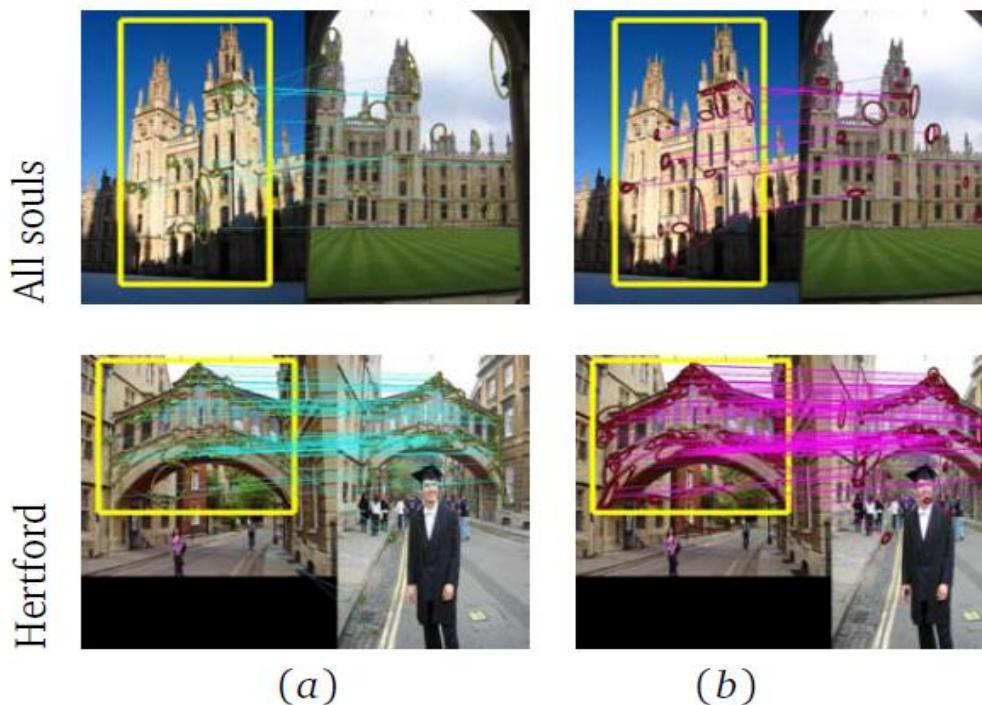
3.4.2.1. Tiền xử lý ảnh truy vấn

Thực hiện ánh xạ ảnh truy vấn như là một tài liệu văn bản. Tương tự như quá trình xử lý đối với các ảnh trong giai đoạn huấn luyện, tức là ảnh truy vấn được xử lý rút trích đặc trưng

RootSIFT và lượng tử hóa các đặc trưng ảnh thành từ vựng nhò vào tập từ điển ảnh hoặc cây KD-Tree thu được ở giai đoạn huấn luyện.

3.4.2.2. Truy vấn ảnh

Kết hợp với cấu trúc chỉ mục IF đã được xây dựng và lưu trữ trong hệ thống, tính toán để trả về một danh sách các ảnh gần giống nhất.



a) Đặc trưng SIFT

Khi sử dụng lược đồ màu toàn cục để đối sánh và tính toán sự phân bố màu sắc của hình ảnh, kết quả tìm kiếm ảnh có độ chính xác thấp trong trường hợp các hình ảnh có cùng số lượng điểm ảnh nhưng có vị trí phân bố màu sắc không tương ứng với nhau. Nếu sự phân bố màu sắc trên hình ảnh không tạo thành một vùng liên thông thì việc đối sánh có thể dẫn đến thiếu chính xác mặc dù các hình ảnh có cùng tỉ lệ màu sắc. Do đó, cần đánh giá độ tương tự của hình ảnh ứng với vị trí phân bố màu sắc để tăng độ chính xác tìm kiếm các hình ảnh tương tự.

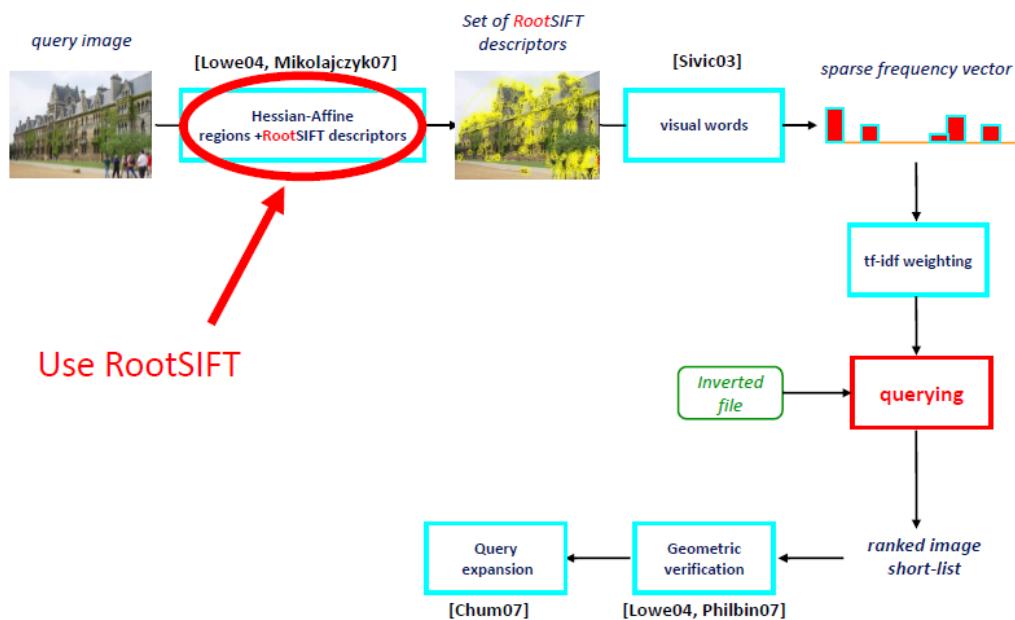
Có nhiều phương pháp dò tìm đặc trưng thông dụng gồm phương pháp dò góc và cạnh được giới thiệu vào năm 1998 bởi Harris và M. Stephens, phương pháp dò tìm đặc trưng SIFT (Scale Invariant Features Transform) dựa trên phép lọc của mặt nạ tích chập giữa hình ảnh và đạo hàm riêng DoG (Difference of Gaussian) nhằm xấp xỉ toán tử Laplace của hàm Gauss được giới thiệu năm 2003 bởi D.Lowe, phương pháp dò tìm đặc trưng SURF (Speeded Up Robust Feature) được giới thiệu vào năm 2006 bởi Bay và cộng sự, phương pháp dò điểm đặc trưng

Harris-Laplace dựa trên toán tử Laplace của hàm Gauss được giới thiệu năm 2001 bởi Mikolajczyk và C.Schmid,...

Đặc trưng SIFT được mô tả như một vector 128 chiều, chứa đựng các thông tin chênh lệch mức xám tại vùng nó biểu diễn, bất biến với vị trí, tỷ lệ và hướng.

Ý nghĩa chính của SIFT: tìm ra những đặc trưng quan trọng của ảnh, so sánh các đặc trưng này với bộ tập ảnh, công thức tính khoảng cách Euclidean thường được áp dụng. Độ đo Euclidean không được chính xác khi có một vector đặc trưng mang giá trị lớn.

b) Đặc trưng ROOTSIFT



Sử dụng độ đo Euclidean $d_E(x, y)$ và độ đo tương đồng Hellinger $H(x, y)$ trong tính khoảng cách giữa các đặc trưng SIFT nhằm nâng cao độ chính xác của mô hình. Với x, y là 2 vector được chuẩn hóa Euclidean. Ta có:

$$d_E(x, y)^2 = \|x - y\|_2^2 = \|x\|_2^2 + \|y\|_2^2 - 2x^T y = 2 - 2x^T y$$

$$H(x, y) = \sum_{i=1}^n \sqrt{x_i y_i}$$

Chuyển từ đặc trưng SIFT sang RootSIFT như sau:

$$\begin{aligned} d_E(\sqrt{x}, \sqrt{y})^2 &= \|\sqrt{x} - \sqrt{y}\|_2^2 = \|x\|_1^2 + \|y\|_1^2 - 2\sqrt{x^T y} = 2 - 2\sqrt{x^T y} \\ &= 2 - 2H(x, y) \end{aligned}$$

Với việc sử dụng RootSIFT, việc so sánh sử dụng công thức Euclidean tương đương với việc sử dụng độ đo Hellinger. Sử dụng Hellinger để nâng cao hiệu quả truy vấn ảnh. Gọi X là

vector có n=128 chiều biểu diễn cho một đặc trưng cục bộ SIFT của ảnh, công thức chuyển đổi đặc trưng SIFT sang RootSIFT như sau:

$$L1(X) = \sum_{i=1}^n x_i$$

$$RootSIFT(X) = \{x'_i | x'_i = \sqrt{x_i/L1(X)}, \forall i \in n\}$$

Trong Matlab:

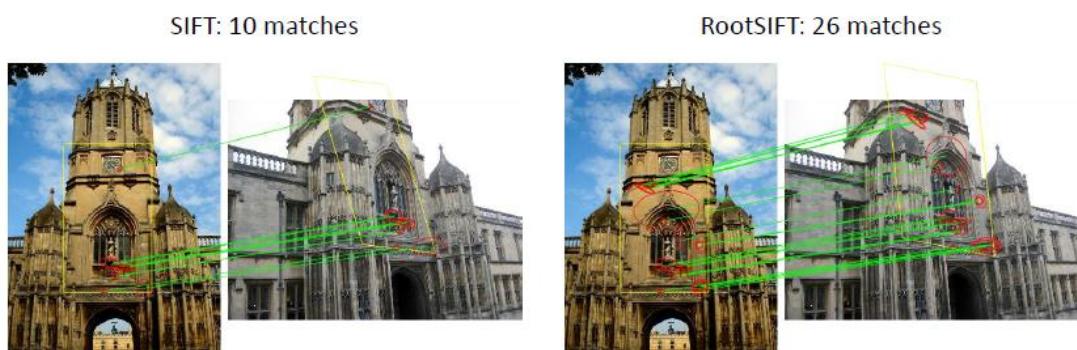
```
rootsift= sqrt( sift / sum(sift) );
```

Philbin et.al. 2007: bag of visual words with:

- tf-idf ranking
- or tf-idf ranking with spatial reranking

Retrieval method	Oxford 5k
SIFT: tf-idf ranking	0.636
SIFT: tf-idf with spatial reranking	0.672
RootSIFT: tf-idf ranking	0.683
RootSIFT: tf-idf with spatial reranking	0.720

Retrieval method	Oxford 5k
SIFT: tf-idf ranking	0.636
SIFT: tf-idf with spatial reranking	0.672
DBN SIFT: tf-idf with spatial reranking	0.707
RootSIFT: tf-idf ranking	0.683
RootSIFT: tf-idf with spatial reranking	0.720

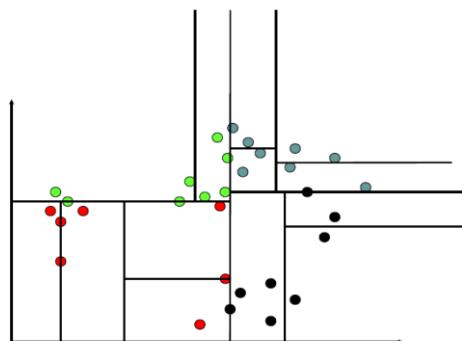


c) Thuật toán APPROXIMATE K-MEANS (AKM)

Ý tưởng chính của thuật toán K-Means: tìm cách phân nhóm n các đối tượng đã cho vào k cụm (với k nguyên dương, cho trước), sao cho tổng bình phương khoảng cách giữa các đối tượng đến tâm nhóm là nhỏ nhất. Trong lĩnh vực thị giác máy tính, dùng K-Means để xây dựng tập từ điển ảnh trong mô hình BoW, với độ phức tạp của thuật toán là $O(nk)$. Khi $k > 10^6$ thì thuật toán K-Means trở nên không hiệu quả.

Năm 2013, J.Wang và cộng sự giới thiệu “Fast approximate k-means via cluster closures” cải tiến K-Means. Ý tưởng chính của AKM: sử dụng cây KD-Tree cho việc phân cụm dữ liệu, việc lượng tử hóa một điểm vào một cụm là phép duyệt và tìm kiếm trên cây nhị phân, độ phức tạp của thuật toán là $O(nlogk)$.

Kd-Tree : construction



Với mô hình BoW-IF sử dụng đặc trưng RootSIFT, xây dựng cây KD-Tree với số chiều là 128 chiều + số lượng phân cụm lớn + số lượng không gian phân tách có thể làm tăng tình trạng phân bố thưa thì sử dụng KNN để duyệt tìm kiếm phần tử trong KD-Tree là không chính xác, không hiệu quả. Khắc phục: sử dụng nhiều cây KD-Tree, quá trình xử lý thực hiện song song.

d) Cấu trúc chỉ mục IF

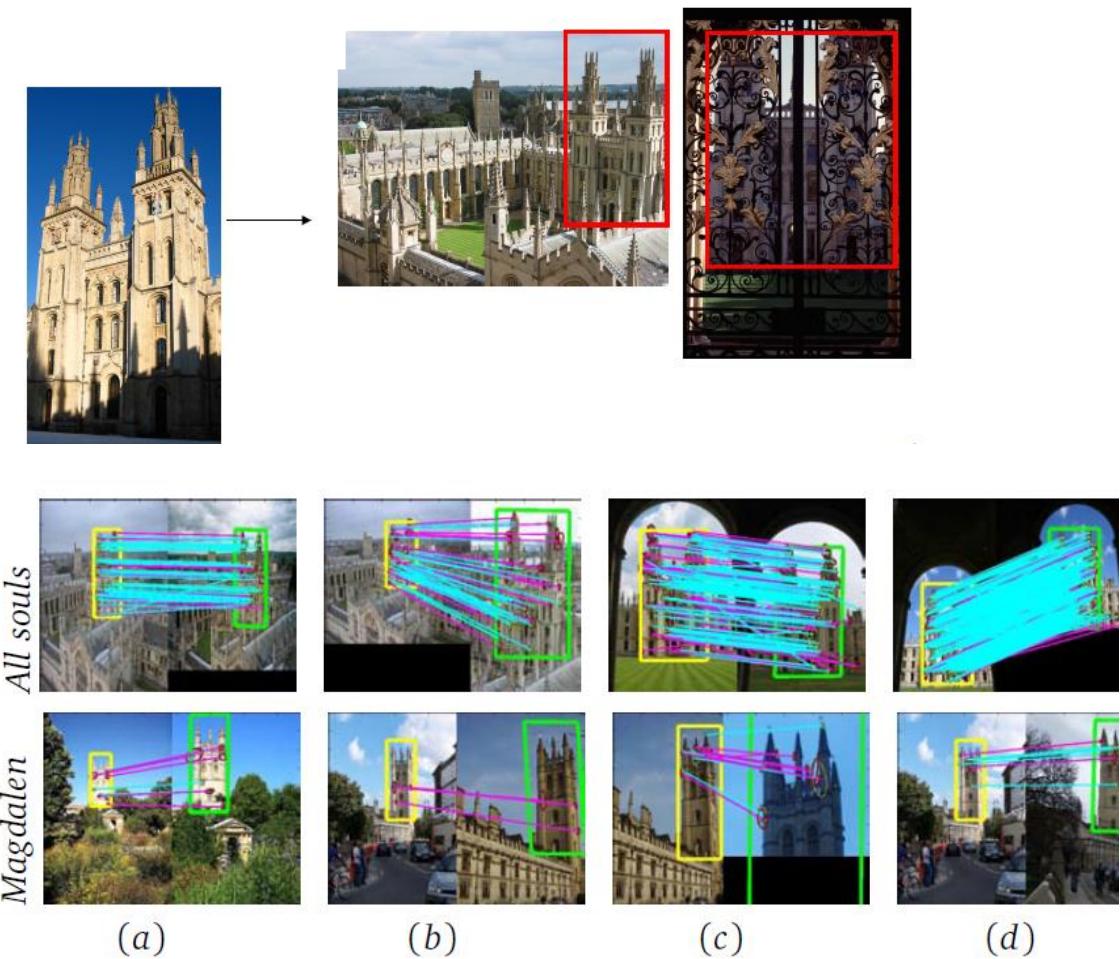
Mục đích lưu trữ tần suất xuất hiện các đặc trưng của ảnh, để tính toán mức độ giống nhau giữa 2 ảnh đối sánh thông qua việc thực hiện tính toán độ sai biệt của tập ảnh được đánh chỉ mục với ảnh truy vấn. Mặt khác, sắp xếp thứ tự các ảnh theo các mức độ khác nhau và ưu tiên trả về tập ảnh giống ảnh truy vấn nhất.

Mô hình sử dụng độ đo heuristic S (độ đo khoảng cách) để đánh giá mức độ khác nhau giữa ảnh huấn luyện và ảnh truy vấn. Độ đo khoảng cách được tính toán dựa trên các thành phần sau:

- f_{dt} : tần suất xuất hiện đặc trưng t trong ảnh lập chỉ mục d .

- f_{qt} : tần suất xuất hiện đặc trưng t trong ảnh truy vấn q.
- f_t : số lượng ảnh chứa đặc trưng t trong toàn bộ tập dữ liệu ảnh.
- F_t : tần suất xuất hiện đặc trưng t trong toàn bộ tập dữ liệu ảnh.
- N: số lượng ảnh trong tập dữ liệu huấn luyện.
- n_d : số lượng đặc trưng có trong ảnh d.
- n: kích thước từ điển trong IF.

Mục đích của việc xây dựng cấu trúc chỉ mục IF là dùng lưu trữ những giá trị tần xuất trên của các ảnh trong tập huấn luyện, đảm bảo dễ dàng truy xuất đặc trưng khi tính toán độ sai khác S của tập ảnh được đánh chỉ mục với ảnh truy vấn. Sau đó, sắp xếp ảnh theo các mức độ khác nhau, ưu tiên trả về tập ảnh giống ảnh truy vấn nhất.



Để thực hiện truy vấn một ảnh cần phải tính toán trọng số của các từ trong ảnh đó. Giá trị trọng số này thường tính toán dựa trên công thức tf-idf. Nếu sử dụng trực tiếp cấu trúc IF ban đầu cho việc lưu trữ các trọng số thì sẽ không thể cập nhật cấu trúc chỉ mục bằng cách thêm ảnh mới vào hệ thống. Các giá trị tần số được biến đổi thành các giá trị trọng số như là một cấu trúc tách biệt để thực hiện truy vấn. Còn cấu trúc ban đầu được sử dụng để cập nhật khi thêm ảnh mới.

Thuật toán truy vấn ảnh sử dụng IF:

Khi tính toán trọng lượng của một từ xuất hiện trong văn bản, công thức phổ biến là tf-idf, cụ thể:

- Trọng lượng sẽ bị giảm khi chúng xuất hiện trong nhiều ảnh khác.
- Trọng lượng tăng khi chúng có tần suất xuất hiện nhiều trong ảnh đó.
- Trọng lượng giảm khi ảnh chứa từ vựng ảnh đó có kích thước lớn.

Công thức tổng quát để tính trọng số của một visual words:

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D) \quad (*)$$

Trong đó:

- $tf(t,d)$: tần số xuất hiện của mục từ vựng ảnh t trong ảnh đang xét d .
- $idf(t,D)$: tần số xuất hiện của các ảnh trong tập dữ liệu có chứa mục từ t .

Sử dụng (*) để tính trọng số của từ đặc trưng cho ảnh đã lập chỉ mục.

Trong hệ thống BoW-IF:

$$tf(t,d) = \frac{f_{d,t}}{n_d}$$

(Tỷ lệ theo độ tăng kích thước của ảnh: mỗi ảnh trong tập dữ liệu có kích thước, tỷ lệ khác nhau, làm tăng số lượng đặc trưng giữa các ảnh.)

$$idf(t,D) = \log_2 \frac{N}{ft}$$

(Độ hỗ trợ của từ này đối với ảnh đang xét: nếu từ này xuất hiện trong nhiều ảnh đang xét khác nhau thì $idf(t,D)$ sẽ làm giảm trọng số của tf-idf, nghĩa là tính phổ biến của từ cao, không mang đặc trưng riêng biệt cho ảnh đang xét)

Đối với ảnh truy vấn, ta tính như sau:

$w_{d,t} = \frac{f_{d,t}}{n_d} \log_2 \frac{N}{ft}$	$w_{q,t} = \frac{f_{q,t}}{n_q} \log_2 \frac{N}{ft}$
$w_d = \sum_{t \in T_d} w_{d,t}$	$w_q = \sum_{t \in T_q} w_{q,t}$

$$K_{q,d} = \sum_{t \in T_q} \left(\left| \frac{w_{d,t}}{w_d} \right| + \left| \frac{w_{q,t}}{w_q} \right| - \left| \frac{w_{d,t}}{w_d} - \frac{w_{q,t}}{w_q} \right| \right)$$

$K_{q,d} \in [0,2]$;

$K_{q,d} = 2$ khi $w_{d,t} = w_{q,t}$;

$K_{q,d} = 0$ khi trong $w_{d,t}$ không tồn tại $w_{q,t}$.

Trong đó:

- $w_{d,t}$: trọng số tf-idf của các từ t trong ô lưới của ảnh đánh chỉ mục d.
- $w_{q,t}$: trọng số tf-idf của các từ t trong ô lưới của ảnh truy vấn q.
- w_d : trọng số tổng của ảnh đánh chỉ mục d.
- w_q : trọng số tổng của ảnh truy vấn q.
- $K_{q,d}$: tính độ sai biệt khoảng cách giữa 2 ảnh: ảnh đánh chỉ mục d và ảnh truy vấn q; sử dụng công thức đo độ tương đồng giữa 2 ảnh. Phản ánh độ giống nhau giữa 2 ảnh.

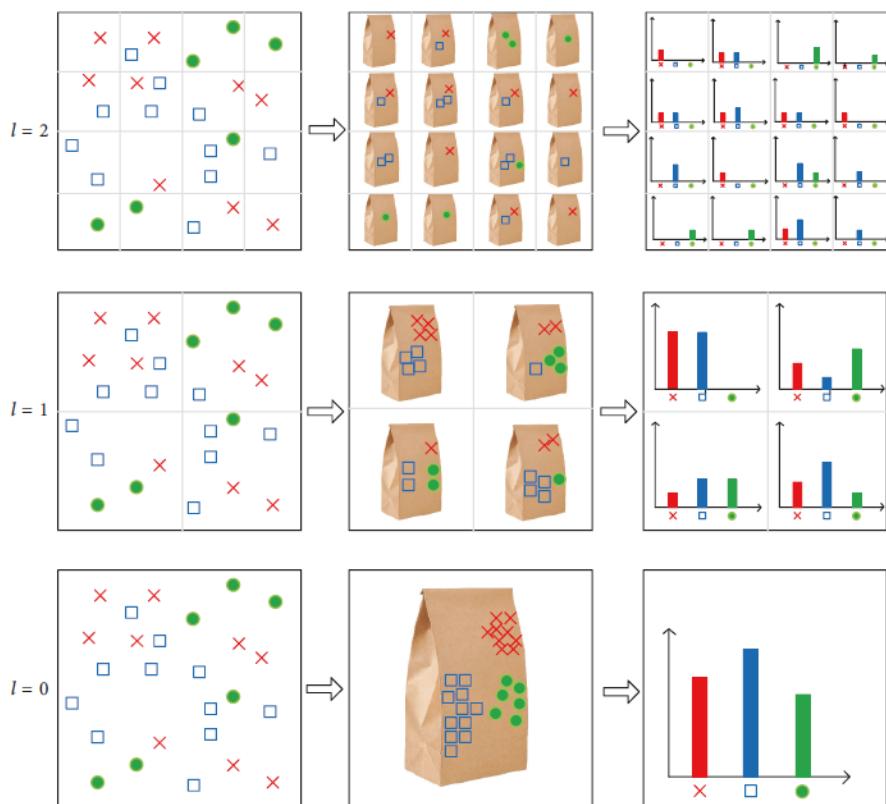
Để sắp xếp các ảnh trong tập huấn luyện được đánh chỉ mục và trả về r ảnh giống với ảnh truy vấn nhất trong mô hình IF:

- (1). Tính các giá trị $w_{q,t}, w_q$ tương ứng với giá trị trọng số tf-idf của mục từ t và trọng số tổng của các từ có trong ảnh cần truy vấn.
- (2). Khởi tạo danh sách ảnh $T = \emptyset$ là danh sách ảnh được trả về.
- (3). Ứng với mỗi mục từ ảnh t có trong ảnh truy vấn q, thực hiện lấy danh sách ảnh có trong list của t. Ứng với mỗi ảnh d có trong danh sách t (nếu d không tồn tại trong t thì thêm d vào t), thực hiện gán giá trị để so sánh sự sai biệt của ảnh d so với ảnh truy vấn.
- (4). Thực hiện truy xuất các giá trị $w_{d,t}, w_d$ đã được tính toán và lưu trữ trước đó (nếu ảnh mới thêm vào bộ dữ liệu thì thực hiện tính toán lại). Tính $S_{q,d} = S_{q,d} - K_{q,d}$ nhằm giảm độ sai biệt giữa ảnh d và q. Sau khi đã duyệt qua từng từ trong ảnh truy vấn, ta có tập $S_{q,d}$ đại diện cho độ sai biệt giữa d, q trong T.
- (5). Sắp xếp các ảnh tăng dần theo giá trị $S_{q,d}$ trong tập T. Chọn r ảnh đầu tiên có giá trị $S_{q,d}$ nhỏ nhất trả về cho kết quả tìm kiếm ảnh của mô hình.

Dataset	# images	# features	Vocabulary size	# feat/#img
Oxford 5K	5062	12.7M	1M	2.5K

3.5. Cấu trúc chỉ mục thông tin không gian phân cấp

Cấu trúc chỉ mục thông tin không gian phân cấp - Spatial Pyramid Inverted File (SPIF): ảnh sẽ được tích hợp không gian lưới ở nhiều mức khác nhau lên không gian tọa độ của tập đặc trưng. Nghĩa là phân hoạch tọa độ đặc trưng như mô hình kim tự tháp, số lượng ô lưới tăng dần theo mức của tháp phân hoạch, giúp thông tin vị trí của đặc trưng được tăng cường. Các đặc trưng của 2 ảnh được so khớp khi chúng cùng nằm trên một ô lưới và cùng trên một mức tương ứng của tháp phân hoạch.



Hình : Mô hình không gian phân cấp của giỏ đặc trưng.

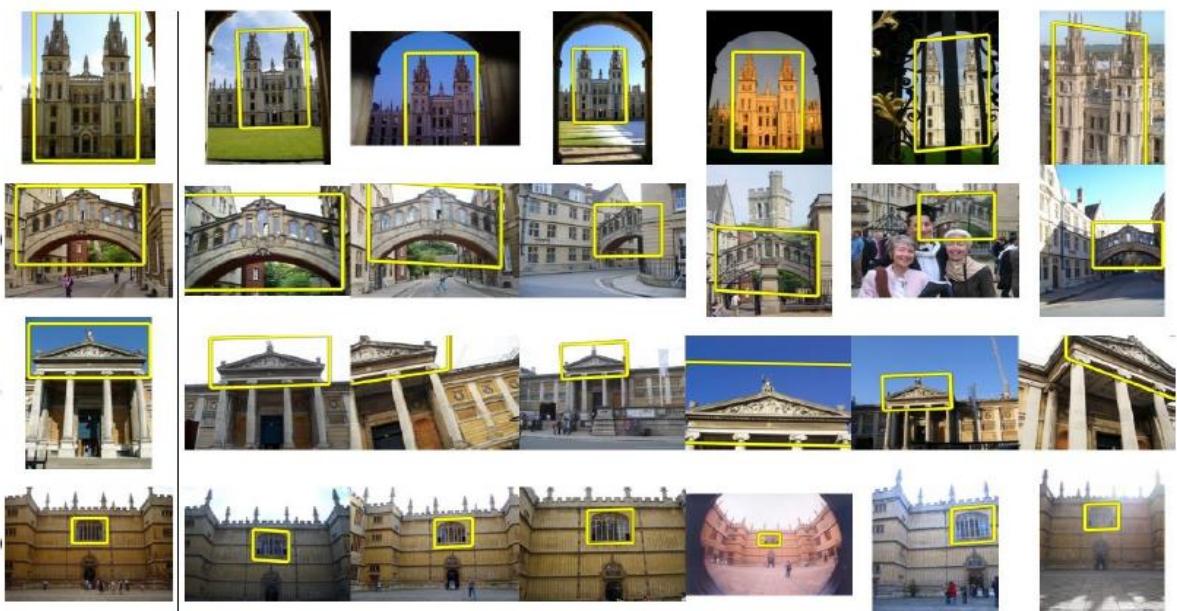
Để áp dụng được mô hình, cần cập nhật thêm thông tin tọa độ phân cấp vào cấu trúc chỉ mục IF, tức cập nhật thêm thông tin vị trí ô lưới mà mục từ đó xuất hiện trong tháp thông tin phân cấp.

Gọi L là số tầng tháp thông tin phân cấp thì số lượng ô lưới tại mức 1 tương ứng là $C_l = 4^{l-1}$. Như vậy cần lưu $\frac{1}{3}(4^{l-1} - 1)$ các giá trị tần số xuất hiện của một mục từ trong từng ô của từng mức. Ta có:

- $f_{d,t}^l(i,j)$: tần số xuất hiện của mục từ t trong ảnh d tại ô lưới có dòng i , cột j , ở mức l .
- $n_d^l(i,j)$: tổng số lượng các đặc trưng có trong ảnh d tại ô lưới có dòng i , cột j , ở mức l .
- $f_{q,t}^l(i,j)$: tần số xuất hiện của mục từ t trong ảnh q tại ô lưới có dòng i , cột j , ở mức l .
- $n_q^l(i,j)$: tổng số lượng các đặc trưng có trong ảnh q tại ô lưới có dòng i , cột j , ở mức l .

Các đặc trưng trong một ô lưới bằng ở mức 1 bằng tập hợp tất cả các đặc trưng ở 4 ô lưới tương ứng ở mức cao hơn. Vì vậy, chỉ cần duyệt qua tập đặc trưng của ảnh một lần và phân hoạch các đặc trưng vào các ô lưới để tính tần số xuất hiện của chúng ở mức cao nhất.

Để thực hiện truy vấn một ảnh cần tính trọng số tf-idf của các đặc trưng trong ảnh đó tại mỗi ô lưới của tháp phân cấp. Nghĩa là thực hiện cập nhật các $f_{d,t}^l(i,j)$ thành $w_{d,t}^l(i,j)$. Trong trường hợp, sử dụng trực tiếp cấu trúc SPIF ngay từ đầu cho việc cập nhật các trọng số, khi thêm ảnh mới thì không thể cập nhật được tiếp theo. Do đó, thực hiện lưu trữ SPIF bằng cách biến đổi các giá trị tần số thành các giá trị trọng số để thực hiện truy vấn. Cấu trúc ban đầu sử dụng để cập nhật thêm ảnh mới.



Các bước thực hiện xây dựng cấu trúc chỉ mục SPIF như sau:

- (1). Khởi tạo cấu trúc chỉ mục SPIF ban đầu rỗng: với thành phần IF là danh sách các mục từ t có trong tập từ điển ảnh: $F_t = 0; f_t = 0$ (F_t là tần suất của từ t có trong tập dữ liệu ảnh, f_t là số lượng ảnh chứa từ vựng ảnh t.) List trong IF được cài đặt với cấu trúc nhị phân hay bảng băm cho phép duyệt nhanh trong quá trình truy vấn.
- (2). Cập nhật các giá trị của SPIF thông qua tập dữ liệu: mỗi ảnh trong tập dữ liệu được thực hiện lượng tử hóa: phân hoạch không gian, xác định các giá trị tần suất $f_{d,t}^l(i,j), n_d^l(i,j)$
- (3). Rút trích thông tin giá trị tần suất của ảnh d.

Tính toán độ sai biệt khoảng cách $S_{q,d}$ dựa vào tổng giá trị $S_{q,d}^l$ ở từng mức thấp không gian phân cấp. $S_{q,d}^l = \sum S_{q,t}^l(i,j)$

Xác định trọng số của các từ như sau:

$w_{d,t}^l(i,j) = \frac{f_{d,t}^l(i,j)}{n_d^l(i,j)} \log_2 \frac{N}{f_t}$	$w_{q,t}^l(i,j) = \frac{f_{q,t}^l(i,j)}{n_q^l(i,j)} \log_2 \frac{N}{f_t}$
$w_d^l(i,j) = \sum_{t \in T_d^l(i,j)} w_{d,t}^l(i,j)$	$w_q^l(i,j) = \sum_{t \in T_q^l(i,j)} w_{q,t}^l(i,j)$
$K_{d,t}^l(i,j) = \sum_{t \in T_d^l(i,j)} \left(\left \frac{w_{d,t}^l(i,j)}{w_d^l(i,j)} \right + \left \frac{w_{q,t}^l(i,j)}{w_q^l(i,j)} \right - \left \frac{w_{d,t}^l(i,j)}{w_d^l(i,j)} - \frac{w_{q,t}^l(i,j)}{w_q^l(i,j)} \right \right)$ $K_{d,t}^l(i,j) \in [0,2];$ $K_{d,t}^l(i,j) = 2 \text{ khi } w_d^l(i,j) = w_q^l(i,j);$ $K_{d,t}^l(i,j) = 0 \text{ khi trong } w_d^l(i,j) \text{ không tồn tại } w_q^l(i,j).$	

Các bước thực hiện để xếp hạng và trả về r ảnh có độ sai biệt khoảng cách thấp nhất trong tập ảnh được đánh chỉ mục theo mô hình SPIF có mức L với một ảnh truy vấn q như sau:

- (1). Tính $w_d^l(i,j), w_q^l(i,j)$ tương ứng với giá trị trọng số tf-idf của mục từ t và trọng số tổng của toàn bộ mục từ t của ô lưới có trong các mức của ảnh truy vấn.
- (2). Khởi tạo danh sách ảnh $T=\emptyset$; T là danh sách ảnh trả về của thuật toán.
- (3). Ứng với từ vựng ảnh t có trong ảnh q, thực hiện lấy danh sách các ảnh có trong list của t. Ứng với mỗi ảnh d có trong list của t (nếu d không tồn tại trong T thì

thêm d vào T) gán giá trị để so sánh độ sai biệt của ảnh d với giá trị khởi tạo $2(L+1)$.

(4). Thực hiện truy xuất các giá trị $w_d^l(i,j), w_q^l(i,j)$ ở từng ô lướt của từng mức lướt.

$$\text{Tính } S_{q,d} = S_{q,d} - \frac{1}{4^l} K_{d,t}^l(i,j)$$

(5). Sắp xếp các ảnh tăng dần theo giá trị $S_{q,d}$ trong tập T. Chọn ảnh đầu tiên có giá trị $S_{q,d}$ nhỏ nhất trả về cho kết quả tìm kiếm ảnh của mô hình.

3.6. Kết hợp mô hình BoVW và CNNs

Để khắc phục nhược điểm của mô hình BoW truyền thống khi tập dữ liệu lớn và đa dạng, chúng ta sử dụng mạng neural thần kinh xoắn (Convolutional neural networks - CNNs) để tối ưu hóa quá trình giải thuật trích xuất hình ảnh, sau đó, chọn một lớp phù hợp để nhúng vào mô hình BoVW, ánh xạ các tập các từ có trong từ điển mà ảnh đó chứa, cuối cùng, nhận được kết quả ảnh trả về trong việc huấn luyện phân lớp ảnh. CNNs là một kiến trúc lấy cảm hứng từ khả năng huấn luyện sinh học có thể học phân cấp đa tính năng, nghĩa là phát triển cấu trúc học tập đa lớp.

Cách đơn giản nhất để tăng hiệu suất của CNNs là tăng độ sâu và chiều rộng của mạng neural thần kinh. Nhược điểm của phương pháp này là: mạng càng lớn, càng nhiều thông số. Việc sử dụng các tài nguyên tính toán được mở rộng bằng cách kiến trúc được kết nối thưa, dùng CNNs để trích xuất các đặc trưng.

3.6.1. Rút trích đặc trưng sử dụng CNNs

Cấu trúc mạng của CNNs so với mạng truyền thống có 3 điểm chính:

- Được kết nối cục bộ: kết nối thưa giữa những đơn vị xoắn giữa các lớp liên tiếp.
- Chia sẻ trọng số: chia sẻ tham số của mô hình giữa các vị trí khác nhau.
- Lấy mẫu phụ, tăng tính toàn cục dựa trên cấu trúc nhiều lớp biểu diễn từ đặc trưng cấp thấp đến cao.

Các biến chính yếu ảnh hưởng đến chất lượng của các đặc trưng được rút trích là: trường thu cảm, hạt nhân, số lượng lớp và số lượng các bản đồ đặc trưng của mỗi lớp. Việc trích xuất các đặc trưng dựa vào các mối tương quan không gian khác nhau một cách tự động và tìm ra thông tin cấu trúc trực quan với sự công nhận tự động trong ảnh và tập trung vào nội dung của hình ảnh theo khu vực đã xác định. Nhập các đặc trưng vào mô hình BoVW để đạt được hiệu quả nhận diện tốt hơn.

Cụ thể như sau:

- Trong lớp xoắn, kích thước của ảnh gốc I là $l \times h$, được định nghĩa là ma trận x . Chia ảnh I thành n ảnh S có kích thước $a \times b$ bằng cách đào tạo mã hóa thưa. Chúng chứa k đặc điểm được tính như sau: $f = \sigma(wx_s + b)$. Trong đó: σ là hàm kích hoạt; w là trọng số; b là độ lệch giữa đơn vị lớp trực quan và đơn vị tiềm ẩn. Đối với mỗi ảnh S , ta có: $f_S = \sigma(w^1x_s + b^1)$.

- Sau khi thu được các đặc trưng bằng cách xoắn, các đặc tính này được phân lớp. Để giới hạn số lượng đặc trưng trong tập từ điển, mỗi trung tâm của phân cụm sẽ được chọn và làm đại diện cho một từ trong từ điển ảnh.

3.6.2. Kết hợp mô hình BoVW và CNNs

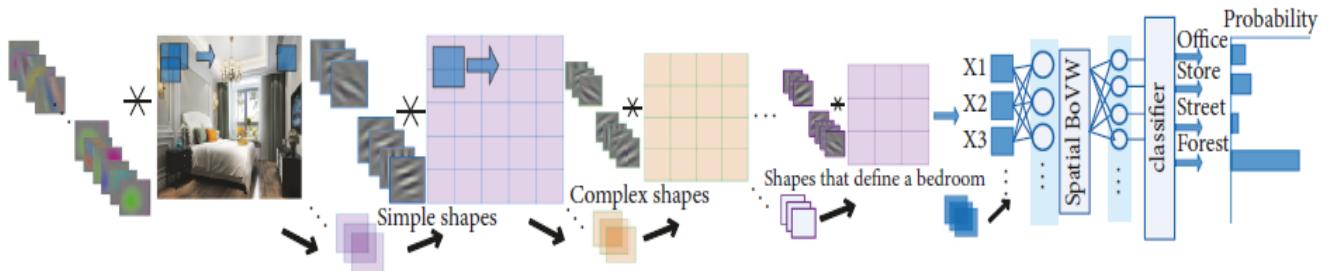


FIGURE 1: The architecture of our model.

Với ảnh I , cắt CNNs ở lớp xoắn cuối cùng xem như một cách rút gọn mô tả tính năng đặc. Đầu ra của lớp xoắn cuối cùng là một bản đồ H^*W^*F , là một tập hợp các mô tả chiều F được chiết xuất tại các vị trí chiều rộng W và chiều cao H , F đại diện cho số lượng các kênh hay số lượng các tính năng của một vùng, được ánh xạ thành các tập từ có trong tập từ điển.

Bộ từ điển xoắn $D=[d_1, d_2, \dots, d_k]$ có thể được giám sát hay không phụ thuộc vào chế độ mã hóa của các vector xoắn. Mỗi hệ số mã hóa trong các vector mã hóa $C_i = [C_{i1}, C_{i2}, \dots, C_{ik}]$ là hàm hợp của các từ xoắn d_k và vector xoắn F_i . Các tham số mô hình:

$$c_{ik} = \frac{\exp(-\beta \|F_i - d_k\|_2^2)}{\sum_{k=1}^K \exp(-\beta \|F_i - d_k\|_2^2)}. \quad (1)$$

Các hệ số mã hóa C_{ik} biểu diễn mức độ thành viên của vector xoắn và các từ xoắn. Trong quá trình huấn luyện mô hình, C_{ik} có xu hướng bằng 0 hoặc bao hòa, dẫn đến sự biến mất của mô hình, làm cho mô hình sử dụng giá trị dương của sản phẩm dấu chấm vector:

$$c_{ik} = [\langle F_i, d_k \rangle]_+. \quad (2)$$

Trong đó: $[.]_+$ lấy phần dương, phần âm trả về 0. $[.]_+$ tương tự như hàm ReLU – một hàm chuyển đổi không bão hòa, không tuyến tính. Sau khi kết thúc về sự tương đồng mã hóa trực tiếp, tất cả các vector mã hóa được kết hợp lại.

Hàm chức năng maxout được sử dụng để thực hiện việc chuyển đổi phi tuyến các hình đại diện có kích thước liền kề và biểu diễn được chọn để tạo thành BoVW. BoVW được sử dụng để thay thế các đại diện có liên kết đầy đủ $P = [p_1, p_2, \dots, p_k / 2]$, mô hình tăng cường là bất biến với các biến đổi hình ảnh khác nhau.

Năm 2004, G. Csurka và cộng sự giới thiệu “Visual categorization with bags of keypoints”, lần đầu tiên đề xuất mô hình BoW của lĩnh vực xử lý ngôn ngữ tự nhiên vào phân loại hình ảnh, trong đó: mỗi hình được biểu diễn dưới dạng một tần số từ trực quan biểu đồ để đạt được kết quả phân loại tốt hơn. Mô hình BoVW yêu cầu một thuật toán không giám sát để trích xuất các đặc trưng từ các ảnh trong bộ training data, thuật toán sử dụng như K-Means, phân cụm các tính năng cấp thấp theo số lượng các trung tâm cụm.

Mô hình BoW truyền thống không xem xét các thông tin thị giác giữa các tính năng địa phương, đồng thời, bỏ qua thông tin vị trí của các đặc trưng trong ảnh. Năm 2006, S. Lazebnik và cộng sự giới thiệu “Beyond bags of features: spatial pyramid matching for recognizing natural scene categories”, phân hoạch tọa độ đặc trưng theo mô hình kim tự tháp: chia ảnh thành các vùng được thể hiện dưới dạng một biểu đồ vector.

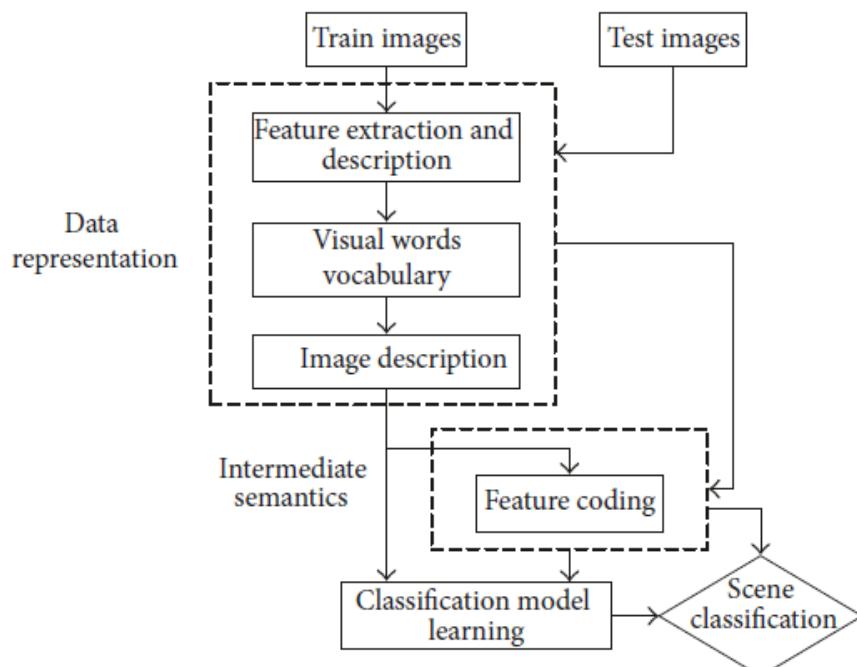


FIGURE 3: Overview of spatial BoVW model in scene classification.

3.6.3. Giai đoạn tiền xử lý

Đối với ảnh đầu vào, áp dụng 2 phương pháp xử lý trước gồm:

- Tính trung bình histogram: tăng độ mượt cho ảnh, cân bằng sáng tối.
- Làm sắc nét bằng hàm Laplacian: tăng độ tương phản cho ảnh.

3.6.4. Cấu hình CNNs

- Kiến trúc mạng: chọn khung học sâu TensorFlow (M. Abadi và cộng sự giới thiệu năm 2015 “TensorFlow: Large-scale machine learning on heterogeneous systems”) để tạo và huấn luyện mô hình CNNs nhằm trích xuất đặc trưng.

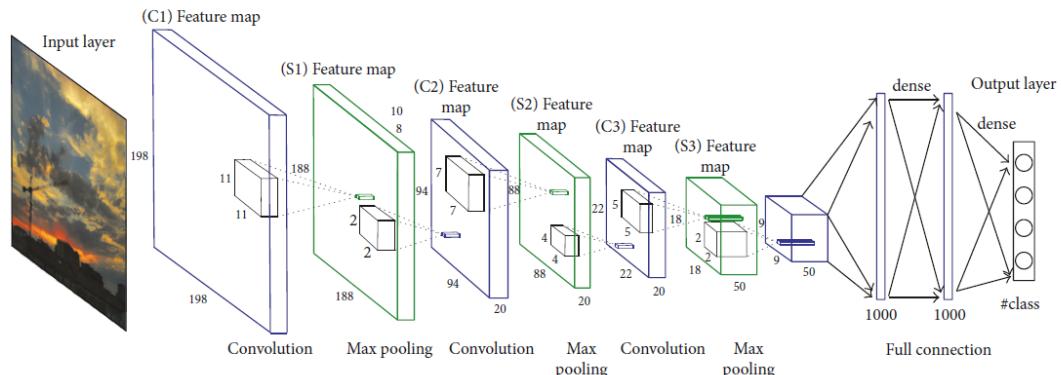


FIGURE 8: The architecture of the proposed CNN.

Ví dụ, ta có kiến trúc mạng như sau: $198 \times 198 - 188 \times 188 - 94 \times 94 \times 20 - 88 \times 88 \times 20 - 22 \times 22 \times 20 - 18 \times 18 \times 50 - 9 \times 9 \times 50 - 1000 - 1000 - M$, M là số lớp học sâu. Mô hình đặc trưng CNNs có 8 lớp:

- 3 lớp Convolution: là lớp xoắn, thiết lập kích thước của trường tiếp nhận là 11×11 , 7×7 , 5×5 – cung cấp đầy đủ thông tin theo ngũ cảnh. Ta có được một bản đồ tính năng bằng cách lặp lại việc áp dụng một chức năng trên các tiêu vùng của toàn bộ hình ảnh, chủ yếu bằng cách tích hợp ảnh đầu vào với một bộ lọc tuyến tính, thêm một thuật ngữ bias và sau đó áp dụng một hàm không tuyến tính.

- 3 lớp Max pooling: là lớp ép cuộn. Ảnh đầu vào được phân chia thành một nhóm các hình chữ nhật không chồng chéo và xác định giá trị lớn nhất cho mỗi tiêu vùng. Sử dụng Max pooling vì: giảm độ tính toán các lớp trên bằng cách loại bỏ các giá trị nhỏ. Kích thước của đại diện trung gian được giảm bởi max pooling vì cung cấp thêm sức mạnh của vị trí.

- 2 lớp cuối Full connection là lớp đầy đủ kết hợp, có 1000 nút theo các lớp xoắn. Lớp cuối cùng là một hồi quy logistic với tối đa mềm – là kết quả đầu ra xác suất trên

mỗi lớp, được định nghĩa như sau:

$$P(y = i | x, W_1, \dots, W_M, b_1, \dots, b_M) = \frac{e^{W_i x + b_i}}{\sum_{j=1}^M e^{W_j x + b_j}}, \quad (3)$$

Trong đó x là đầu ra của tầng kết nối thứ hai, W_i và b_i là trọng số và thành kién của nơron trong lớp này, M là số lớp. Tất cả các trọng số được khởi tạo đến 10^2 và các thành kién được đặt thành 0. Lớp đưa ra xác suất tối đa được lấy như là lớp dự đoán, có thể được mô tả trong phương trình sau (y biểu thị lớp dự đoán):

$$\hat{y} = \arg \max_i P(y = i | x, W_1, \dots, W_M, b_1, \dots, b_M). \quad (4)$$

TensorFlow cung cấp một công cụ trực quan TensorBoard mà có thể được sử dụng để hình dung những thay đổi về entropy chéo trong tập huấn luyện.

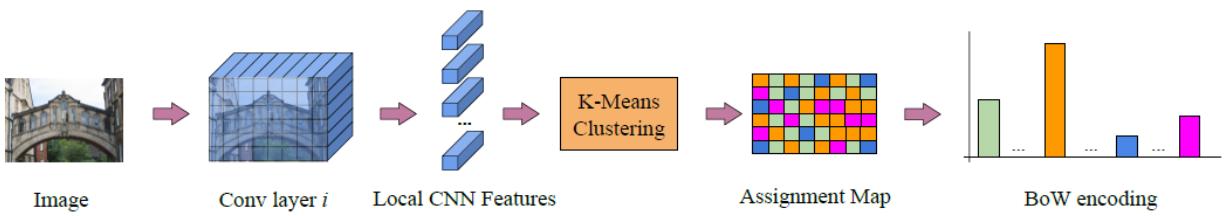
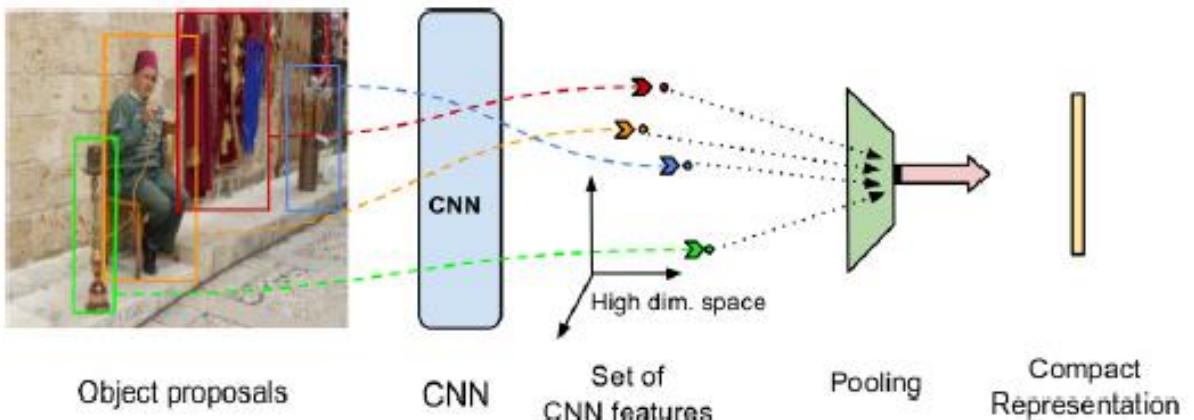


Figure 2: The Bag of Local Convolutional Features pipeline (BLCF).

3.6.5. Phân tích chọn lọc các đặc trưng cuộn xoắn

Mục đích của nghiên cứu này là kết hợp các khả năng học tập đặc trưng mạnh mẽ của CNNs và bản chất bất biến mã hóa của mô hình BoVW. Phần này mô tả cách nhúng mô hình BoVW trong cấu trúc CNNs và chọn lọc đặc trưng phân cấp của CNNs bằng cách tái tạo lại hình ảnh.

Tái tạo lại hình ảnh sử dụng các đặc trưng từ các lớp sâu hơn của mạng có xu hướng để quyết định lớp xoắn sẽ được nhúng vào mô hình BoVW bằng cách sử dụng các thông số tối ưu cung cấp từ lớp xoắn thứ ba của mô hình huấn luyện. Để tái tạo được hình ảnh tốt nhất có thể đảo ngược các đặc trưng từ các lớp – phần thử và nhận được một kết quả tốt nhất. Việc tái tạo lại hình ảnh sử dụng các đặc trưng từ các lớp sâu hơn có xu hướng không giống với hình ảnh ban đầu.



3.6.6. Cài đặt thông số

Sau khi có được các đặc trưng xoắn, ta sử dụng đầy đủ tính bát biến của mô hình BoVW để có được những đặc trưng có tính phân biệt cao cấp hơn. Sau khi nhúng vào mô hình BoVW,

- Chọn kích cỡ từ điển K.
- Xác định hàm Kernel: phân lớp đặc trưng bằng mô hình GMM kiểu xác suất, SVM, Neural Networks. Giả sử chọn thuật toán SVM để phân lớp đặc trưng.

3.6.7. Kết luận

Mô hình BoVW kết hợp với CNNs có khả năng thực hiện tốt việc phân loại từng vùng ảnh tương ứng với các words trong từ điển, nâng cao thời gian cũng như độ chính xác trong truy vấn hình ảnh. Mặt khác, CNNs thực hiện quá trình tiền xử lý ảnh hiệu quả hơn các thuật toán phân lớp khác – phụ thuộc vào các đặc trưng sẵn có và các đặc trưng xử lý thủ công.

Nếu mô hình CNNs sử dụng thêm GPU thì hiệu suất truy vấn hình ảnh sẽ mạnh mẽ hơn.

Các lưới xoắn (Convolutional nets) có thể được sử dụng để phân loại ảnh bằng cách so sánh sự tương đồng trong tìm kiếm ảnh và thực hiện nhận dạng đối tượng trong các cảnh. Mô hình các mạng lưới xoắn – CNNs có thể nhận dạng nhiều đối tượng ở nhiều khía cạnh khác nhau của dữ liệu thị giác. Hiệu quả của CNNs trong nhận diện hình ảnh là một trong những lý do chính khiến các nhà nghiên cứu chú ý đến việc học sâu (deep learning), một số các ứng dụng cụ thể như sau: xe tự lái, máy bay không người lái, robot, các phương pháp điều trị cho người khiếm thị,... CNNs chạy trên GPUs phân tán sử dụng Spark là một trong những công cụ hiệu quả nhất, nhanh nhất hiện nay. Các lưới xoắn ngược lấy và xử lý ảnh như các bộ cảm biến – là các ma trận với các kích thước bổ sung. Công nghệ cho mô hình lớn cho dữ liệu lớn hiện nay là Hadoop, Spark, TensorFlow.

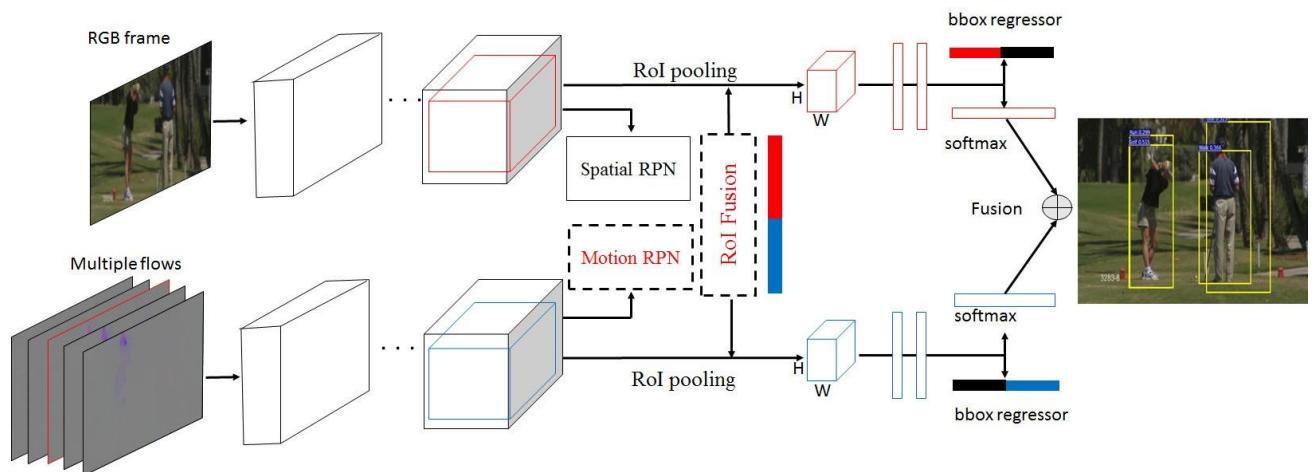
3.6.8. Hướng nghiên cứu mở rộng

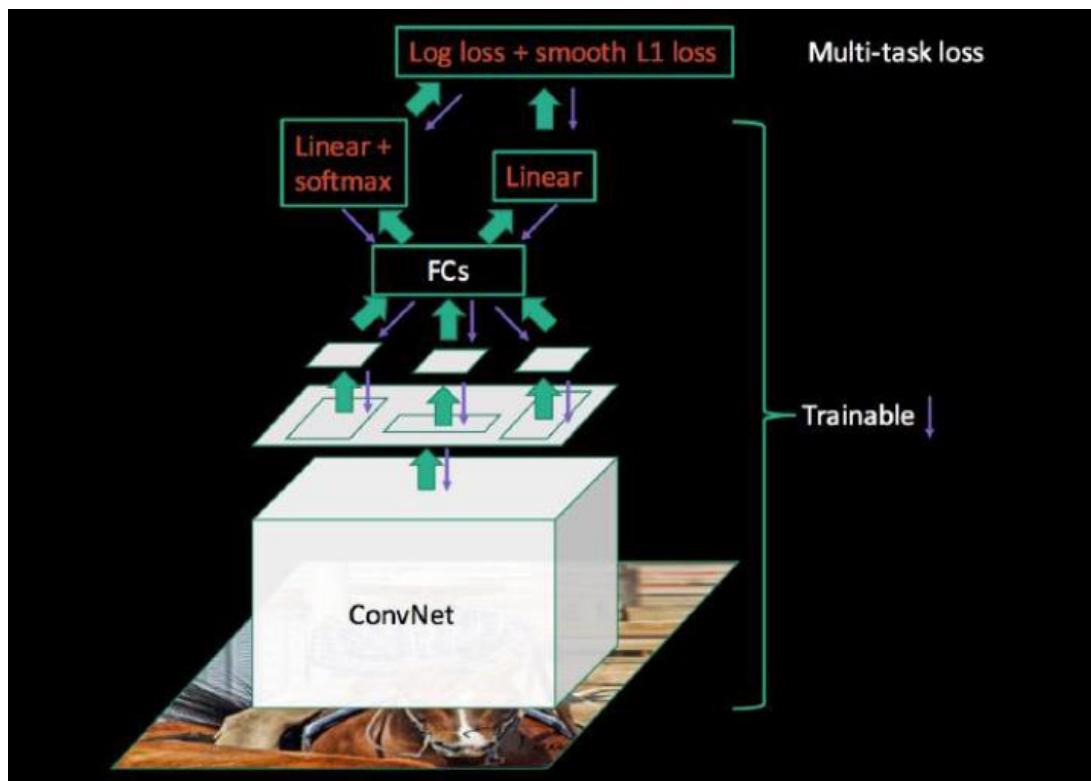
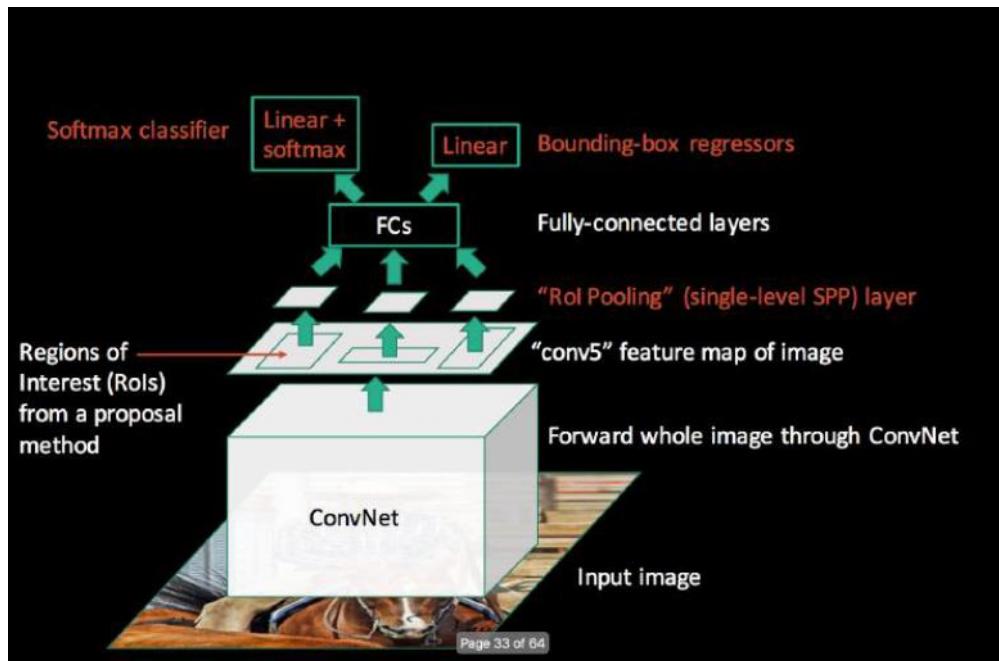
CNNs đã được nghiên cứu đầy đủ trong những năm gần đây, cụ thể: năm 2011, Aggarwal và cộng sự giới thiệu “Human activity analysis: A review”; năm 2015, LeCun và cộng sự giới thiệu “Deep learning”.

Năm 2014, Ross Girshick và cộng sự đã đề xuất phương pháp **Regions with CNN features - R-CNN**, có một bước cải tiến đáng kể cho việc phát hiện đối tượng trong ảnh tĩnh, cụ thể như sau:

- Trích xuất các vùng đề xuất bằng cách sử dụng phương pháp tìm kiếm có chọn lọc.
- Chuẩn hóa các vùng đề xuất về một kích thước cố định để lưu trữ.
- Quá trình phân lớp: dựa vào các đặc trưng rút trích bằng CNNs, sau đó dùng phương pháp học có giám sát, sử dụng thuật toán SVM để phân lớp đối tượng.
- Thực hiện hồi quy tuyến tính trên vùng đề xuất để tạo ra tọa độ giới hạn phù hợp với đối tượng nghiên cứu.

Đối với một ảnh đầu vào, R-CNN có khoảng 2000 vùng đề xuất. Vì vậy, khi dữ liệu đầu vào lớn thì đòi hỏi hệ thống tài nguyên tính toán lớn, tốc độ tính toán cũng phần nào hạn chế. Đến tháng 4/2015, Kaiming He và cộng sự giới thiệu “Spatial pyramid pooling in deep convolutional networks for visual recognition” nhằm cải thiện tốc độ R-CNN bằng **Fast R-CNN (Fast Region-based Convolutional Neural Network)**, cụ thể là loại bỏ giới hạn của kích thước đầu vào cố định với một chiến lược tổng hợp không gian hình chéo (Spatial pyramid pooling - SPP), Fast R-CNN tăng tốc độ R-CNN bằng cách đưa ra một lược đồ tổng hợp các vùng trọng yếu (Regions of Interest – RoI), tất cả các ROI đều sử dụng chung một conv feature map, các lớp huấn luyện và biên hồi quy các hộp giới hạn cùng một lúc.

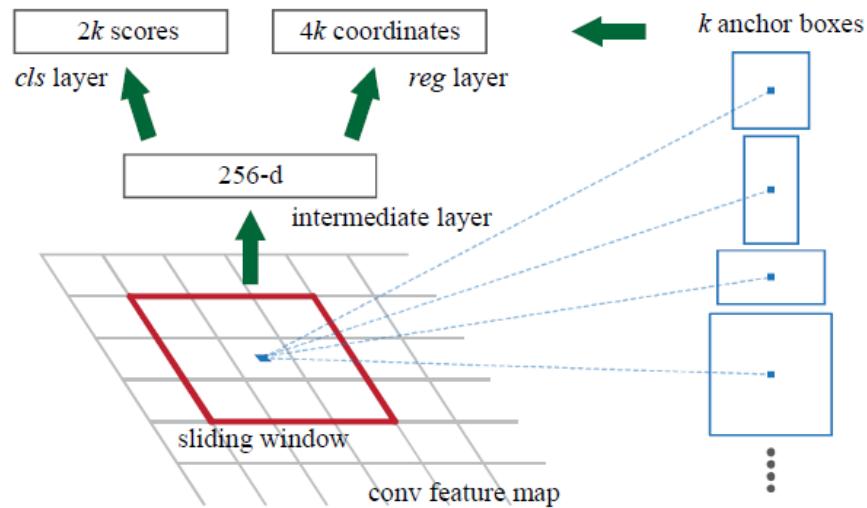




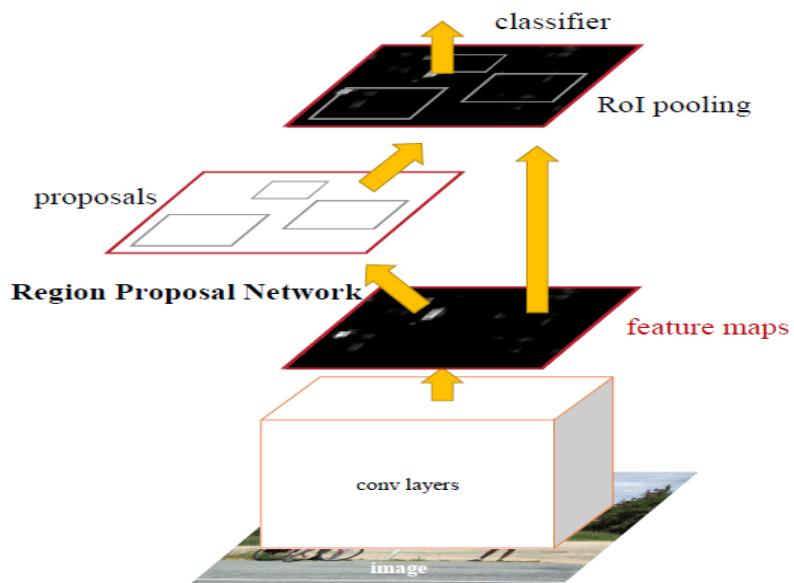
Đến tháng 01/2016, Shaoqing Ren và cộng sự giới thiệu “**Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks**”, nhằm phát hiện đối tượng trong thời gian thực. Faster R-CNN cải tiến từ Fast R-CNN, cụ thể:

- Đầu vào là một ảnh, không cần các vùng đề xuất như R-CNN hay Fast R-CNN, đầu ra là hộp giới hạn các đối tượng trong ảnh và lớp của đối tượng.
- Faster R-CNN = Fast R-CNN + RPN. Trong đó: RPN: Region Proposal Networks. Ý tưởng chính của RPN: loại bỏ sự phụ thuộc vào thuật toán đề xuất khu

vực bên ngoài, thay vào đó là đưa ra đề xuất khu vực từ cùng một CNN, chia sẻ tính năng, phát hiện đối tượng trong một lần. RPN là một mạng xoắn đề xuất các hộp giới hạn và điểm cho hộp giới hạn. RPN sử dụng cửa sổ trượt trên conv feature map, đề xuất k hộp giới hạn và điểm số đánh giá tương ứng cho từng hộp giới hạn. Tại mỗi vị trí cửa sổ trượt, số vùng đề xuất tối đa dự đoán được là k .



- Faster R-CNN cho phép tạo ra các đề xuất nhanh hơn và ít chi phí hơn, hiệu suất phân loại cũng như độ chính xác cũng tốt hơn. Tốc độ của Faster R-CNN nhanh hơn gấp 10 lần Fast R-CNN và 250 lần R-CNN.



Hai biến thể quan trọng của Deep Neural Networks là Autoencoder và Convolution Neural networks (CNNs). CNNs đã gặt hái rất nhiều thành công sau, đặc biệt là trong lĩnh vực computer vision và sau đó có nhiều cải tiến. Do đó, CNNs cũng đã đến lúc bão hòa, nếu có cải

thiện thì cũng không có gì đột biến hơn các cải tiến trước. Để tăng hiệu quả của mô hình BoVW, ta có thể **kết hợp mô hình BoVW và Faster R-CNN**.

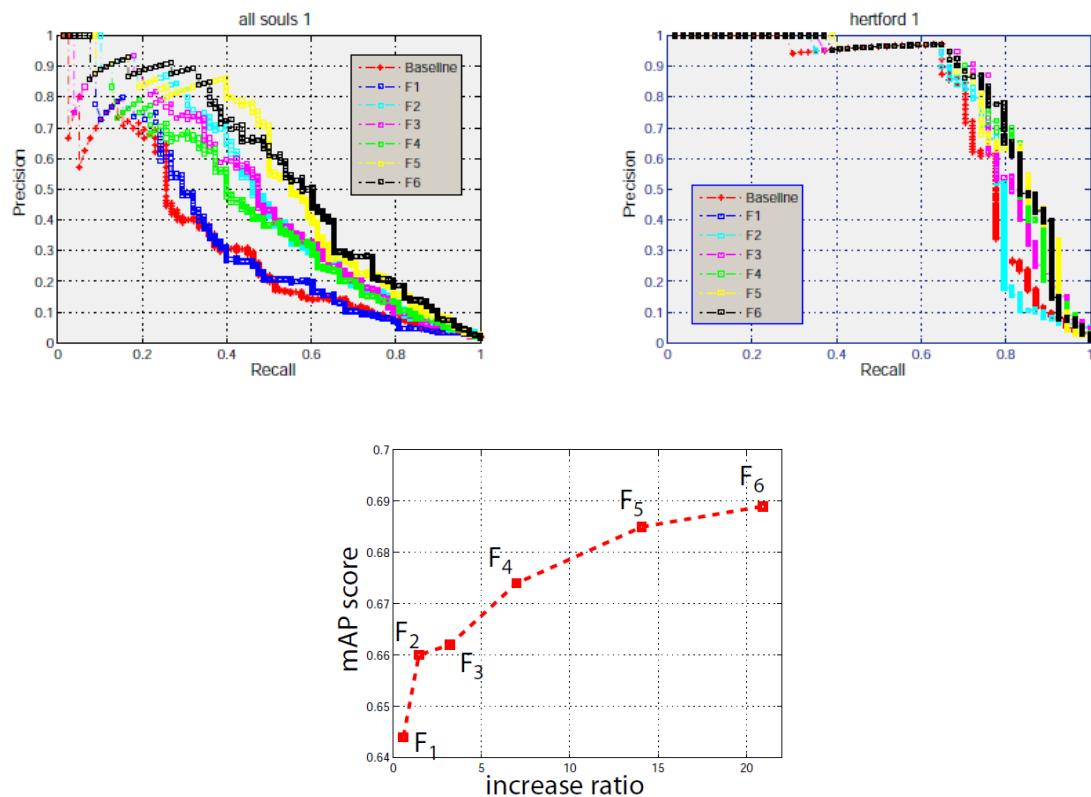
Hướng nghiên cứu tiếp theo, có thể **kết hợp mô hình BoVW với mô hình sinh dữ liệu nhiều tầng (Deep generative models)**.

3.7. Các cách tính độ đo tương đồng

Độ đo tương tự được sử dụng để xác định sự giống nhau giữa 2 ảnh. Các độ đo khác nhau sử dụng để biểu diễn các đặc trưng khác nhau. Trong thực tế, khoảng cách Euclidean được sử dụng rộng rãi nhất, ngoài ra còn có Hellinger, Minkowsky, Manhattan distance, Cosine similarity, Hausdorff, Mallows distance, Kullback-Leibler divergence, Jaccard co-efficient, Minkowski form distance, Mahalanobis distance and histogram intersection, Hamming distance,...

3.8. Các độ đo đánh giá hiệu suất

Đánh giá hiệu năng là công việc rất quan trọng nhằm đánh giá mức độ hiệu quả đạt được của các kỹ thuật hiện hữu để có những cải tiến.



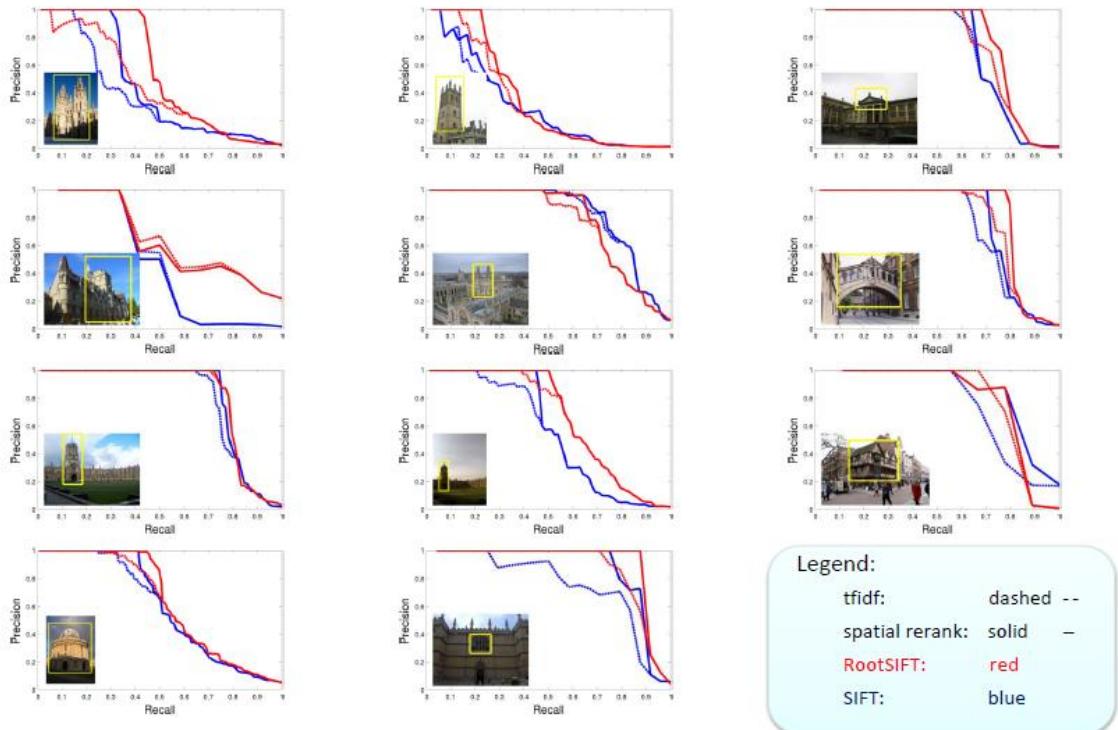
Để đánh giá hiệu quả của phương pháp tìm kiếm ảnh, phần thực nghiệm của luận án tính các giá trị gồm: độ chính xác (precision), độ phủ (recall) và độ đo dung hòa F-measure. Các giá trị thực nghiệm mô tả bằng đường cong recall-precision và ROC. Năm 2015, Kommineni

Jenni và cộng sự giới thiệu “Content Based Image Retrieval Using Colour Strings Comparison” các giá trị hiệu suất được mô tả như sau:

$precision = \frac{ relevant\ images \cap retrieved\ images }{ retrieved\ images }$
$recall = \frac{ relevant\ images \cap retrieved\ images }{ relevant\ images }$
$F - measure = 2 \times \frac{(precision \times recall)}{(precision + recall)}$
$P(c) = \frac{1}{N} \sum_{i=1}^N p(i)$
$MAP = \frac{1}{C} \sum_{j=1}^c P(j)$
R-Precision Rank of the first relevant image Confusion matrix Average Normalised Modified Retrieval Rank

Trong đó:

- Relevant images là tập ảnh tương tự với ảnh tra cứu và có trong tập dữ liệu ảnh.
- Retrieved images là tập ảnh đã tìm kiếm được.
- Các giá trị độ chính xác, độ phủ và F-measure được tính theo tỉ lệ % và quy đổi thành giá trị trên đoạn [0, 1].



Legend:

tfidf: dashed --
 spatial rerank: solid - —
RootSIFT: red
SIFT: blue

CHƯƠNG 4. CÀI ĐẶT CHƯƠNG TRÌNH TRUY VẤN ẢNH THỦ NGHIỆM

Chương trình chạy trên nền Eclipse Modeling Oxygen, ngôn ngữ lập trình Java, được thực nghiệm trên máy tính có bộ xử lý Intel(R) CoreTM i7-4500U, CPU 1,80GHz 2,40GHz, RAM 8GB, HDD 1TB và hệ điều hành Windows 10 Professional.

Sử dụng tập dữ liệu Oxford Building như mô tả ở Chương 3.

4.1. THỦ NGHIỆM 1

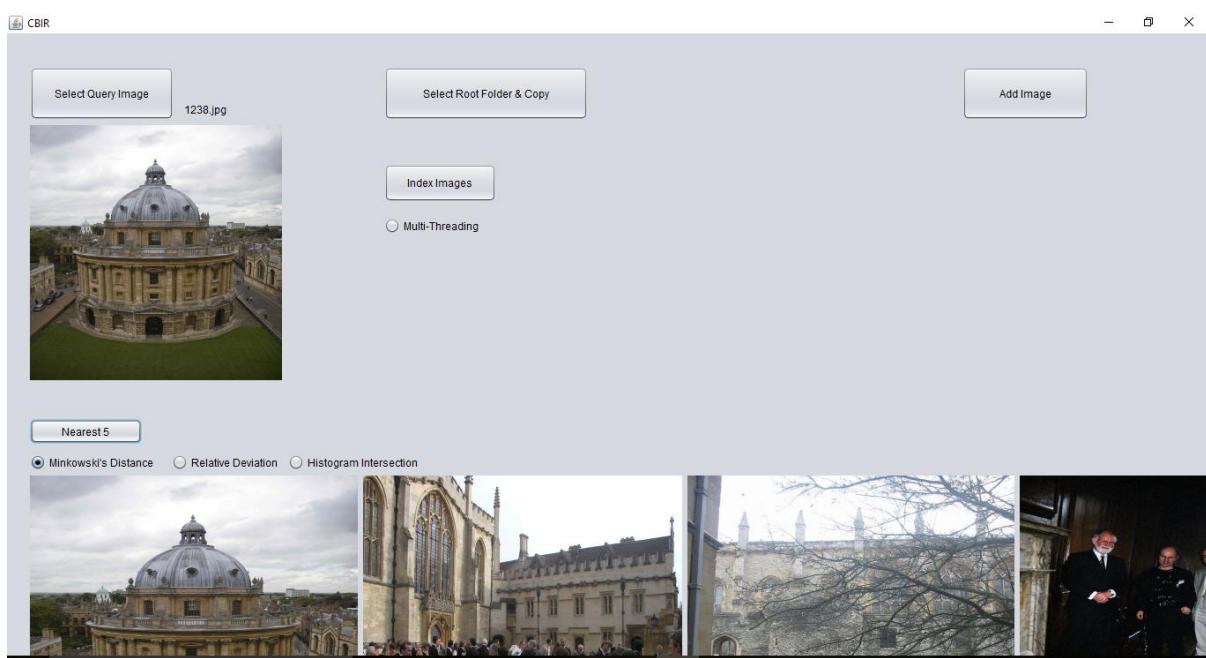
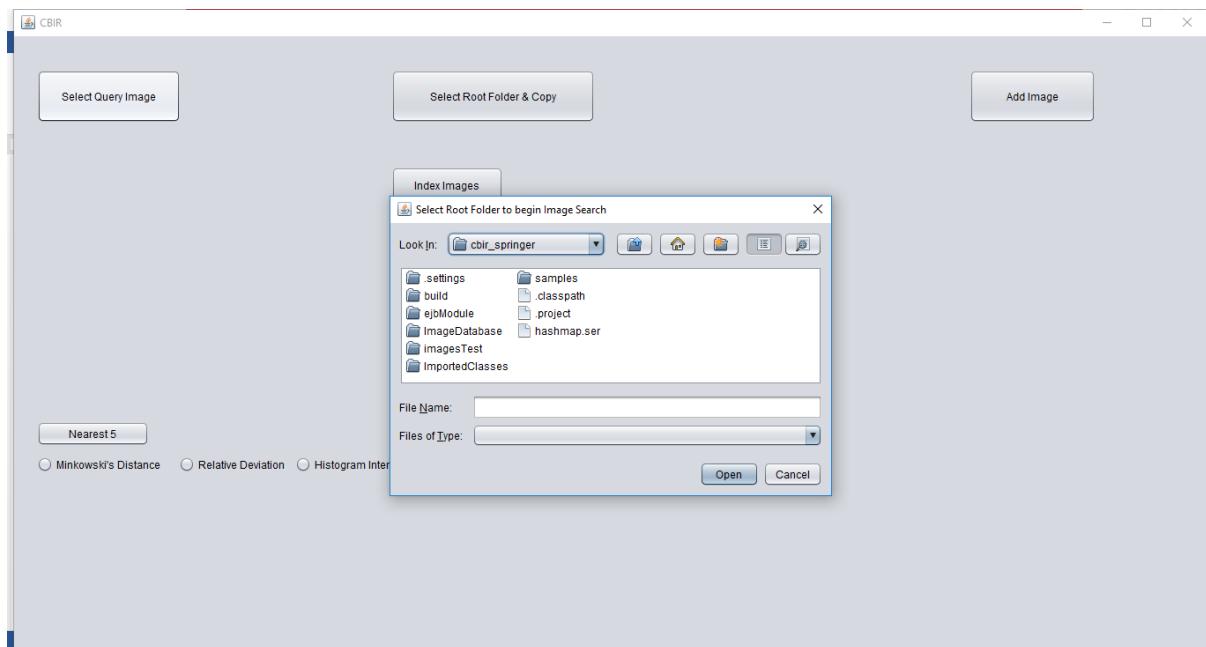
Chương trình truy vấn hình ảnh theo đặc trưng lược đồ màu, theo khoảng cách Minkowsky, histogram intersection, relative deviation.

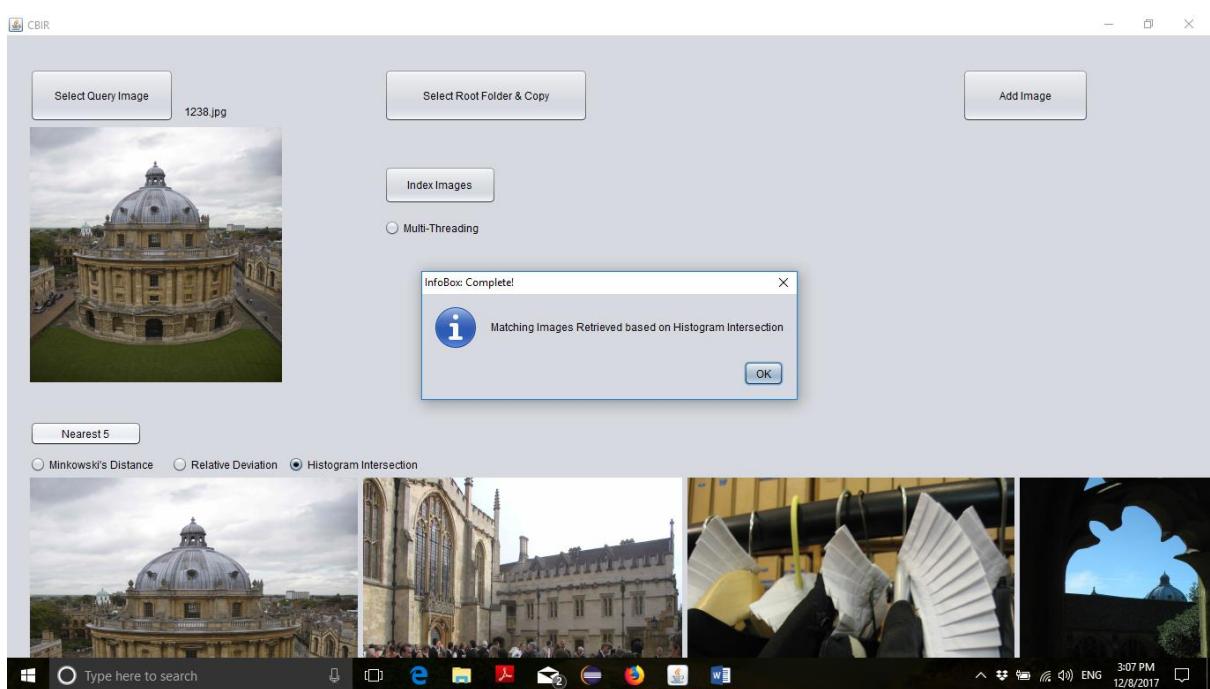
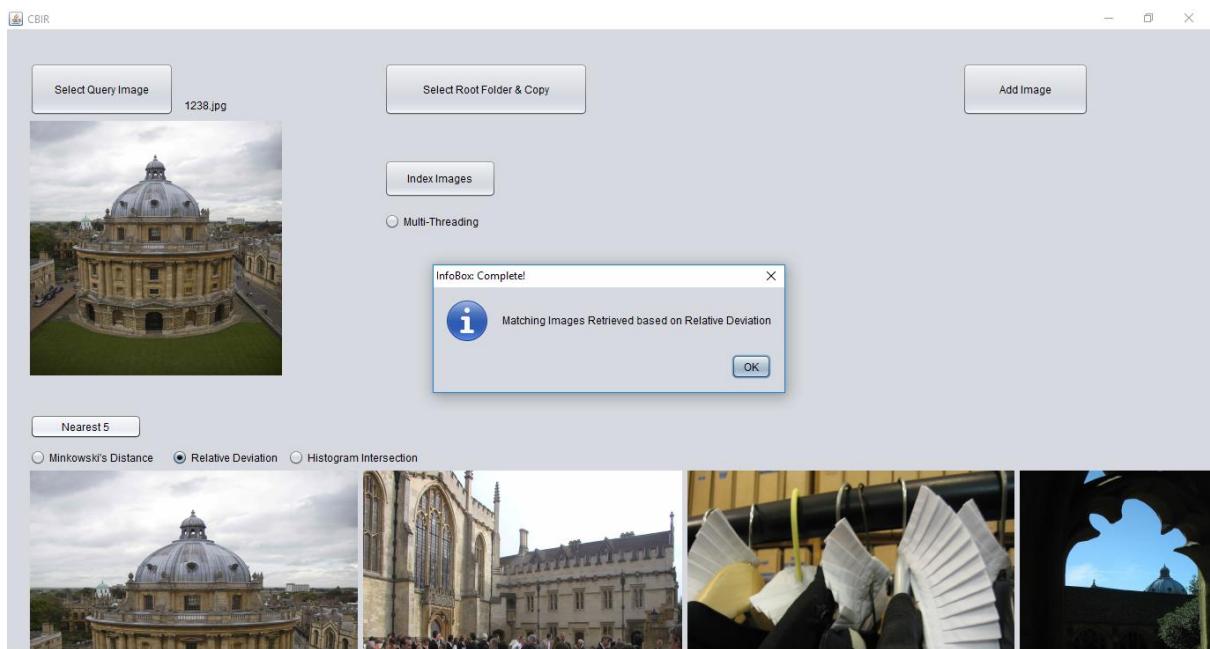
Thực nghiệm chỉ đánh giá hiệu năng về độ chính xác tra cứu trong các kết quả top 5 hình ảnh. Độ chính xác của chương trình thử nghiệm này thấp hơn so với chương trình được mô tả ở Chương 3. Tuy nhiên, chương trình này đơn giản hơn, chạy nhanh hơn và cho phép thêm ảnh mới vào bộ datasets.

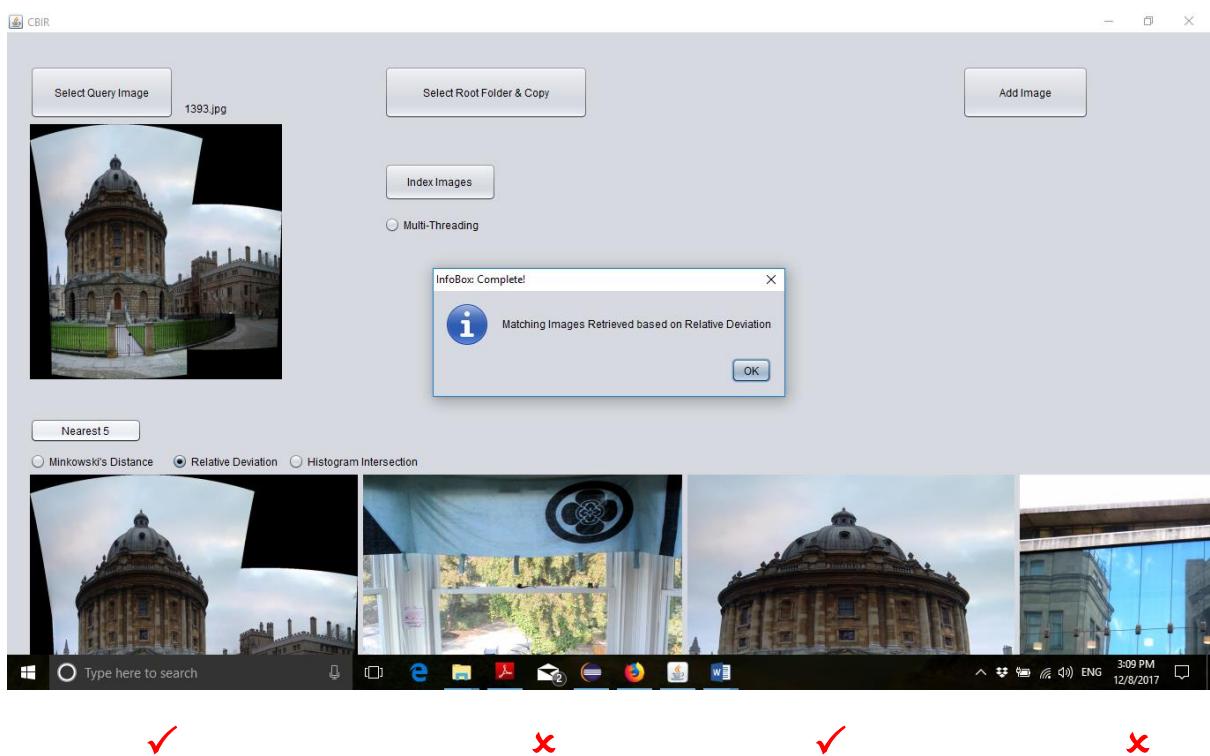
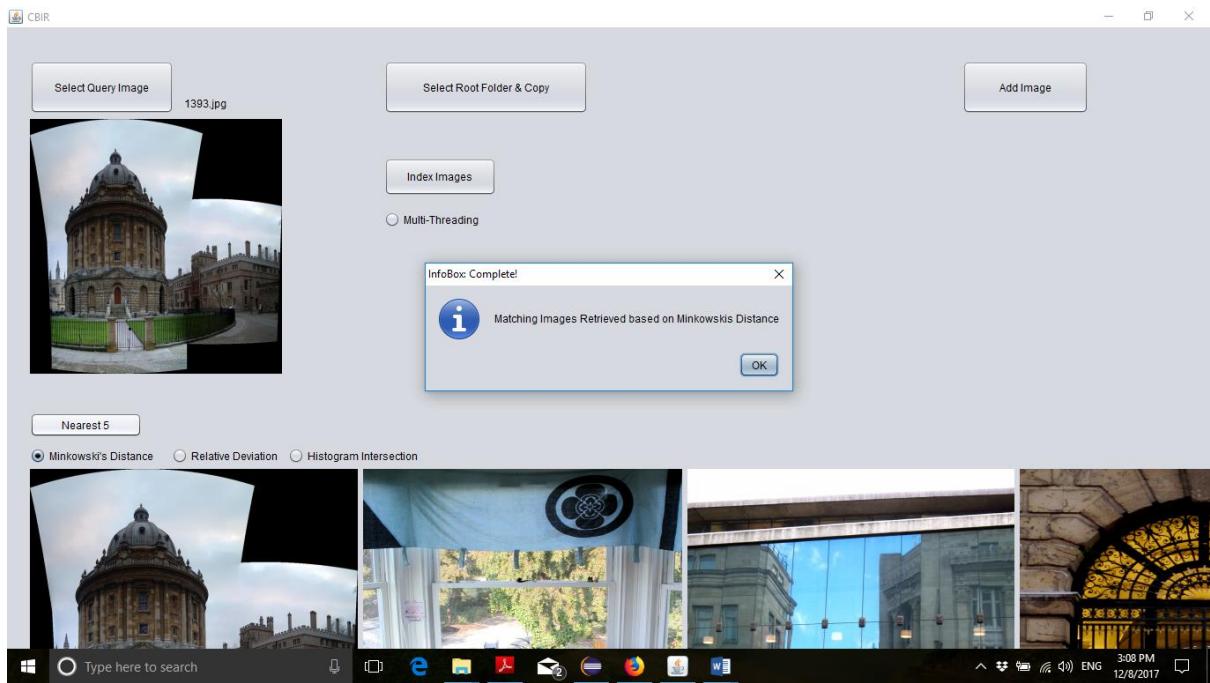
Source code (file đính kèm). Kết quả tra cứu của chương trình như sau:

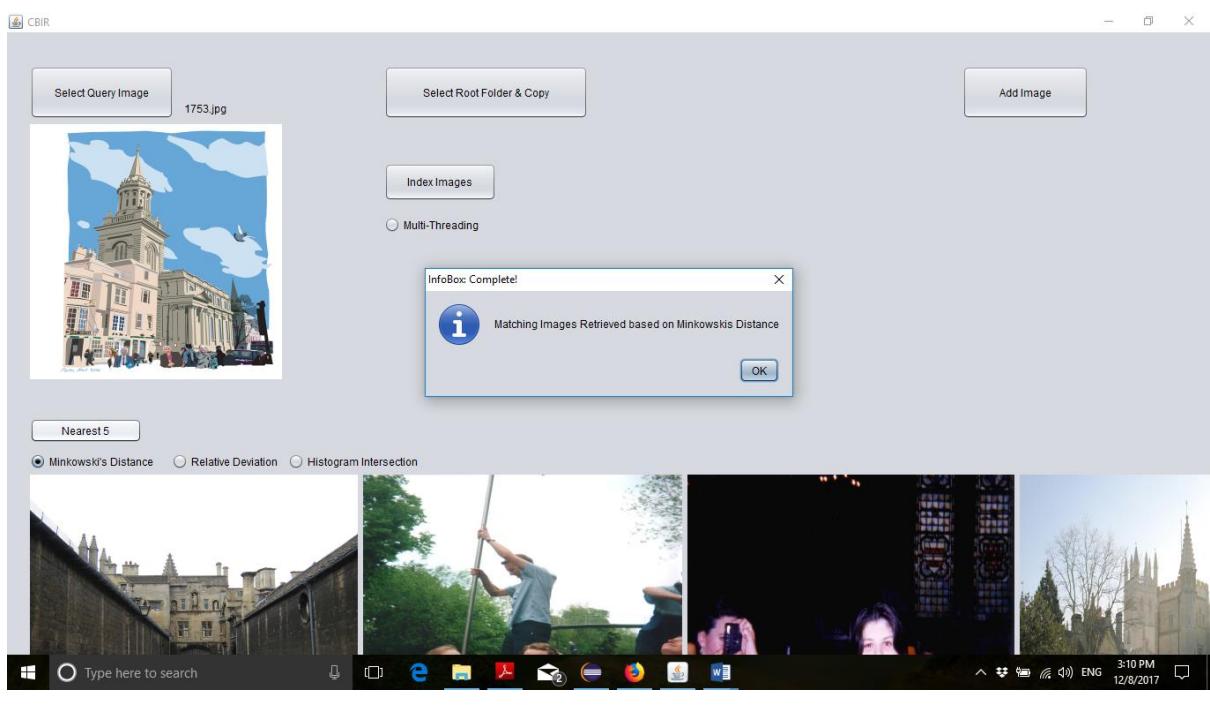
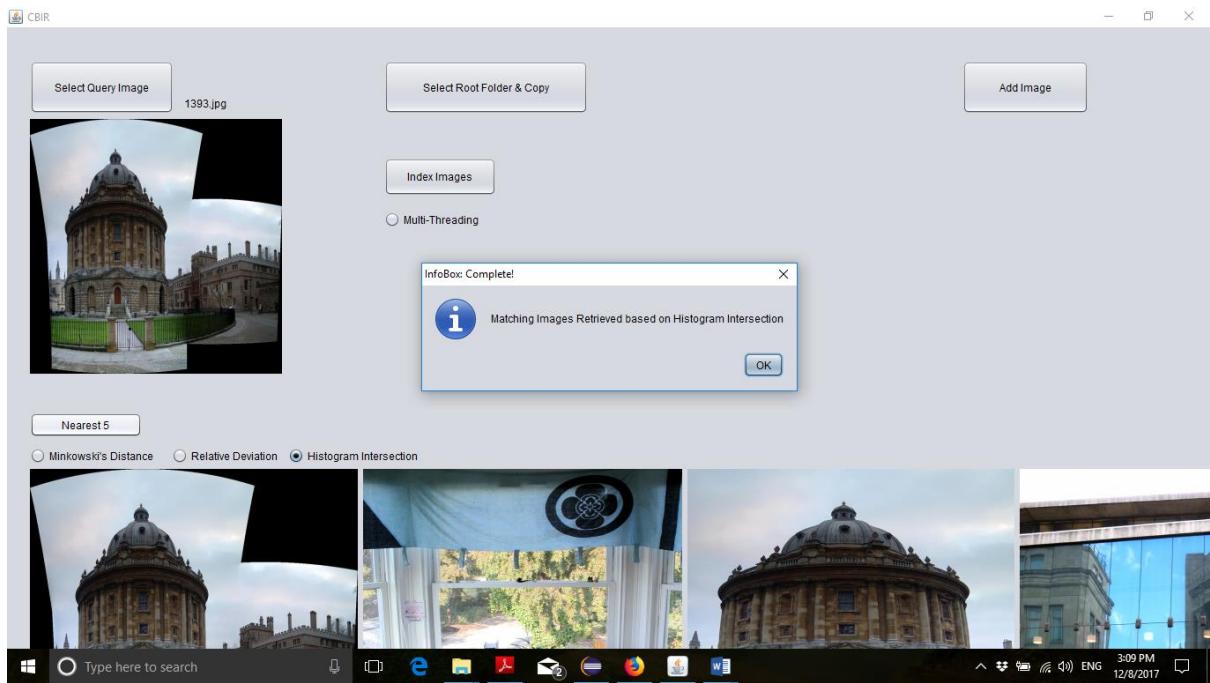
4.1.1. Sử dụng bộ datasets Oxford Building như mô tả ở Chương 3

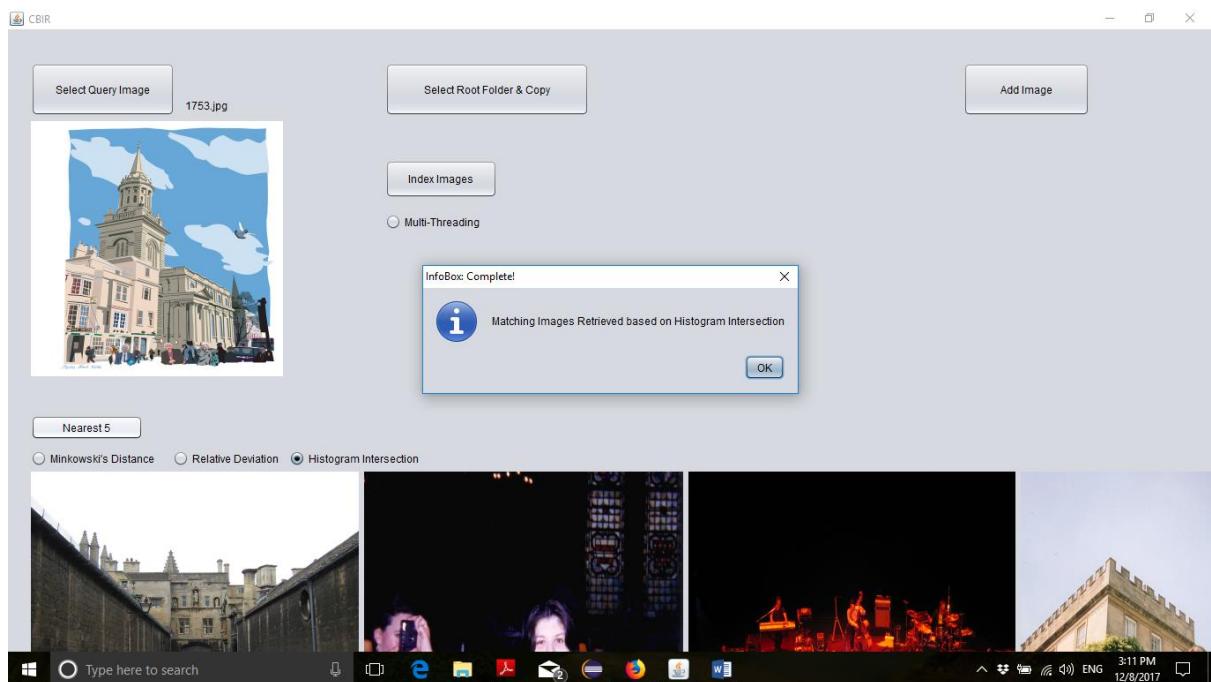
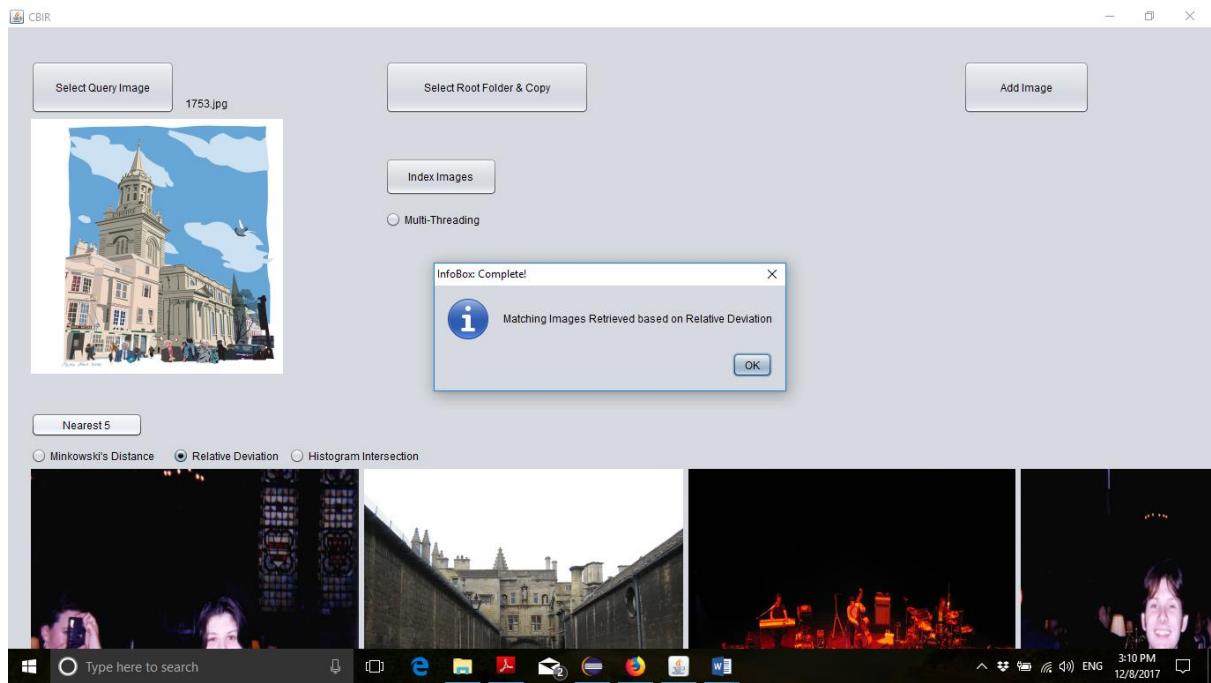


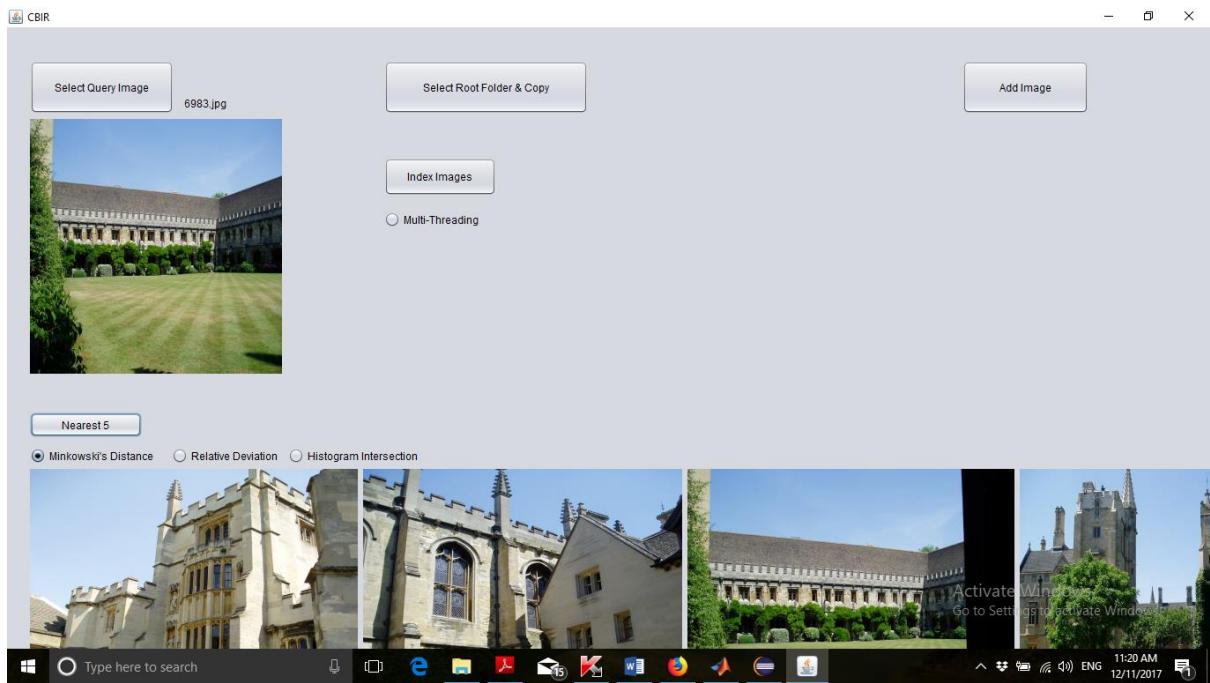




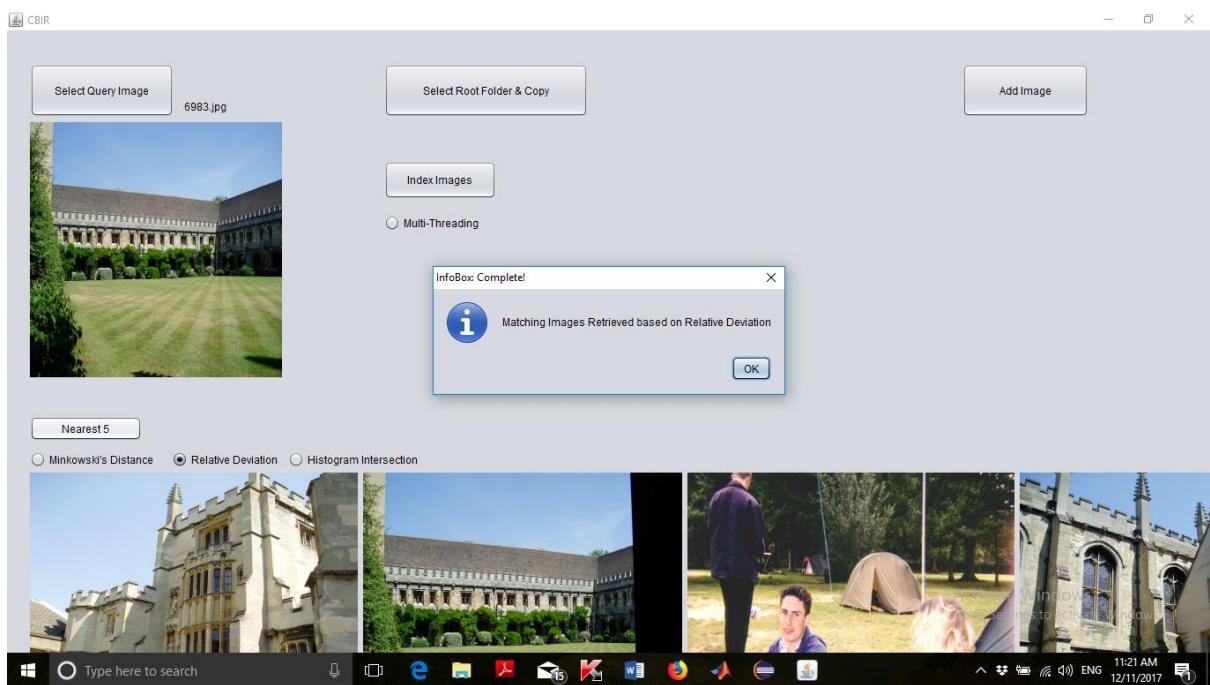




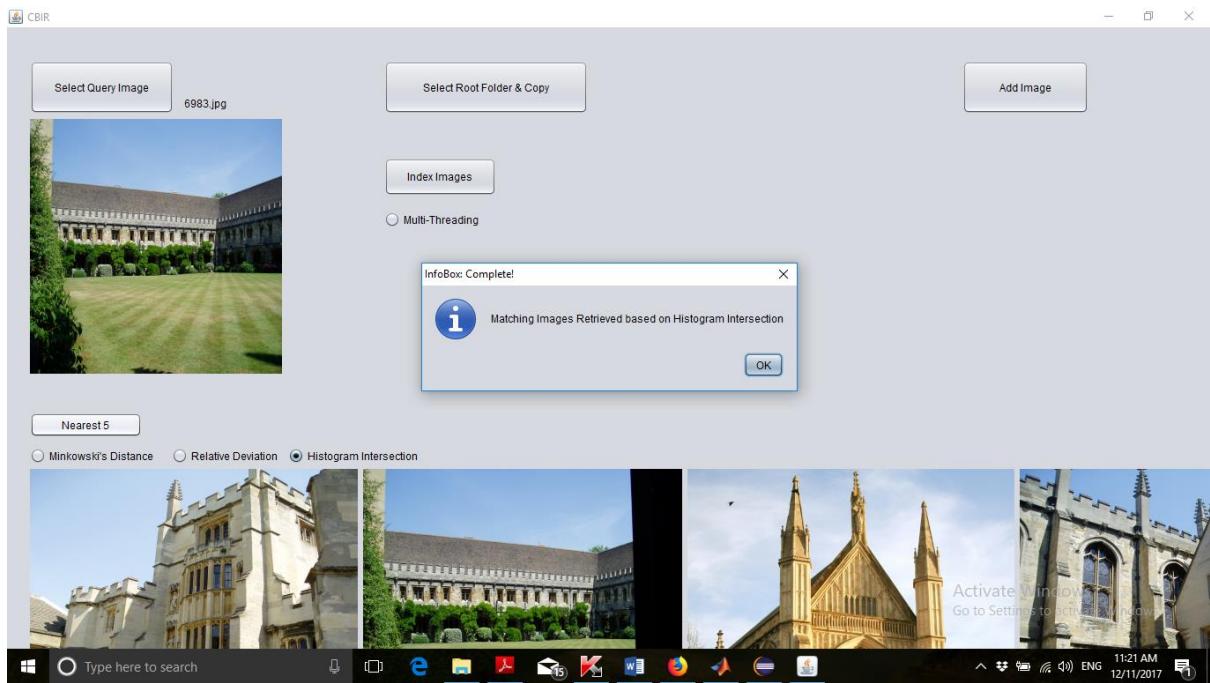




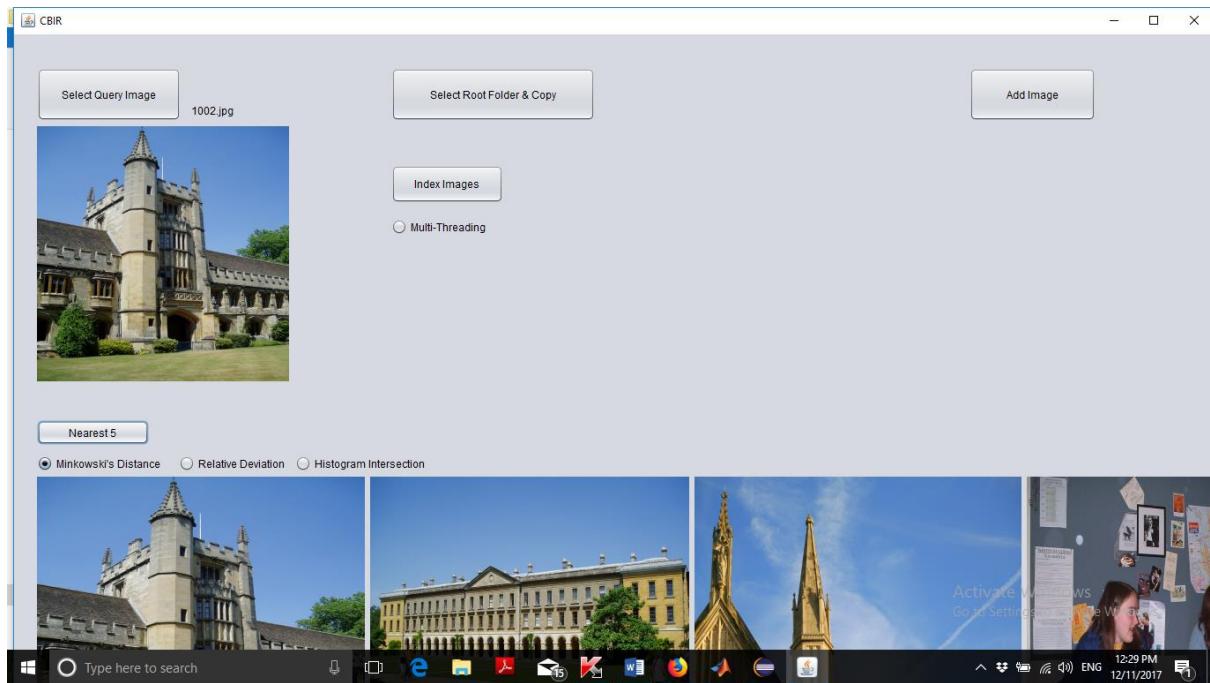
✗ ✗ ✓ ✗



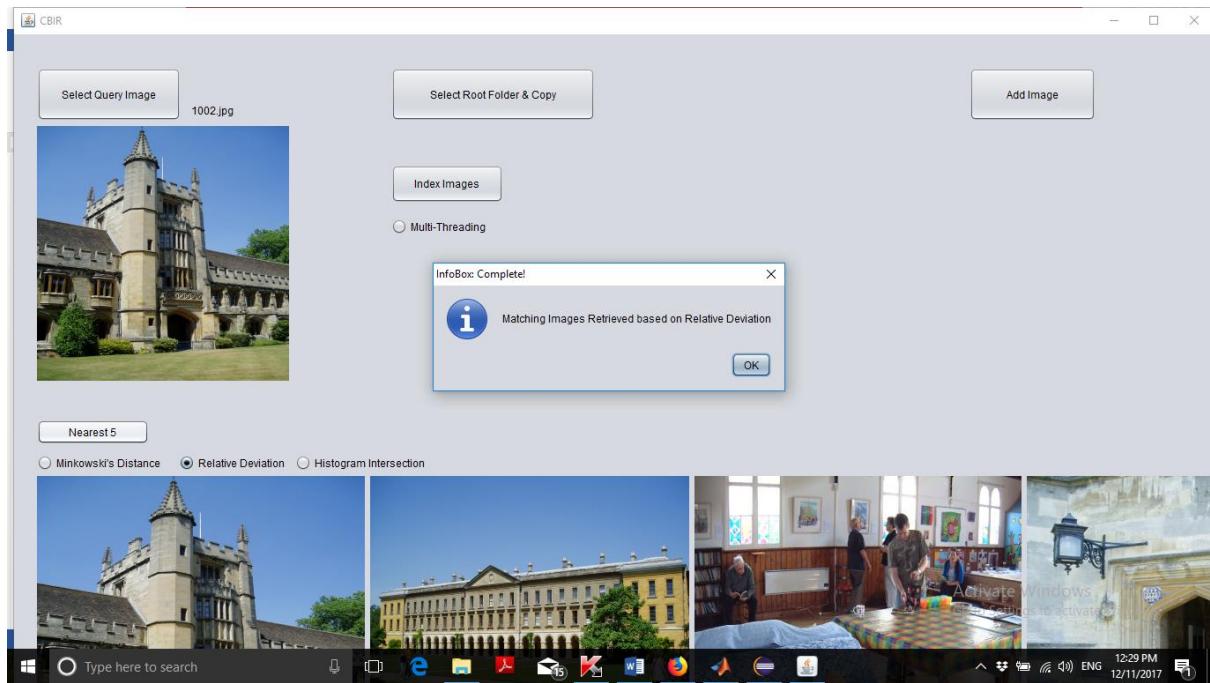
✗ ✓ ✗ ✗



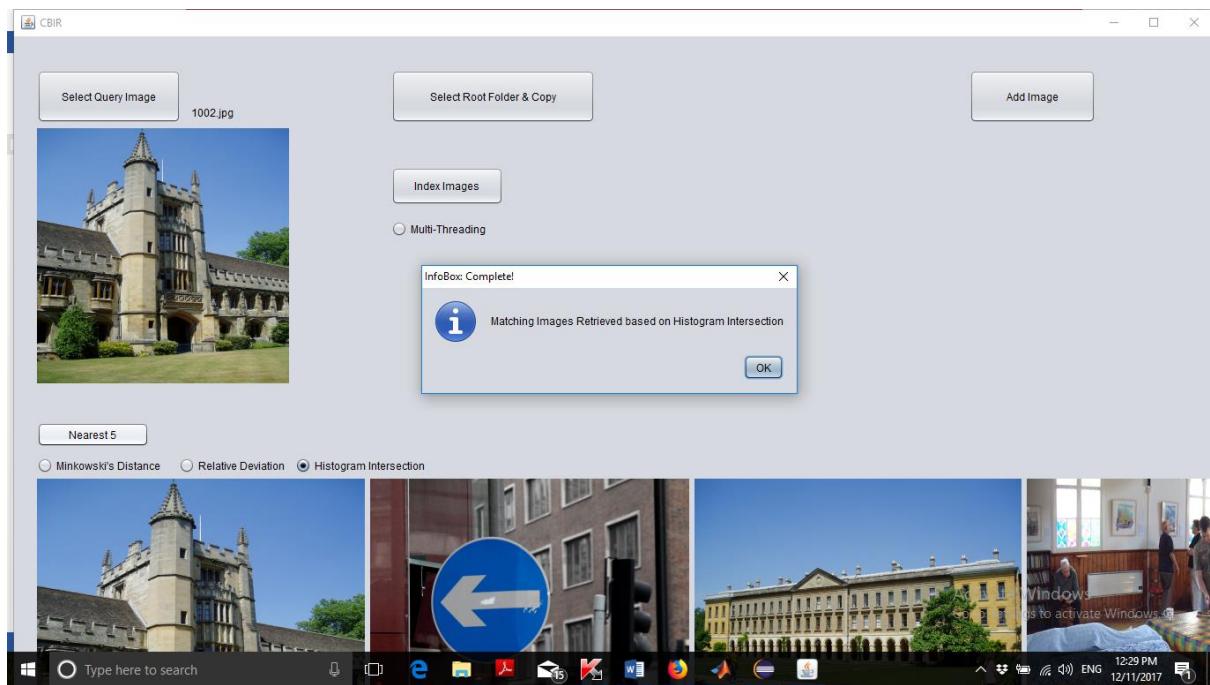
✗ ✓ ✗ ✗



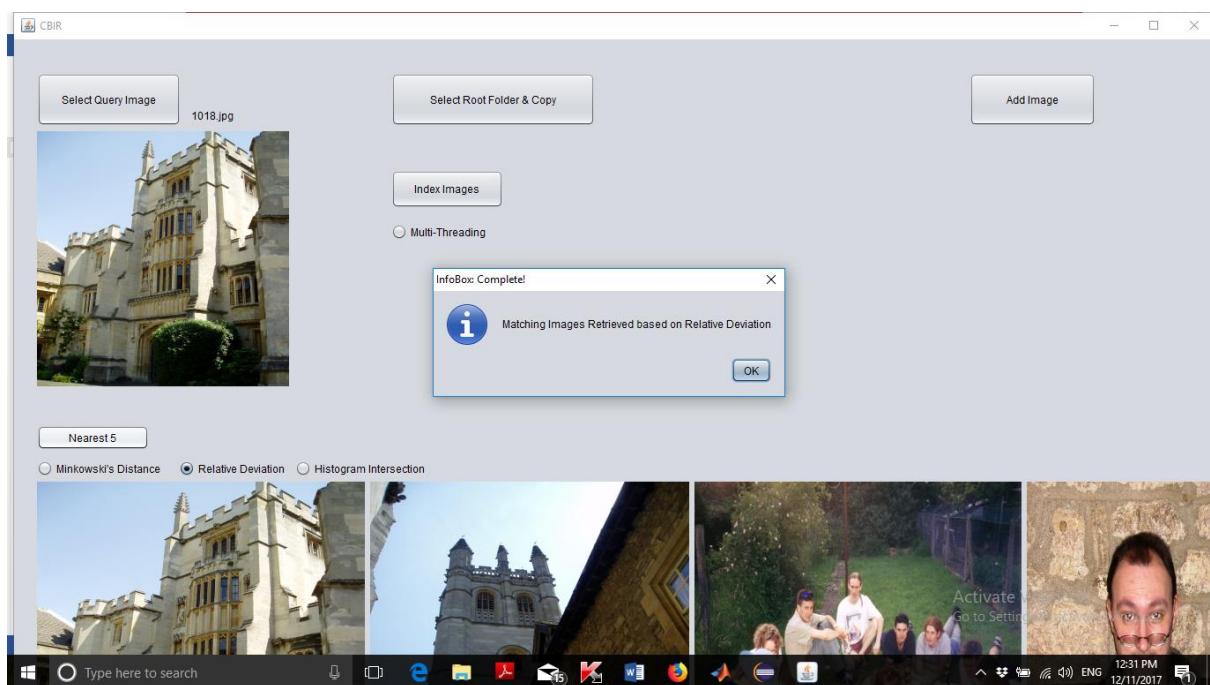
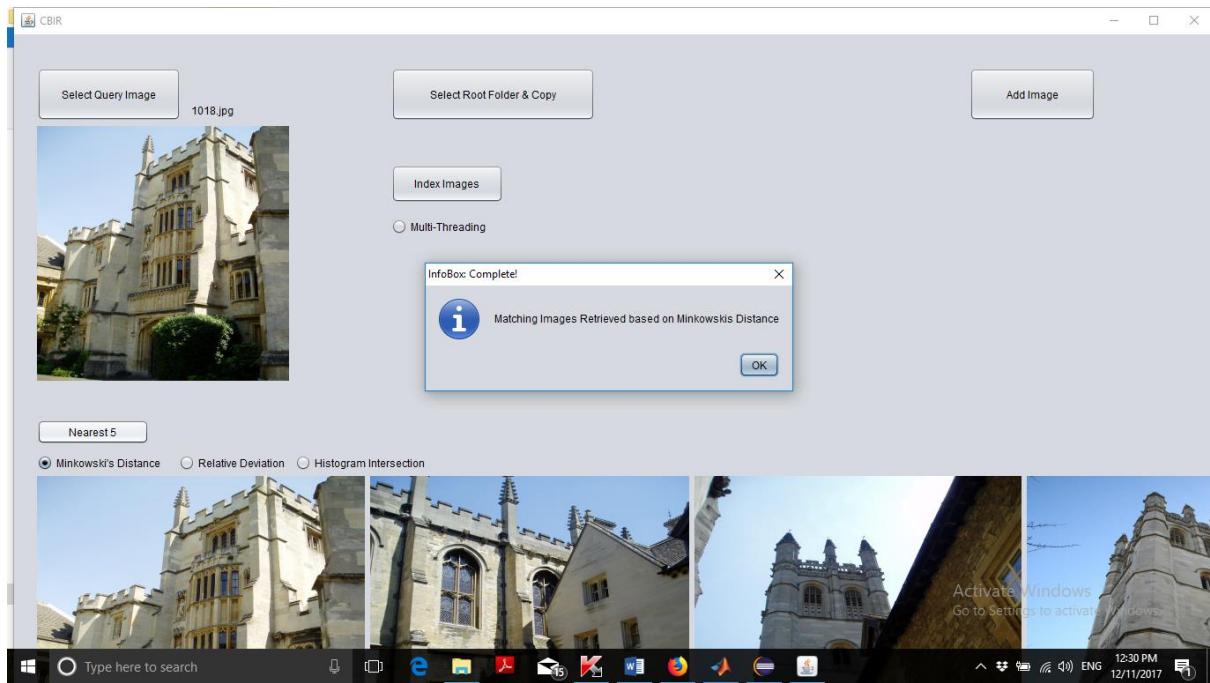
✓ ✗ ✗ ✗

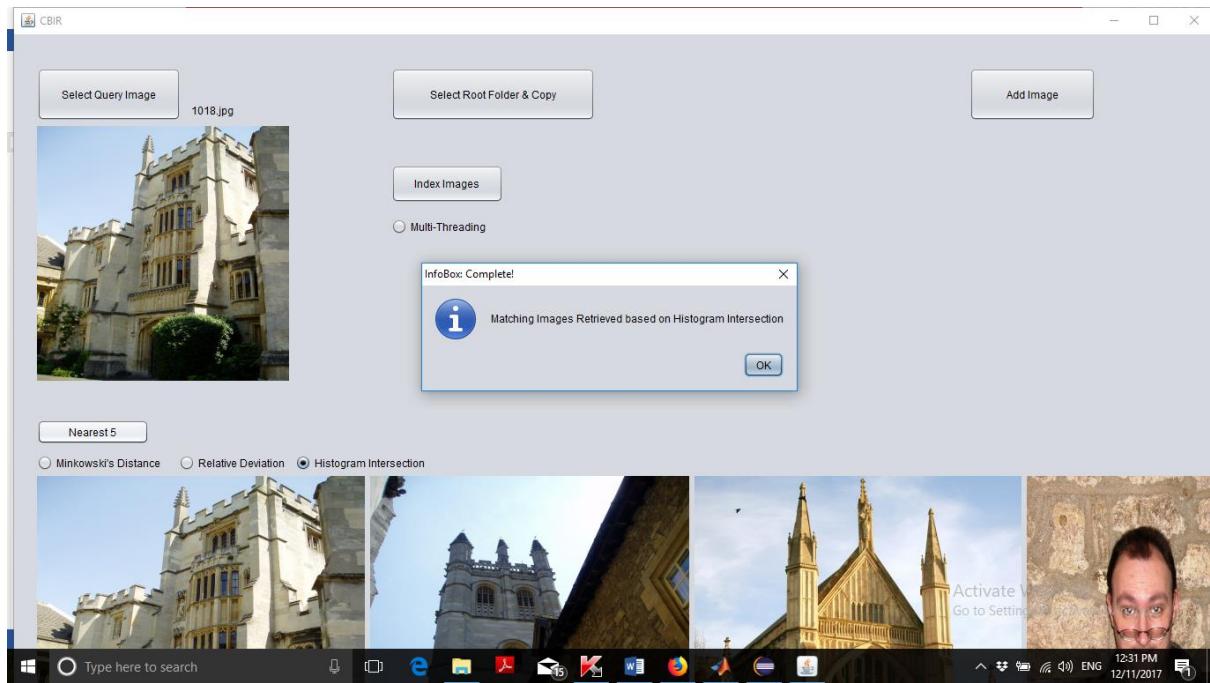


✓ ✗ ✗ ✗



✓ ✗ ✗ ✗

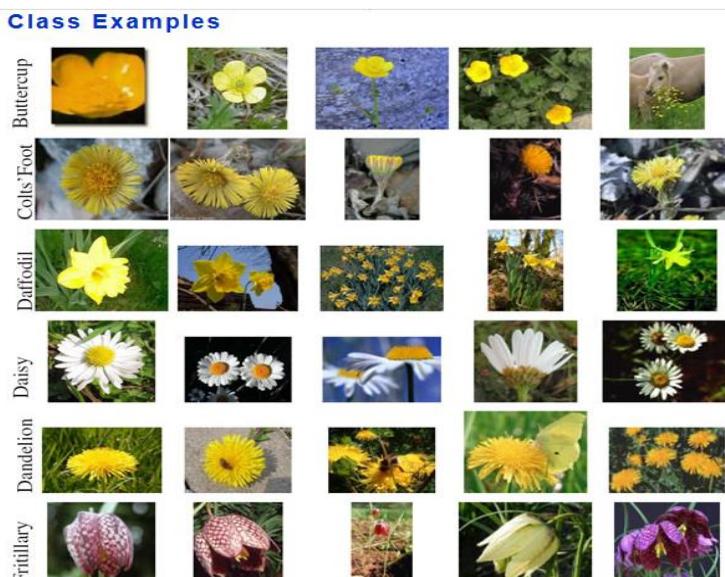




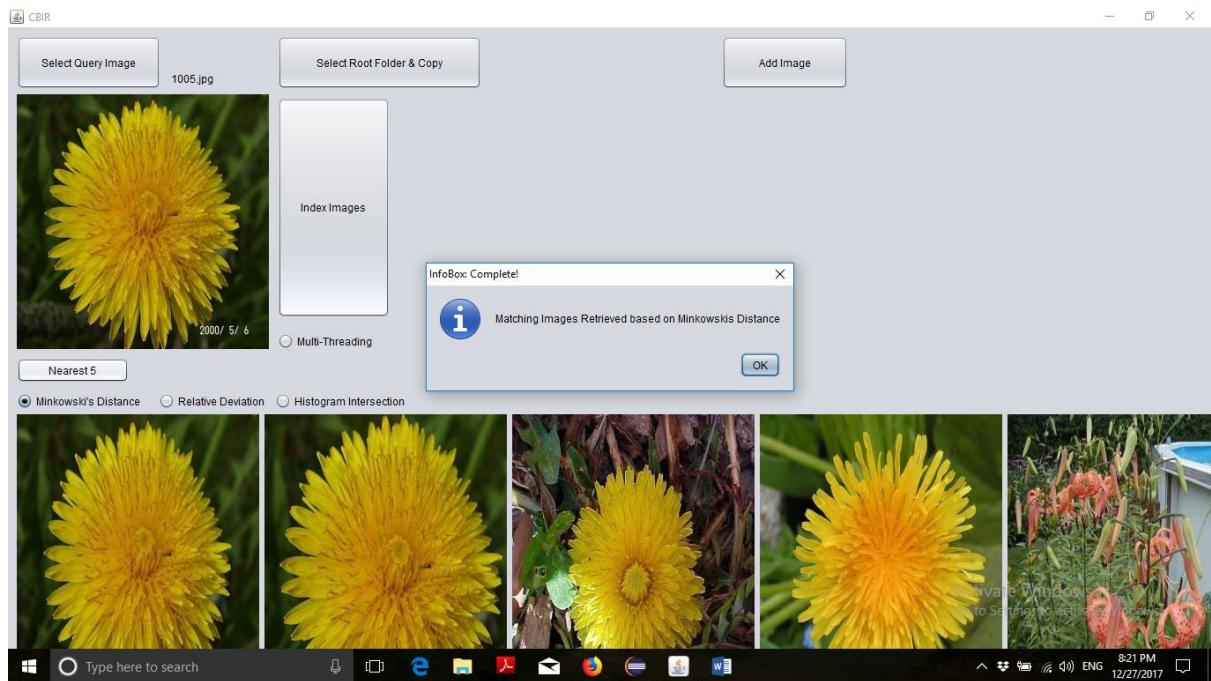
4.1.2. Sử dụng bộ datasets flowers

<http://www.robots.ox.ac.uk/~vgg/data/flowers/17/index.html>

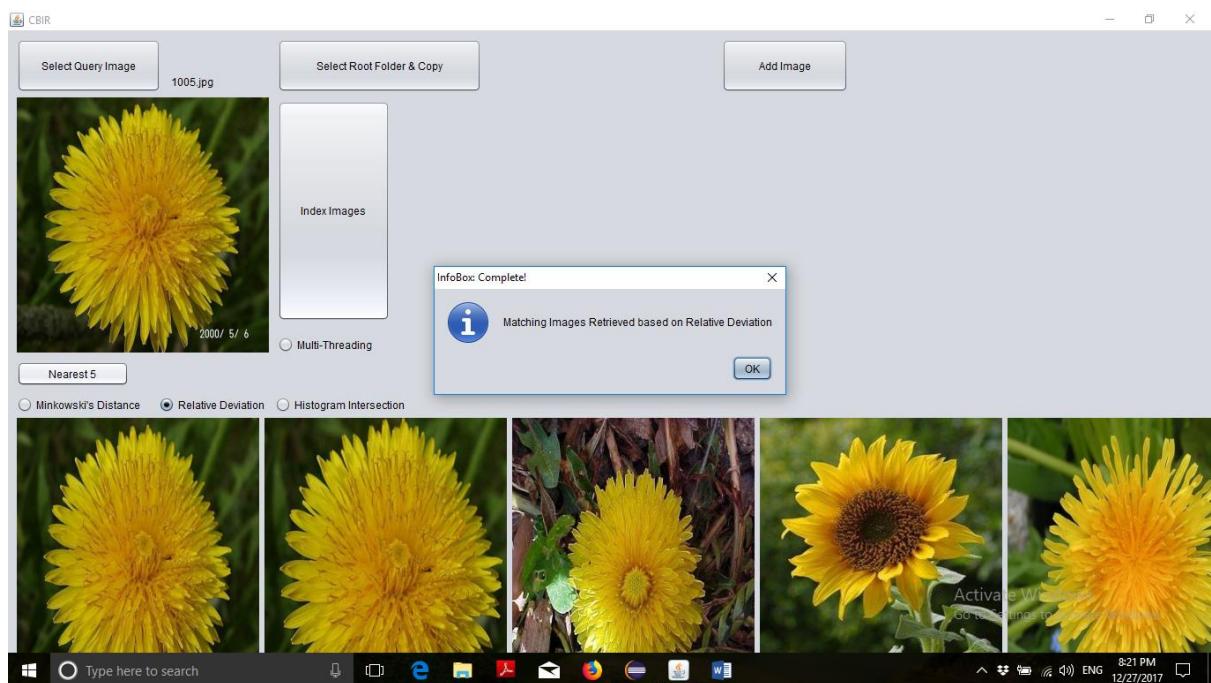
Bộ dữ liệu hoa 17 loại với 80 hình ảnh cho mỗi lớp. Hoa được chọn là một số hoa phổ biến ở Anh. Các hình ảnh có các biến thể quy mô lớn, hiện tại và ánh sáng và cũng có các lớp có nhiều biến thể của hình ảnh trong lớp và gần giống với các lớp khác. Các thể loại có thể được nhìn thấy trong hình bên dưới. Chia ngẫu nhiên bộ dữ liệu thành 3 nhóm đào tạo, xác nhận và thử nghiệm khác nhau. Một tập hợp con của các hình ảnh đã được groundtruth dán nhãn cho phân khúc.



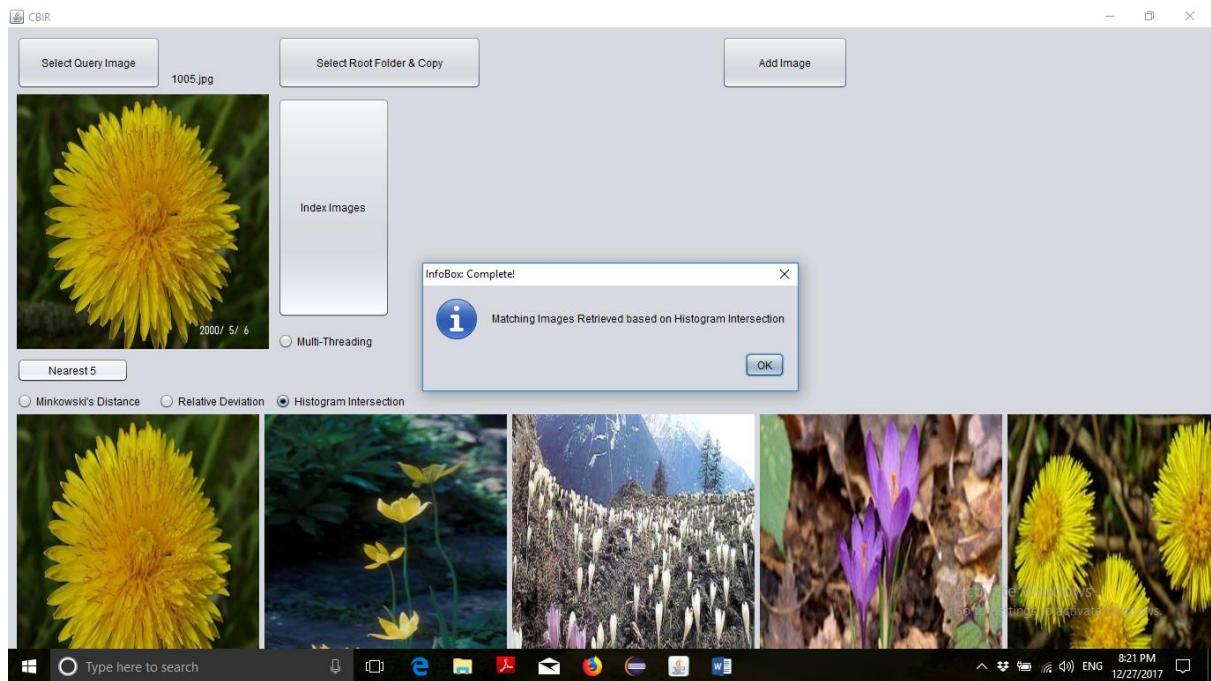




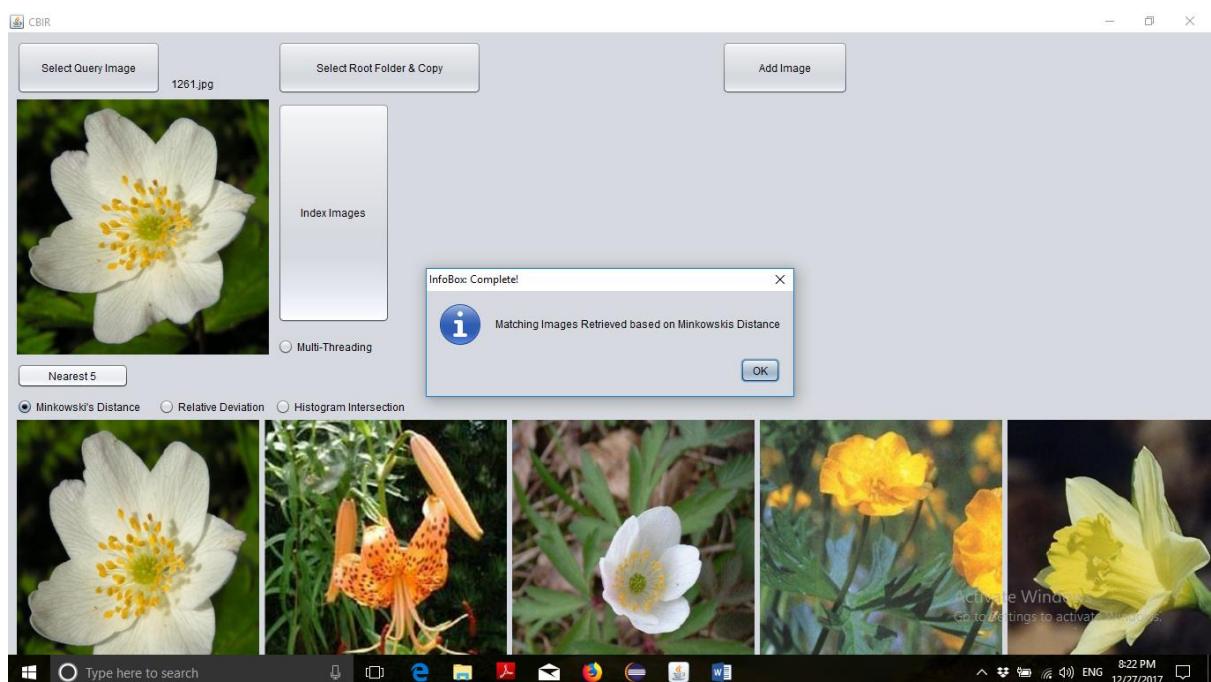
✓ ✓ ✓ ✓ ✗



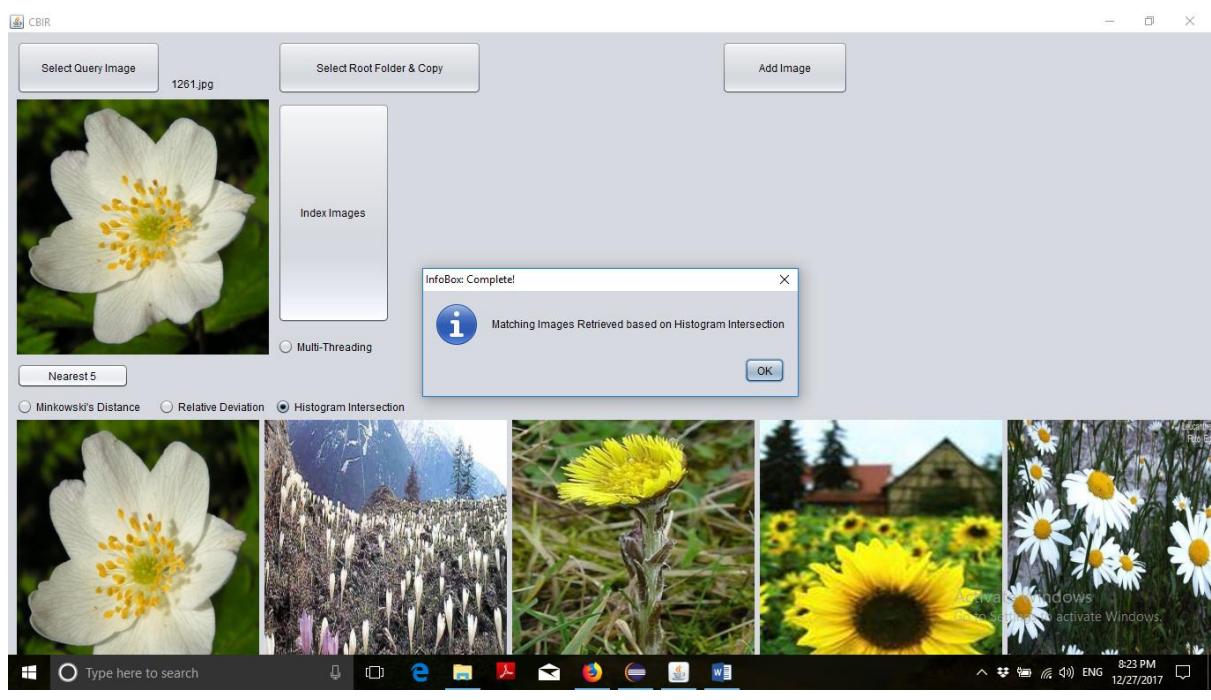
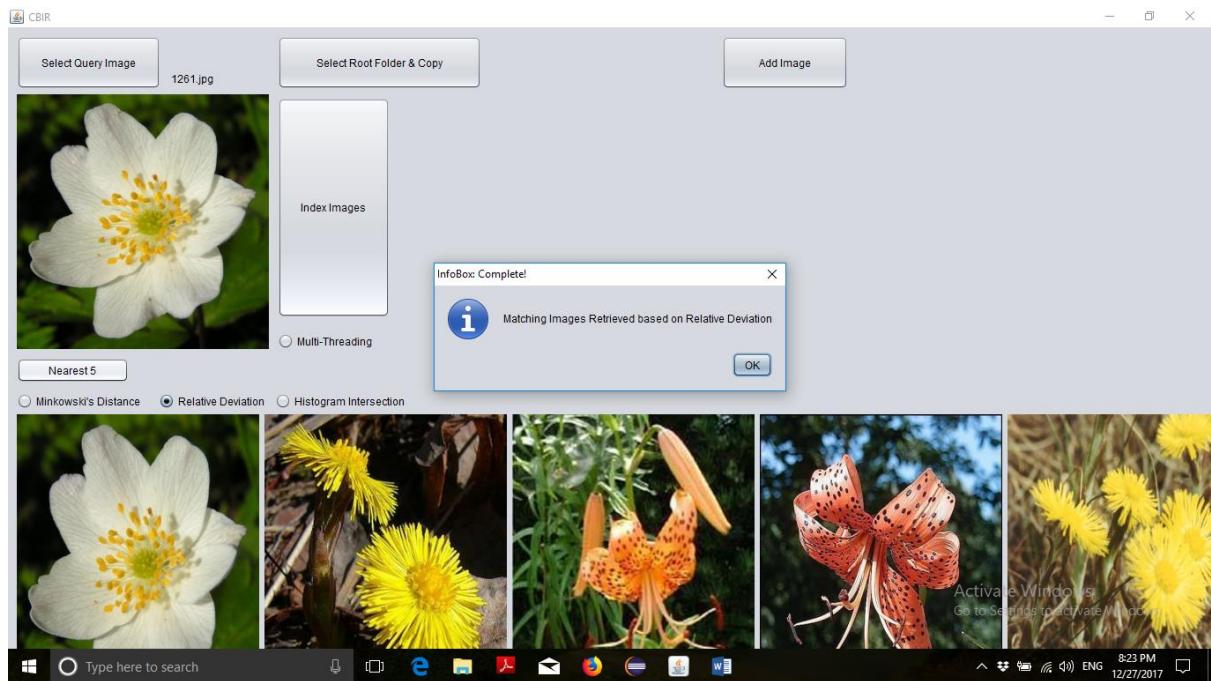
✓ ✓ ✓ ✗ ✓

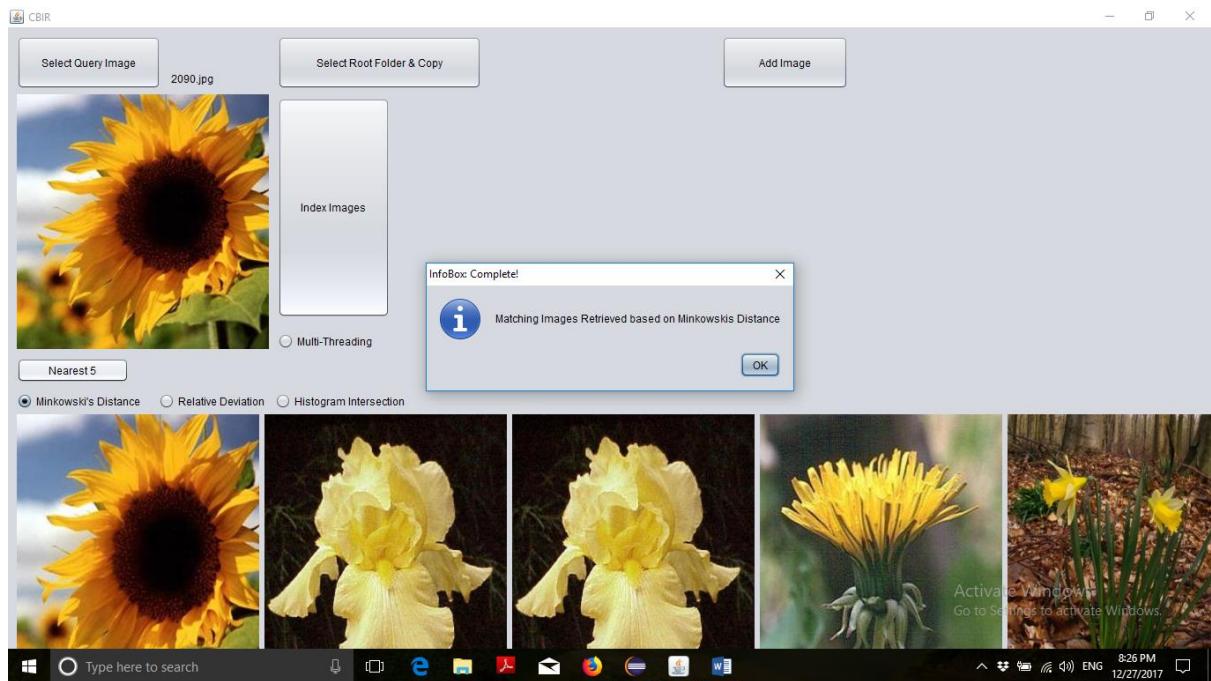


✓ ✗ ✗ ✗ ✓

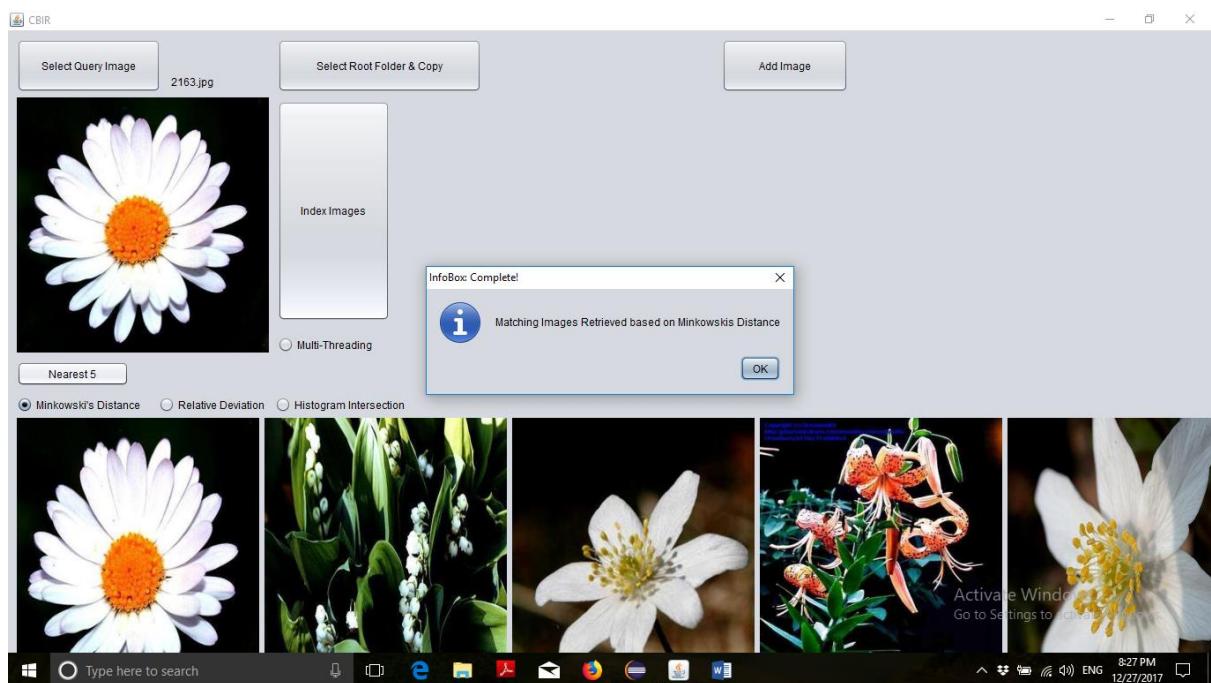


✓ ✗ ✓ ✗ ✗

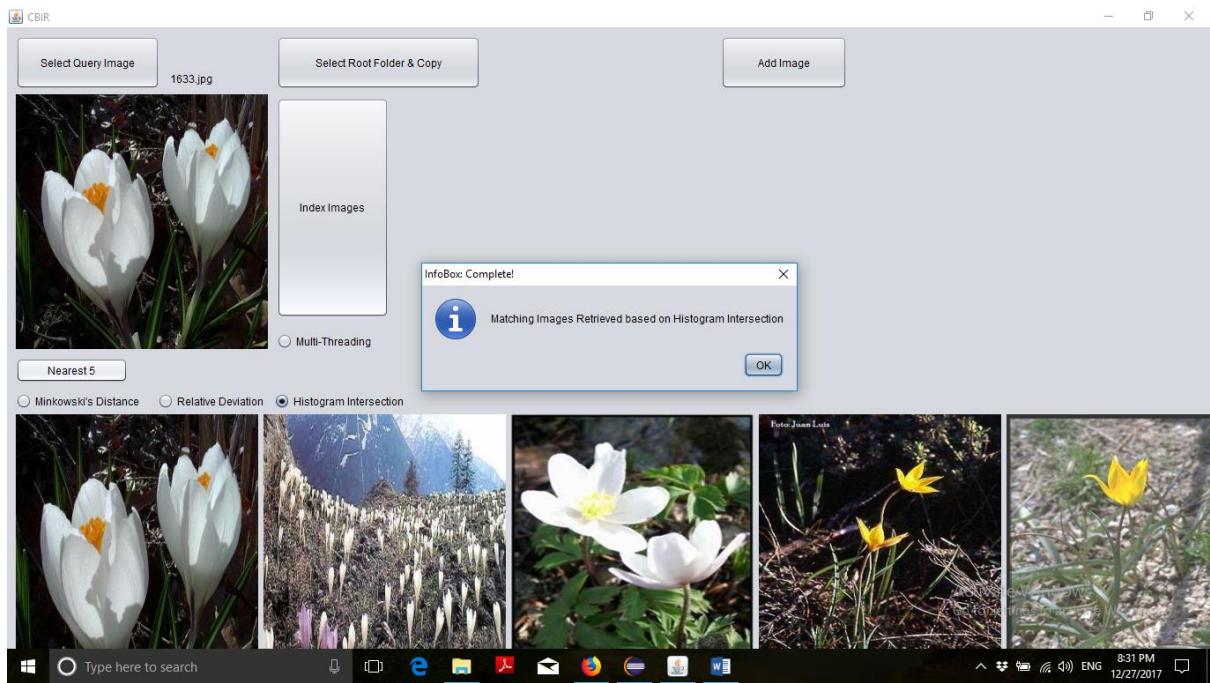




✓ ✗ ✗ ✗ ✗



✓ ✗ ✗ ✗ ✗



4.2. THỬ NGHIỆM 2

<http://cophir.isti.cnr.it/>

<http://www.sobigdata.eu/content/cophir>

<http://grepcode.com>

Học viên sử dụng source code của The Institute of Information Science and Technologies – ISTI. Sau đó chỉnh sửa lại cho phù hợp. Sau khi chỉnh sửa thì build success nhưng do không tìm được bộ datasets CoPhIR nên học viên phải tạo lập bộ dữ liệu theo tiêu chuẩn của CoPhIR nhưng do thời gian gấp rút nên học viên chưa làm kịp. Học viên nhận thấy rằng nhóm tác giả ISTI-CNR làm bài bản, đầy đủ nên học viên tìm hiểu, học hỏi thêm.

Bộ sưu tập thử nghiệm CoPhIR (Content-based Photo Image Retrieval) dựa trên nội dung đã được phát triển để làm các bài kiểm tra đáng kể về khả năng mở rộng của cơ sở hạ tầng dự án SAPIR (SAPIR: Tìm kiếm trong nội dung âm thanh bằng cách sử dụng IR ngang hàng) để tìm kiếm tương tự. CoPhIR là kết quả của một nỗ lực chung của Phòng thí nghiệm NMIS và Phòng thí nghiệm HPC của ISTI-CNR ở Pisa, Italy. Chúng tôi đang trích xuất siêu dữ liệu từ kho lưu trữ Flickr, sử dụng EGEE European GRID, thông qua dự án DILIGENT. Đối với mỗi ảnh, tính năng hình ảnh chuẩn MPEG-7 đã được trích xuất. Mỗi mục nhập của test-bed chứa:

- Liên kết tới mục nhập tương ứng vào trang web Flickr - Ảnh thu nhỏ hình ảnh - Cấu trúc XML với thông tin người dùng Flickr trong mục nhập Flickr tương ứng: Tiêu đề, vị trí, GPS, thẻ, nhận xét, vv - Một cấu trúc XML với 5 tiêu chuẩn trích xuất các tính năng hình ảnh MPEG-

7: Màu sắc có thể thay đổi, cấu trúc màu, bộ cục màu, Histogram cạnh, kết cấu đồng nhất Dữ liệu thu thập được bao gồm 106 triệu hình ảnh được xử lý.

Nhóm tác giả:

- (1).Paolo Bolettieri, ISTI-CNR
- (2).Andrea Esuli, ISTI-CNR
- (3).Fabrizio Falchi, ISTI-CNR
- (4).Claudio Lucchese, ISTI-CNR
- (5).Raffaele Perego, ISTI-CNR
- (6).Tommaso Piccioli, ISTI-CNR
- (7).Fausto Rabitti, ISTI-CNR

Ngày bắt đầu: 2009 – Ngày công bố: 2015.

Datasets: 106 triệu ảnh.

```
eclipse-workspace_6 - ch11khmt/src/test/java/ir/p/distance/L2Test.java - Eclipse
File Edit Refactor Source Navigate Search Project Run Window Help

Project Explorer
CBR
cbir_CH11_P
cbir_springer
cbir2
ch11khmt
src/main/java
src/main/resources
src/test/java
src/test/resources
test-resource
JRE System Library [JavaSE-1.8]
Maven Dependencies
Referenced Libraries
src
target
pom.xml

L2Test.java
1 package ir.p.distance;
2 import org.junit.Test;
3
4 public class L2Test {
5
6     @Test
7     public void test() {
8
9         int length = 4096;
10        float[] f1 = new float[length];
11        float[] f2 = new float[length];
12
13    }
14
15    @Test
16    public void test2() {
17
18        int length = 4096;
19        float[] f1 = new float[length];
20        float[] f2 = new float[length];
21
22    }
23
24    @Test
25    public void test3() {
26
27        int length = 4096;
28        float[] f1 = new float[length];
29        float[] f2 = new float[length];
30
31    }
32
33    @Test
34    public void test4() {
35
36        int length = 4096;
37        float[] f1 = new float[length];
38        float[] f2 = new float[length];
39
40    }
41
42    @Test
43    public void test5() {
44
45        int length = 4096;
46        float[] f1 = new float[length];
47        float[] f2 = new float[length];
48
49    }
50
51    @Test
52    public void test6() {
53
54        int length = 4096;
55        float[] f1 = new float[length];
56        float[] f2 = new float[length];
57
58    }
59
60    @Test
61    public void test7() {
62
63        int length = 4096;
64        float[] f1 = new float[length];
65        float[] f2 = new float[length];
66
67    }
68
69    @Test
70    public void test8() {
71
72        int length = 4096;
73        float[] f1 = new float[length];
74        float[] f2 = new float[length];
75
76    }
77
78    @Test
79    public void test9() {
80
81        int length = 4096;
82        float[] f1 = new float[length];
83        float[] f2 = new float[length];
84
85    }
86
87    @Test
88    public void test10() {
89
90        int length = 4096;
91        float[] f1 = new float[length];
92        float[] f2 = new float[length];
93
94    }
95
96    @Test
97    public void test11() {
98
99        int length = 4096;
100       float[] f1 = new float[length];
101      float[] f2 = new float[length];
102
103    }
104
105    @Test
106    public void test12() {
107
108        int length = 4096;
109        float[] f1 = new float[length];
110        float[] f2 = new float[length];
111
112    }
113
114    @Test
115    public void test13() {
116
117        int length = 4096;
118        float[] f1 = new float[length];
119        float[] f2 = new float[length];
120
121    }
122
123    @Test
124    public void test14() {
125
126        int length = 4096;
127        float[] f1 = new float[length];
128        float[] f2 = new float[length];
129
130    }
131
132    @Test
133    public void test15() {
134
135        int length = 4096;
136        float[] f1 = new float[length];
137        float[] f2 = new float[length];
138
139    }
140
141    @Test
142    public void test16() {
143
144        int length = 4096;
145        float[] f1 = new float[length];
146        float[] f2 = new float[length];
147
148    }
149
150    @Test
151    public void test17() {
152
153        int length = 4096;
154        float[] f1 = new float[length];
155        float[] f2 = new float[length];
156
157    }
158
159    @Test
160    public void test18() {
161
162        int length = 4096;
163        float[] f1 = new float[length];
164        float[] f2 = new float[length];
165
166    }
167
168    @Test
169    public void test19() {
170
171        int length = 4096;
172        float[] f1 = new float[length];
173        float[] f2 = new float[length];
174
175    }
176
177    @Test
178    public void test20() {
179
180        int length = 4096;
181        float[] f1 = new float[length];
182        float[] f2 = new float[length];
183
184    }
185
186    @Test
187    public void test21() {
188
189        int length = 4096;
190        float[] f1 = new float[length];
191        float[] f2 = new float[length];
192
193    }
194
195    @Test
196    public void test22() {
197
198        int length = 4096;
199        float[] f1 = new float[length];
200        float[] f2 = new float[length];
201
202    }
203
204    @Test
205    public void test23() {
206
207        int length = 4096;
208        float[] f1 = new float[length];
209        float[] f2 = new float[length];
210
211    }
212
213    @Test
214    public void test24() {
215
216        int length = 4096;
217        float[] f1 = new float[length];
218        float[] f2 = new float[length];
219
220    }
221
222    @Test
223    public void test25() {
224
225        int length = 4096;
226        float[] f1 = new float[length];
227        float[] f2 = new float[length];
228
229    }
230
231    @Test
232    public void test26() {
233
234        int length = 4096;
235        float[] f1 = new float[length];
236        float[] f2 = new float[length];
237
238    }
239
240    @Test
241    public void test27() {
242
243        int length = 4096;
244        float[] f1 = new float[length];
245        float[] f2 = new float[length];
246
247    }
248
249    @Test
250    public void test28() {
251
252        int length = 4096;
253        float[] f1 = new float[length];
254        float[] f2 = new float[length];
255
256    }
257
258    @Test
259    public void test29() {
260
261        int length = 4096;
262        float[] f1 = new float[length];
263        float[] f2 = new float[length];
264
265    }
266
267    @Test
268    public void test30() {
269
270        int length = 4096;
271        float[] f1 = new float[length];
272        float[] f2 = new float[length];
273
274    }
275
276    @Test
277    public void test31() {
278
279        int length = 4096;
280        float[] f1 = new float[length];
281        float[] f2 = new float[length];
282
283    }
284
285    @Test
286    public void test32() {
287
288        int length = 4096;
289        float[] f1 = new float[length];
290        float[] f2 = new float[length];
291
292    }
293
294    @Test
295    public void test33() {
296
297        int length = 4096;
298        float[] f1 = new float[length];
299        float[] f2 = new float[length];
300
301    }
302
303    @Test
304    public void test34() {
305
306        int length = 4096;
307        float[] f1 = new float[length];
308        float[] f2 = new float[length];
309
310    }
311
312    @Test
313    public void test35() {
314
315        int length = 4096;
316        float[] f1 = new float[length];
317        float[] f2 = new float[length];
318
319    }
320
321    @Test
322    public void test36() {
323
324        int length = 4096;
325        float[] f1 = new float[length];
326        float[] f2 = new float[length];
327
328    }
329
330    @Test
331    public void test37() {
332
333        int length = 4096;
334        float[] f1 = new float[length];
335        float[] f2 = new float[length];
336
337    }
338
339    @Test
340    public void test38() {
341
342        int length = 4096;
343        float[] f1 = new float[length];
344        float[] f2 = new float[length];
345
346    }
347
348    @Test
349    public void test39() {
350
351        int length = 4096;
352        float[] f1 = new float[length];
353        float[] f2 = new float[length];
354
355    }
356
357    @Test
358    public void test40() {
359
360        int length = 4096;
361        float[] f1 = new float[length];
362        float[] f2 = new float[length];
363
364    }
365
366    @Test
367    public void test41() {
368
369        int length = 4096;
370        float[] f1 = new float[length];
371        float[] f2 = new float[length];
372
373    }
374
375    @Test
376    public void test42() {
377
378        int length = 4096;
379        float[] f1 = new float[length];
380        float[] f2 = new float[length];
381
382    }
383
384    @Test
385    public void test43() {
386
387        int length = 4096;
388        float[] f1 = new float[length];
389        float[] f2 = new float[length];
390
391    }
392
393    @Test
394    public void test44() {
395
396        int length = 4096;
397        float[] f1 = new float[length];
398        float[] f2 = new float[length];
399
400    }
401
402    @Test
403    public void test45() {
404
405        int length = 4096;
406        float[] f1 = new float[length];
407        float[] f2 = new float[length];
408
409    }
410
411    @Test
412    public void test46() {
413
414        int length = 4096;
415        float[] f1 = new float[length];
416        float[] f2 = new float[length];
417
418    }
419
420    @Test
421    public void test47() {
422
423        int length = 4096;
424        float[] f1 = new float[length];
425        float[] f2 = new float[length];
426
427    }
428
429    @Test
430    public void test48() {
431
432        int length = 4096;
433        float[] f1 = new float[length];
434        float[] f2 = new float[length];
435
436    }
437
438    @Test
439    public void test49() {
440
441        int length = 4096;
442        float[] f1 = new float[length];
443        float[] f2 = new float[length];
444
445    }
446
447    @Test
448    public void test50() {
449
450        int length = 4096;
451        float[] f1 = new float[length];
452        float[] f2 = new float[length];
453
454    }
455
456    @Test
457    public void test51() {
458
459        int length = 4096;
460        float[] f1 = new float[length];
461        float[] f2 = new float[length];
462
463    }
464
465    @Test
466    public void test52() {
467
468        int length = 4096;
469        float[] f1 = new float[length];
470        float[] f2 = new float[length];
471
472    }
473
474    @Test
475    public void test53() {
476
477        int length = 4096;
478        float[] f1 = new float[length];
479        float[] f2 = new float[length];
480
481    }
482
483    @Test
484    public void test54() {
485
486        int length = 4096;
487        float[] f1 = new float[length];
488        float[] f2 = new float[length];
489
490    }
491
492    @Test
493    public void test55() {
494
495        int length = 4096;
496        float[] f1 = new float[length];
497        float[] f2 = new float[length];
498
499    }
499
500    @Test
501    public void test56() {
502
503        int length = 4096;
504        float[] f1 = new float[length];
505        float[] f2 = new float[length];
506
507    }
508
509    @Test
510    public void test57() {
511
512        int length = 4096;
513        float[] f1 = new float[length];
514        float[] f2 = new float[length];
515
516    }
517
518    @Test
519    public void test58() {
520
521        int length = 4096;
522        float[] f1 = new float[length];
523        float[] f2 = new float[length];
524
525    }
526
527    @Test
528    public void test59() {
529
530        int length = 4096;
531        float[] f1 = new float[length];
532        float[] f2 = new float[length];
533
534    }
535
536    @Test
537    public void test60() {
538
539        int length = 4096;
540        float[] f1 = new float[length];
541        float[] f2 = new float[length];
542
543    }
544
545    @Test
546    public void test61() {
547
548        int length = 4096;
549        float[] f1 = new float[length];
550        float[] f2 = new float[length];
551
552    }
553
554    @Test
555    public void test62() {
556
557        int length = 4096;
558        float[] f1 = new float[length];
559        float[] f2 = new float[length];
560
561    }
562
563    @Test
564    public void test63() {
565
566        int length = 4096;
567        float[] f1 = new float[length];
568        float[] f2 = new float[length];
569
570    }
571
572    @Test
573    public void test64() {
574
575        int length = 4096;
576        float[] f1 = new float[length];
577        float[] f2 = new float[length];
578
579    }
579
580    @Test
581    public void test65() {
582
583        int length = 4096;
584        float[] f1 = new float[length];
585        float[] f2 = new float[length];
586
587    }
588
589    @Test
590    public void test66() {
591
592        int length = 4096;
593        float[] f1 = new float[length];
594        float[] f2 = new float[length];
595
596    }
597
598    @Test
599    public void test67() {
599
600        int length = 4096;
601        float[] f1 = new float[length];
602        float[] f2 = new float[length];
603
604    }
604
605    @Test
606    public void test68() {
607
608        int length = 4096;
609        float[] f1 = new float[length];
610        float[] f2 = new float[length];
611
612    }
613
614    @Test
615    public void test69() {
616
617        int length = 4096;
618        float[] f1 = new float[length];
619        float[] f2 = new float[length];
620
621    }
622
623    @Test
624    public void test70() {
625
626        int length = 4096;
627        float[] f1 = new float[length];
628        float[] f2 = new float[length];
629
630    }
631
632    @Test
633    public void test71() {
634
635        int length = 4096;
636        float[] f1 = new float[length];
637        float[] f2 = new float[length];
638
639    }
639
640    @Test
641    public void test72() {
642
643        int length = 4096;
644        float[] f1 = new float[length];
645        float[] f2 = new float[length];
646
647    }
648
649    @Test
650    public void test73() {
651
652        int length = 4096;
653        float[] f1 = new float[length];
654        float[] f2 = new float[length];
655
656    }
656
657    @Test
658    public void test74() {
659
660        int length = 4096;
661        float[] f1 = new float[length];
662        float[] f2 = new float[length];
663
664    }
665
666    @Test
667    public void test75() {
668
669        int length = 4096;
670        float[] f1 = new float[length];
671        float[] f2 = new float[length];
672
673    }
673
674    @Test
675    public void test76() {
676
677        int length = 4096;
678        float[] f1 = new float[length];
679        float[] f2 = new float[length];
680
681    }
681
682    @Test
683    public void test77() {
684
685        int length = 4096;
686        float[] f1 = new float[length];
687        float[] f2 = new float[length];
688
689    }
689
690    @Test
691    public void test78() {
692
693        int length = 4096;
694        float[] f1 = new float[length];
695        float[] f2 = new float[length];
696
697    }
697
698    @Test
699    public void test79() {
699
700        int length = 4096;
701        float[] f1 = new float[length];
702        float[] f2 = new float[length];
703
704    }
704
705    @Test
706    public void test80() {
707
708        int length = 4096;
709        float[] f1 = new float[length];
710        float[] f2 = new float[length];
711
712    }
712
713    @Test
714    public void test81() {
715
716        int length = 4096;
717        float[] f1 = new float[length];
718        float[] f2 = new float[length];
719
720    }
720
721    @Test
722    public void test82() {
723
724        int length = 4096;
725        float[] f1 = new float[length];
726        float[] f2 = new float[length];
727
728    }
728
729    @Test
729    public void test83() {
730
731        int length = 4096;
732        float[] f1 = new float[length];
733        float[] f2 = new float[length];
734
735    }
735
736    @Test
737    public void test84() {
738
739        int length = 4096;
740        float[] f1 = new float[length];
741        float[] f2 = new float[length];
742
743    }
743
744    @Test
745    public void test85() {
746
747        int length = 4096;
748        float[] f1 = new float[length];
749        float[] f2 = new float[length];
750
751    }
751
752    @Test
753    public void test86() {
754
755        int length = 4096;
756        float[] f1 = new float[length];
757        float[] f2 = new float[length];
758
759    }
759
760    @Test
761    public void test87() {
762
763        int length = 4096;
764        float[] f1 = new float[length];
765        float[] f2 = new float[length];
766
767    }
767
768    @Test
769    public void test88() {
768
769        int length = 4096;
770        float[] f1 = new float[length];
771        float[] f2 = new float[length];
772
773    }
773
774    @Test
775    public void test89() {
774
775        int length = 4096;
776        float[] f1 = new float[length];
777        float[] f2 = new float[length];
778
779    }
779
780    @Test
781    public void test90() {
780
781        int length = 4096;
782        float[] f1 = new float[length];
783        float[] f2 = new float[length];
784
785    }
785
786    @Test
787    public void test91() {
786
787        int length = 4096;
788        float[] f1 = new float[length];
789        float[] f2 = new float[length];
790
791    }
791
792    @Test
793    public void test92() {
792
793        int length = 4096;
794        float[] f1 = new float[length];
795        float[] f2 = new float[length];
796
797    }
797
798    @Test
799    public void test93() {
798
799        int length = 4096;
800        float[] f1 = new float[length];
801        float[] f2 = new float[length];
802
803    }
803
804    @Test
805    public void test94() {
804
805        int length = 4096;
806        float[] f1 = new float[length];
807        float[] f2 = new float[length];
808
809    }
809
810    @Test
811    public void test95() {
810
811        int length = 4096;
812        float[] f1 = new float[length];
813        float[] f2 = new float[length];
814
815    }
815
816    @Test
817    public void test96() {
816
817        int length = 4096;
818        float[] f1 = new float[length];
819        float[] f2 = new float[length];
820
821    }
821
822    @Test
823    public void test97() {
822
823        int length = 4096;
824        float[] f1 = new float[length];
825        float[] f2 = new float[length];
826
827    }
827
828    @Test
829    public void test98() {
828
829        int length = 4096;
830        float[] f1 = new float[length];
831        float[] f2 = new float[length];
832
833    }
833
834    @Test
835    public void test99() {
834
835        int length = 4096;
836        float[] f1 = new float[length];
837        float[] f2 = new float[length];
838
839    }
839
840    @Test
841    public void test100() {
840
841        int length = 4096;
842        float[] f1 = new float[length];
843        float[] f2 = new float[length];
844
845    }
845
846    @Test
847    public void test101() {
846
847        int length = 4096;
848        float[] f1 = new float[length];
849        float[] f2 = new float[length];
850
851    }
851
852    @Test
853    public void test102() {
852
853        int length = 4096;
854        float[] f1 = new float[length];
855        float[] f2 = new float[length];
856
857    }
857
858    @Test
859    public void test103() {
858
859        int length = 4096;
860        float[] f1 = new float[length];
861        float[] f2 = new float[length];
862
863    }
863
864    @Test
865    public void test104() {
864
865        int length = 4096;
866        float[] f1 = new float[length];
867        float[] f2 = new float[length];
868
869    }
869
870    @Test
871    public void test105() {
870
871        int length = 4096;
872        float[] f1 = new float[length];
873        float[] f2 = new float[length];
874
875    }
875
876    @Test
877    public void test106() {
876
877        int length = 4096;
878        float[] f1 = new float[length];
879        float[] f2 = new float[length];
880
881    }
881
882    @Test
883    public void test107() {
882
883        int length = 4096;
884        float[] f1 = new float[length];
885        float[] f2 = new float[length];
886
887    }
887
888    @Test
889    public void test108() {
888
889        int length = 4096;
890        float[] f1 = new float[length];
891        float[] f2 = new float[length];
892
893    }
893
894    @Test
895    public void test109() {
894
895        int length = 4096;
896        float[] f1 = new float[length];
897        float[] f2 = new float[length];
898
899    }
899
900    @Test
901    public void test110() {
900
901        int length = 4096;
902        float[] f1 = new float[length];
903        float[] f2 = new float[length];
904
905    }
905
906    @Test
907    public void test111() {
906
907        int length = 4096;
908        float[] f1 = new float[length];
909        float[] f2 = new float[length];
910
911    }
911
912    @Test
913    public void test112() {
912
913        int length = 4096;
914        float[] f1 = new float[length];
915        float[] f2 = new float[length];
916
917    }
917
918    @Test
919    public void test113() {
918
919        int length = 4096;
920        float[] f1 = new float[length];
921        float[] f2 = new float[length];
922
923    }
923
924    @Test
925    public void test114() {
924
925        int length = 4096;
926        float[] f1 = new float[length];
927        float[] f2 = new float[length];
928
929    }
929
930    @Test
931    public void test115() {
930
931        int length = 4096;
932        float[] f1 = new float[length];
933        float[] f2 = new float[length];
934
935    }
935
936    @Test
937    public void test116() {
936
937        int length = 4096;
938        float[] f1 = new float[length];
939        float[] f2 = new float[length];
940
941    }
941
942    @Test
943    public void test117() {
942
943        int length = 4096;
944        float[] f1 = new float[length];
945        float[] f2 = new float[length];
946
947    }
947
948    @Test
949    public void test118() {
948
949        int length = 4096;
950        float[] f1 = new float[length];
951        float[] f2 = new float[length];
952
953    }
953
954    @Test
955    public void test119() {
954
955        int length = 4096;
956        float[] f1 = new float[length];
957        float[] f2 = new float[length];
958
959    }
959
960    @Test
961    public void test120() {
960
961        int length = 4096;
962        float[] f1 = new float[length];
963        float[] f2 = new float[length];
964
965    }
965
966    @Test
967    public void test121() {
966
967        int length = 4096;
968        float[] f1 = new float[length];
969        float[] f2 = new float[length];
970
971    }
971
972    @Test
973    public void test122() {
972
973        int length = 4096;
974        float[] f1 = new float[length];
975        float[] f2 = new float[length];
976
977    }
977
978    @Test
979    public void test123() {
978
979        int length = 4096;
980        float[] f1 = new float[length];
981        float[] f2 = new float[length];
982
983    }
983
984    @Test
985    public void test124() {
984
985        int length = 4096;
986        float[] f1 = new float[length];
987        float[] f2 = new float[length];
988
989    }
989
990    @Test
991    public void test125() {
990
991        int length = 4096;
992        float[] f1 = new float[length];
993        float[] f2 = new float[length];
994
995    }
995
996    @Test
997    public void test126() {
996
997        int length = 4096;
998        float[] f1 = new float[length];
999        float[] f2 = new float[length];
1000
1001    }
1001
1002    @Test
1003    public void test101() {
1002
1003        int length = 4096;
1004        float[] f1 = new float[length];
1005        float[] f2 = new float[length];
1006
1007    }
1007
1008    @Test
1009    public void test102() {
1008
1009        int length = 4096;
1010        float[] f1 = new float[length];
1011        float[] f2 = new float[length];
1012
1013    }
1013
1014    @Test
1015    public void test103() {
1014
1015        int length = 4096;
1016        float[] f1 = new float[length];
1017        float[] f2 = new float[length];
1018
1019    }
1019
1020    @Test
1021    public void test104() {
1020
1021        int length = 4096;
1022        float[] f1 = new float[length];
1023        float[] f2 = new float[length];
1024
1025    }
1025
1026    @Test
1027    public void test105() {
1026
1027        int length = 4096;
1028        float[] f1 = new float[length];
1029        float[] f2 = new float[length];
1030
1031    }
1031
1032    @Test
1033    public void test106() {
1032
1033        int length = 4096;
1034        float[] f1 = new float[length];
1035        float[] f2 = new float[length];
1036
1037    }
1037
1038    @Test
1039    public void test107() {
1038
1039        int length = 4096;
1040        float[] f1 = new float[length];
1041        float[] f2 = new float[length];
1042
1043    }
1043
1044    @Test
1045    public void test108() {
1044
1045        int length = 4096;
1046        float[] f1 = new float[length];
1047        float[] f2 = new float[length];
1048
1049    }
1049
1050    @Test
1051    public void test109() {
1050
1051        int length = 4096;
1052        float[] f1 = new float[length];
1053        float[] f2 = new float[length];
1054
1055    }
1055
1056    @Test
1057    public void test110() {
1056
1057        int length = 4096;
1058        float[] f1 = new float[length];
1059        float[] f2 = new float[length];
1060
1061    }
1061
1062    @Test
1063    public void test111() {
1062
1063        int length = 4096;
1064        float[] f1 = new float[length];
1065        float[] f2 = new float[length];
1066
1067    }
1067
1068    @Test
1069    public void test112() {
1068
1069        int length = 4096;
1070        float[] f1 = new float[length];
1071        float[] f2 = new float[length];
1072
1073    }
1073
1074    @Test
1075    public void test113() {
1074
1075        int length = 4096;
1076        float[] f1 = new float[length];
1077        float[] f2 = new float[length];
1078
1079    }
1079
1080    @Test
1081    public void test114() {
1080
1081        int length = 4096;
1082        float[] f1 = new float[length];
1083        float[] f2 = new float[length];
1084
1085    }
1085
1086    @Test
1087    public void test115() {
1086
1087        int length = 4096;
1088        float[] f1 = new float[length];
1089        float[] f2 = new float[length];
1090
1091    }
1091
1
```

TÀI LIỆU THAM KHẢO

1. <http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/index.html>
2. <https://github.com/nvtiep/Instance-Search>
3. <http://www.robots.ox.ac.uk/~vgg/data/>
4. <https://deeplearning4j.org>
5. <https://www.tensorflow.org/>
6. <https://github.com/tensorflow/tensorflow>
7. <https://github.com/ffalchi/it.cnr.isti.vir>
8. <http://grepcode.com>
9. <http://www.sobigdata.eu/private/ouhn3inebchcerhncurhebcnhn8749892/datasets>
10. Lý Quốc Ngọc (2008), Xây dựng hệ thống truy vấn thông tin thị giác dựa vào nội dung, Đại học Khoa học tự nhiên, Đại học Quốc gia TP.HCM, Luận án tiến sĩ Toán học.
11. Kushal Vyas, Yash Vora and Raj Vastani (2016), “Using Bag of Visual Words and Spatial Pyramid Matching for Object Classification along with Applications for RIS”, Elsevier, Procedia Computer Science 89 (2016) 457 – 464.
12. Relja Arandjelović and Andrew Zisserman (CVPR 2012), “Three things everyone should know to improve object retrieval”, University of Oxford.
13. Yanzhi Chen (2013), Efficient and Robust image ranking for Object Retrieval, PhD thesis, The University of Adelaide, Australia.
14. Rahayu A Hamid (2017), Image search: An investigation of factors affecting search behaviour of users, PhD Thesis, RMIT University, USA.
15. Boca Raton (2016), Context-based image concept detection and annotation, PhD Thesis, Florida Atlantic University, USA.
16. Yanzhi Chen (2013), Efficient and Robust image ranking for Object Retrieval, PhD Thesis, The University of Adelaide, Australia.
17. Jiangfan Feng, Yuanyuan Liu, and Lin Wu (2017), Bag of Visual Words Model with Deep Spatial Features for Geographical scene classification, Computational Intelligence and Neuroscience, Volume 2017.
18. G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray (2004), “Visual categorization with bags of keypoints,” Workshop on Statistical Learning in Computer Vision Eccv, pp. 1–22, 2004.
19. S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: spatial pyramid matching for recognizing natural scene categories”, in Proceedings of the IEEE

- Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06), pp.2169–2178, June 2006.
- 20. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, and Z. Chen (2015), “TensorFlow: Large-scale machine learning on heterogeneous systems”,
 - 21. Nadhir Ben Halima, Osama Hosam (2016), Bag of Words Based Surveillance System Using SVM, Vol. 10, No. 4 (2016), pp.331-346.
 - 22. Nanayakkara Wasam Uluwitige Dinesha Chathurani (2017), Content-Based Image Retrieval with Image Signatures, PhD Thesis, Queensland University of Technology.
 - 23. Xiaojiang Peng, Cordelia Schmid (2016), Multi-region two-stream R-CNN for action detection. ECCV- European Conference on Computer Vision, Oct 2016, Amsterdam, Netherlands. Springer, 9908, pp.744-759, 2016, Lecture Notes in Computer Science.
 - 24. Aggarwal, J.K., Ryoo, M.S.: Human activity analysis: A review. ACM Computing Surveys (CSUR) 43(3) (2011) 16.
 - 25. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521(7553) (2015) 436–444.
 - 26. Uijlings, J.R., van de Sande, K.E., Gevers, T., Smeulders, A.W (2013) “Selective search for object recognition”, IJCV 104(2) (2013) 154–171.
 - 27. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015), Spatial pyramid pooling in deep convolutional networks for visual recognition. PAMI 37(9) (2015) 1904–1916.
 - 28. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun (2016), Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, IEEE Transactions on Pattern Analysis and Machine Intelligence.
 - 29. Yulong Wu (2017), Active control of camera parameters and algorithm selection for object detection, Master Thesis, York University Toronto, Ontario, Canada.