ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO MÔN HỌC: NHẬN DẠNG THỊ GIÁC VÀ ỨNG DỤNG

CÀI ĐẶT HỆ THỐNG TÌM KIẾM ẢNH

Giảng viên:

TS. LÊ ĐÌNH DUY

TS. NGUYỄN TẤN TRẦN MINH KHANG

Học viên thực hiện:

TRẦN TRUNG ĐỨC – CH1601003

TP Hồ Chí Minh, tháng 12 năm 2017

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành đến **TS. Lê Đình Duy** và **TS. Nguyễn Tấn Trần Minh Khang** vì những kiến thức quý báu mà các thầy truyền đạt cho chúng em trong suốt quá trình học tập môn **Nhận dạng thị giác và ứng dụng**, đó cũng chính là những nền tảng, tư liệu giúp em có thể hoàn thành đồ án này.

Em xin chân thành cảm ơn!

Học viên

Trần Trung Đức

MỤC LỤC

1. Thông tin đồ án

Link github:

https://github.com/ductrandev/uit-vra/tree/master/VRA.Final.TranTrungDuc

Link video demo:

https://youtu.be/j2RFNhnPbx0

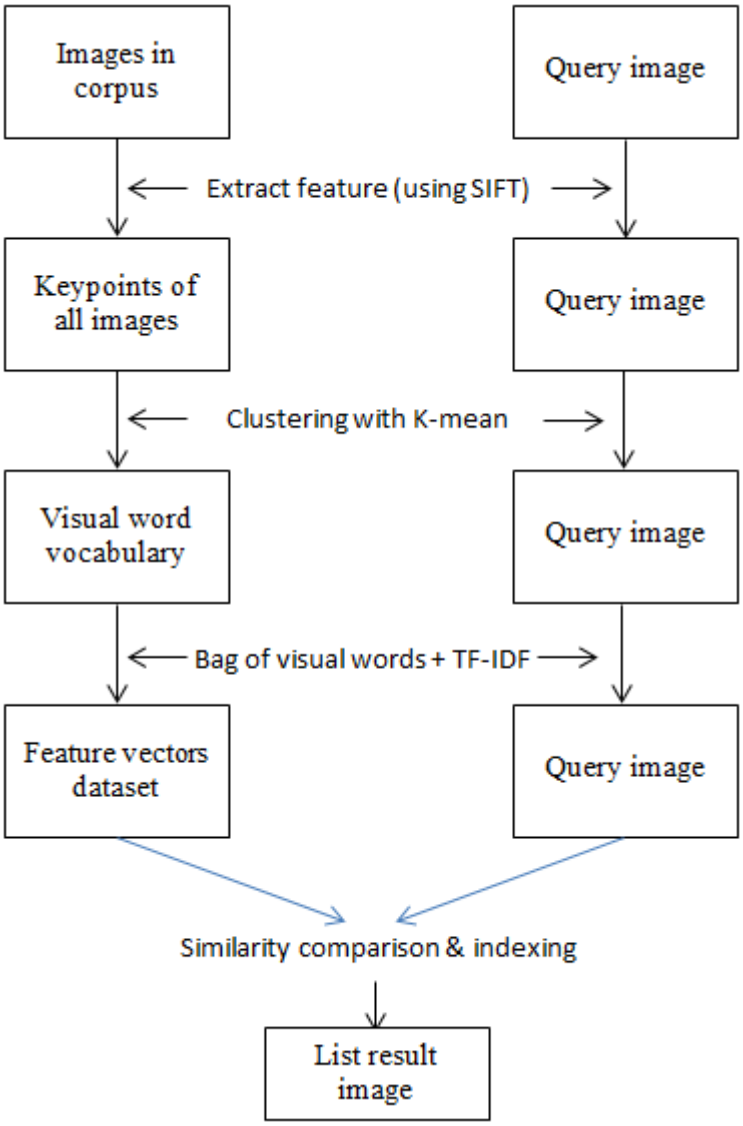
2. Mục tiêu và phạm vi của đồ án

Xây dựng hệ thống tìm kiếm ảnh dựa trên nội dung bức ảnh (content based) với phạm vi và yêu cầu như sau:

- **Tài nguyên (resource)**: cơ sở dữ liệu hình ảnh (đồ án sử dụng dataset Oxford Building 5K).
- **Input**: ảnh truy vấn (đồ án sử dụng một số ảnh chọn từ Groundtruth Queries của dataset Oxford Building).
- **Output**: danh sách các ảnh trong dataset gần giống nhất với ảnh truy vấn (output trong cài đặt của đồ án là danh sách chứa 10 ảnh kết quả).

3. Phương pháp thực hiện

Hệ thống tìm kiếm ảnh được cài đặt áp dụng Vector Space Model với mô hình tổng quát sau:



3.1 Xây dựng vector đặc trưng cho ảnh trong corpus và ảnh truy vấn

Xây dựng visual word vocabulary:

- Trong computer vision, chúng ta thường định nghĩa kích cỡ các cửa sổ(window) rồi quét ảnh với các cửa sổ đó để tìm những đặc tính(feature) của ảnh. Sử dụng **SIFT** tính DoG (Difference of Gaussians) trên từng

pixel bằng cách lấy diff của Gaussian Blur với 2 \sigma khác nhau. Sau khi tính được DoG của toàn ảnh, xét trên từng pixel so sánh với 8 neighbors và 9 pixels tương ứng của scale ảnh ngay trên và 9 pixels tương ứng ở scale dưới, nếu pixel đó là local extrema (lớn nhất) thì nó sẽ được coi như là 1 keypoint ở scale đó.

- SIFT sẽ tính Keypoint descriptor bằng cách lấy 16x16 neighborhood(điểm liền kề) của keypoint đó, rồi chia thành 16 sub-blocks với kích thước 4x4. Với mỗi sub-block, ta sẽ tạo được 8 bin oritation (như hình dưới). Do đó tất cả sẽ có 128 bin giá trị tương ứng với 1 vector biểu hiện cho keypoint descriptor.
- Từ tập tất cả các keypoint descriptors của tất cả các ảnh (là các vector trong không gian 128 chiều), ta tiến hành gom cụm để thu được k cụm (với k là số lượng từ trong visual word vocabulary – do người cài đặt tự quy định). Tâm của mỗi cụm trong k cụm sẽ đại diện cho một visual word.

Biến đổi ảnh về dạng vector trong Vector space model:

- Thể hiện một ảnh dưới dạng tập hợp các visual word: sử dụng khái niệm bag of visual word, là một histogram thể hiện cho số lần xuất hiện của các visual word trong ảnh.

  Ví dụ giả sử xây dựng visual vocabulary với số từ vựng k = 5. Sau khi thực hiện đếm số lần xuất hiện của các visual word trong ảnh I cho một ảnh X, ta thu được vector như sau:

vw1	vw2	vw3	vw4	vw5
7	5	9	3	0

  Vector V(7, 5, 9, 3, 0) sẽ đại diện cho hình ảnh I.
- Áp dụng phương pháp đánh trọng số TF-IDF weighting để cập nhật giá trị cho các feature vector đại diện cho các ảnh:
  - Sau bước trên mỗi ảnh sẽ là một vector $\mathbf{V\ (t_1, t_2, t_{3, \dots}, t_k)}$ với số chiều k = số lượng từ vựng.

- Giá trị t_i được cập nhật lại theo công thức TF-IDF weighting như sau:

  $$\mathbf{t_i} = \frac{n_{id}}{n_d} \times \left(1 + \log\left(\frac{N}{n_i}\right)\right)$$

  trong đó t_i là giá trị tại chiều thứ i (tại vị trí visual word thứ i trong tập visual vocabulary) của feature vector đại diện cho ảnh.

  - ✓ n_{id}: *số lần xuất hiện của visual word thứ i trong ảnh*
  - ✓ n_d: *tổng số visual words trong ảnh*
  - ✓ *N: là tổng số ảnh trong CSDL*
  - ✓ n_i: *số ảnh trong CSDL có chứa visual word thứ i*

Kết quả của toàn bộ quá trình này là một vector đặc trưng đại diện cho ảnh truy vấn và một CSDL chứa thông tin về toàn bộ các vector đặc trưng của tất cả các ảnh trong CSDL.

3.2 So sánh độ tương đồng và trả về kết quả

Với mô hình **Vector Space Model**, việc tìm kiếm được ảnh tương tự với ảnh truy vấn được hiện thực bằng các tìm các vector đại diện cho các ảnh trong CSDL sao cho các vector này tương đồng với vector đặc trưng của ảnh truy vấn. Độ đo tương đồng giữa 2 vector được sử dụng ở đây là là độ Cosin:

$$\cos(q, v) = q.v / |q|*|v|$$

trong đó q là vector đặc trưng của ảnh truy vấn; v là vector đặc trưng cho 1 ảnh trong CSDL. Khi giá trị cos (q, v) càng gần giá trị 1 thì độ tương đồng càng cao, nghĩa là ảnh có vector đặc trưng v càng giống với ảnh truy vấn.

Lần lượt tính độ tương đồng cosin của vector q với tất cả các vector đặc trưng của các ảnh trong CSDL, sau đó sắp xếp các kết quả giảm dần theo độ tương đồng cosin → ta thu được danh sách các ảnh giống với ảnh truy vấn (theo mức độ giống nhau giảm dần). Cuối cùng, hiển thị danh sách ảnh kết quả cho người dùng.

4. Cài đặt và thử nghiệm

4.1 Các công cụ và thư viện

Matlab

Lý do lựa chọn:

- Dễ triển khai một ứng dụng đơn giản.
- Nhiều công cụ xử lí ảnh và máy học được tích hợp sẵn trong các toolbox

Thư viện VLFeat

Lý do lựa chọn:

- Chứa các hàm cài đạt sẵn các thuật toán rút trích đặc trưng như SIFT, HOGF, .. và các thuật toán machine learning như k-mean, svm, …
- Sử dụng được trên nhiều nền tảng (Windows, Mac OS, Linux).
- Miến phí.

4.2 Các thành phần cài đặt chính

4.2.1 Xây dựng Retrieval Engine

Trong Matlab, ta xây dựng lớp **ImageRetrievalEngine** *(file matlab: ImageRetrievalEngine.m)* bao gồm tất cả các phương thức cần thiết cho việc truy vấn và lấy về danh sách các ảnh kết quả.

Các thuộc tính:

- **data_path**: đường dẫn tới Oxford Building Dataset.
- **vl_feat_toolbox**: đường dẫn tới file vl_setup.m của VLFeat library.
- **vocab_size**: số lượng từ trong bộ từ vựng (tương ứng với giá trị k cụm khi tiến hành gom cụm các SIFT descriptors của các ảnh trong CSDL)
- **db_size**: số lượng ảnh trong CSDL.

Các phương thức chính:

- function build_visual_vocabulary(obj, number_of_random_images)

- function build_db_images_vectors(obj)
- function descending_ranked_list = get_ranked_result_for_query (obj, path_of_query_image)

Chi tiết cài đặt các hàm sẽ được mô tả ở mục kế tiếp. Để truy vấn rất đơn giản ta chỉ cần tạo ra một đối tượng của lớp **ImageRetrievalEngine** (với các tham số để khởi tạo các thuộc tính) sau đó gọi các phương thức cần thiết. Ví dụ:

```
% create an instance of IrEngine to perform query task.
IrEngine = ImageRetrievalEngine(vl_feat, data_path, vocab_size,db_size );

%build visual vocabulary and vectors for all the images in dataset
IrEngine.build_visual_vocabulary(number_of_random_images);
IrEngine.build_db_images_vectors();

% get vector of query image then search and get results by comparing query
% vector and vectors of all the image in dataset
descending_ranked_list = IrEngine.get_ranked_result_for_query(path_of_query_image);
```

4.2.2 Chi tiết cài đặt của các hàm trong lớp ImageRetrievalEngine

❖ function build_visual_vocabulary: Hàm này giúp tạo ra bộ visual vocabulary.

Do bộ dataset lớn nên việc dùng toàn bộ ảnh để tạo ra bộ visual vocabulary rất tốn thời gian, do đó ta chỉ sử dụng một tập con các ảnh ngẫu nhiên trên tổng số ảnh của dataset để thực hiện công việc này *(số lượng ảnh ngẫu nhiên trong dataset dùng để xây dựng bộ visual vocabulary trong cài đặt thực tế của đề tài là 750)*

```
[~, all_SIFT_features] = vl_dsift(single(rgb2gray(images{1})),'fast', 'step', 50);

for i = 2:(size(images,1))
    [~, SIFT_features] = vl_dsift(single(rgb2gray(images{i})),'fast', 'step', 50);
    all_SIFT_features = cat(2, all_SIFT_features, SIFT_features);
end

[vocab, ~] = vl_kmeans(single(all_SIFT_features), vocab_size);
```

Với mỗi ảnh đầu vào, tính được SIFT descriptors với hàm vl_dsift của thư viện VLFeat. Tiến hành gom tất cả những SIFT descriptors thành k cụm (*k* =

vocab_size = số lượng từ trọng visual vocabulary; trong code cài đặt tác giả sử dụng giá trị vocab_size = 700) thu được sử dụng hàm vl_kmeans của thư viện VLFeat.

Biến **vocab** thu được là một matrix chứa thông tin về các vector, mỗi vector là tâm của 1 cụm (ở bước gom cụm phía trên) đại diện cho 1 visual word. Ta tiến hành lưu biến **vocab** này lại thành tập tin với hàm **save** của matlab để không phải xây dựng visual vocabulary sau này.

❖ function build_db_images_vectors:

Hàm này giúp tạo ra CSDL mới gồm tập hợp các vector đặc trưng cho các ảnh trong CSDL hình ảnh ban đầu. Hàm nhận tập hợp tất cả các đường dẫn của các ảnh trong CSDL. Với mỗi ảnh trong CSDL sử dụng hàm vl_shift của thư viện VLFeat để tính ra các SIFT descriptors. Với mỗi SIFT descriptor thu được, tiến hành xác định xem nó thuộc về visual vocabulary nào (sử dụng hàm knnsearch - K nearest neighbors của Matlab) ➔ thu được bag of visual words → tiến hành cập nhật trọng số cho vector theo TF-IDF weighting.

```
parfor i=1:(size(images,1))
    [~, feats] = vl_dsift(single(rgb2gray(images{i})), 'fast', 'step', 5);
    D_matrix = zeros([1 vocab_size]);

    all_nearest = knnsearch(single(vocab_inv), single(feats'));
    for n=1:size(all_nearest, 1)
        nearest_vocab=all_nearest(n);
        D_matrix(nearest_vocab) = D_matrix(nearest_vocab)+1;
    end
    image_feats(i, :) = D_matrix;
end
```

```
for i=1:size(image_feats, 1)
    for j = 1: size(image_feats, 2)
    %update tf-idf weighting
    total_words = number_of_words_in_each_doc(i,1);
    word_frequency = vocab_frequencies_in_DB(1,j);
    image_feats(i, j) = image_feats(i, j)/total_words * (1 + log(db_size/word_frequency));
    end
end
```

Biến imgvectors thu được là một matrix chứa thông tin về các vector đặc trưng đại diện cho các ảnh trong CSDL. Ta tiến hành lưu biến imgvectors này lại thành tập tin với hàm save của matlab để không phải tính lại sau này.

❖ function descending_ranked_list = get_ranked_result_for_query

```
query_vector = get_query_vector(path_of_query_image, obj.db_size);

%get ranked list result
ranked_list =  get_ranked_result(query_vector, obj.db_size);

[~,I]=sort(ranked_list(2,:), 'descend');
descending_ranked_list = ranked_list(:,I);
```

Hàm này nhận vào tham số là đường dẫn của ảnh truy vấn, đầu tiên vector đặc trưng cho ảnh này được tính. Các thức tính hoàn toàn tương tự như việc tính vector đặc trưng của các ảnh trong CSDL như trong hàm **build_db_images_vectors.** Sau đó, tiến hình tính độ tương đồng cosin của query vector và các vector đại diện của các ảnh trong CSDL cho về danh sách các ảnh phù hợp sau đó sắp xếp danh sách này theo độ tương đồng giảm dần.

```
for i = 1:nRow
    u = query_vector;
    v = imgvectors(i,:);
    ranked_list(1, i) = i;

    % calculate COSIN similarity
    ranked_list(2, i) = dot(u,v)/(norm(u,2)*norm(v,2));
end
```

Tính vô hướng của 2 vector và độ dài của vector lần lượt tính bằng hàm dot() và hàm norm được cung cấp sẵn bởi Matlab.

4.2.3 Hiển thị ảnh kết quả

Hàm function *displayResult(dataset_image_paths, descending_ranked_list)* được cài đặt để hiển thị kết quả trả về. Hàm này nhận vào một matrix descending_ranked_list có kích thước 2 x L (với L là số ảnh mong muốn hiển thị). Vì lý do kỹ thuật ở đây, chương trình cài đặt chỉ hiển thị 10 ảnh giống nhất với ảnh truy vấn, ứng với L = 10. Với mỗi cột thứ i, descending_ranked_list (1, i) là giá trị thể hiện vị trí của ảnh kết quả trong CSDL (ảnh thứ mấy trong CSDL) và descending_ranked_list (2, i) thể hiện độ tương đồng của ảnh đó với ảnh truy vấn.

Ta sử dụng hàm imgshow và suplot của matlab để hiển thị các ảnh giống nhất với ảnh truy vấn:

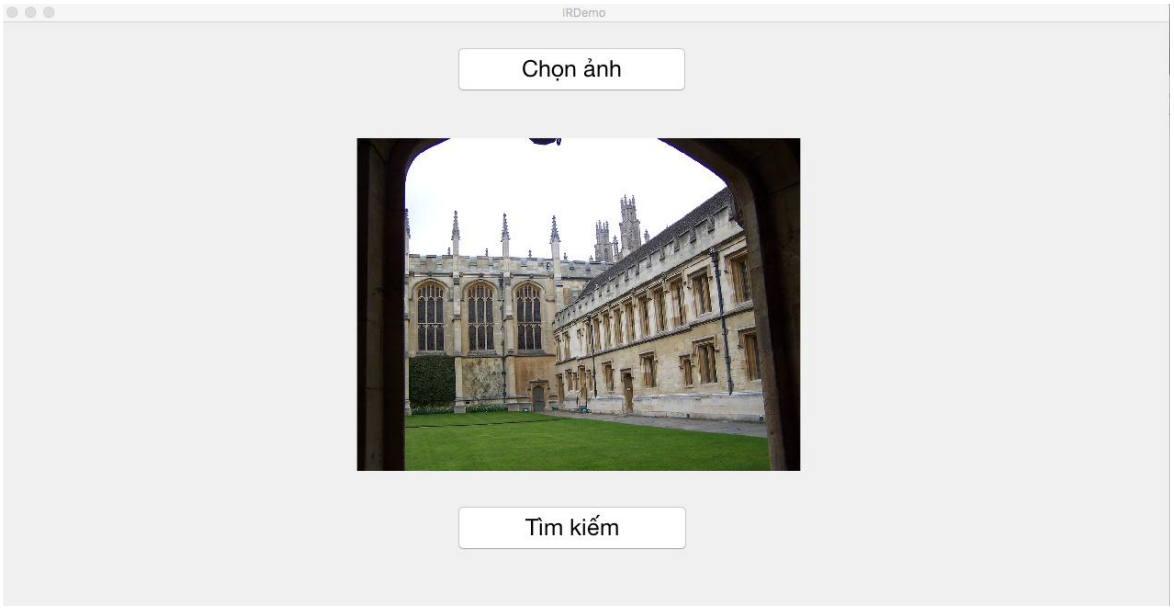
```
%show them in subplots
figure('Name', 'KET QUA TIM KIEM');
for i=1:number_of_result
    subplot(number_rows,3,i);
    h = imshow(imgs{i}, 'InitialMag',100, 'Border','tight');

    similarityValue = ['Similarity: ',num2str(descending_ranked_list(2,i))];
    title(similarityValue);

    imagePath = dataset_image_paths{descending_ranked_list(1,i)};
    set(h, 'ButtonDownFcn',{@callback,i,imagePath,similarityValue})
end
```

4.3 Thử nghiệm và đánh giá

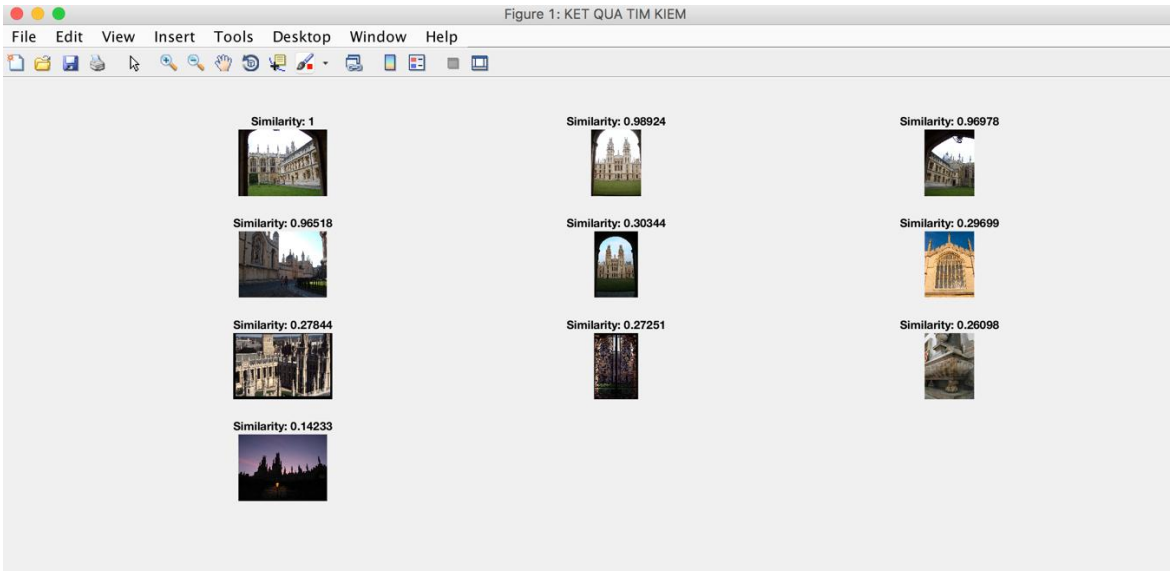
4.3.1 Hướng dẫn chạy thử nghiệm

- Cài đặt **Matlab** 2016 hoặc các phiên bản mới hơn.
- Tải và cài đặt **VLFeat** cho Matlab theo hướng dẫn tại: http://www.vlfeat.org/install-matlab.html
- Tải bộ dữ liệu **Oxford Buiding 5k** tại: http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/, sau đó giải nén và copy hết ảnh dữ liệu vào thư mục **dataset** trong thư mục chứa source code.
- Trong file **IRDemo.m** của source code, tại function **pushbutton2_Callback**, thiết lập lại các giá trị cấu hình sau cho phù hợp:
    - **vl_feat**: đường dẫn đến thư mục thư viện VLFead .
    - **data_path**: đường dẫn đến thư mục dataset.
- Mở Matlab, trỏ đến thư mục chứa source code, chạy lệnh IRDemo → giao diện input ảnh truy vấn sẽ xuất hiện.
- Tiến hành chọn ảnh truy vấn và click tìm kiếm để hiển thị ảnh kết quả.

Dưới đây là một số hình giao diện khi chạy thử nghiệm chương trình:



Giao diện danh sách 10 ảnh kết quả:



4.3.2 Đánh giá

4.3.2.1 Phương pháp

- Bước 1: Tính *precision* khi chạy thử nghiệm đối với mỗi ảnh truy vấn theo công thức:

P(i) = Số kết quả chính xác / Tổng số kết quả trả về

- Bước 2: Độ chính xác của hệ thống được tính bằng cách lấy trung bình cộng của các P(i).

4.3.2.2 Kết quả

Chạy thử nghiệm lần lượt với 10 ảnh truy vấn:

Ảnh truy vấn	**Số kết quả chính xác**	**Độ chính xác (%)**
all_souls_000013	2	20
all_souls_000026	3	30
ashmolean_000000	4	40
ashmolean_000007	2	20
balliol_000051	2	20
balliol_000187	2	20
bodleian_000108	7	70
bodleian_000163	9	90
christ_church_000179	9	90
christ_church_001020	3	30
Trung bình		**43**

→ Độ chính xác trung bình của hệ thống là **43%**

5. Kết luận

Kết quả đạt được:

- Hiểu rõ cơ chế hoạt động của hệ thống Content Based Image Retrieval (CBIR); kỹ thuật TF-IDF weighting và mô hình Vector Space Model (VSM) trong truy vấn thông tin.
- Cài đặt được một ứng dụng truy vấn hình ảnh đơn giản (áp dụng CBIR, VSM và TF-IDF weighting) trả về danh sách ảnh kết quả với độ tương đồng so với ảnh truy vấn; cài đặt ứng dụng sử dụng Matlab và thư viện VLFeat.

Hạn chế và hướng phát triển:

- Ứng dụng demo muốn chạy phải cài đặt Matlab, VLFeat; chưa phải là một sản phẩm có thể chạy riêng biệt ➔ cần nghiên cứu việc export code Matlab ra file .dll hoặc .jar để xây dựng ứng dụng mang tính thực tế.
- Chưa tối ưu hoá việc truy vấn dựa vào kỹ thuật Inverted Index Files ➔ cần áp dụng kỹ thuật này vào để tăng tốc quá trình truy vấn và lấy về kết quả.
- Phương pháp đánh giá còn đơn giản và mang tính chủ quan, chưa thực hiện so sánh kết quả của hệ thống đã cài đặt với các hệ thống truy vấn ảnh khác ➔ cần phải một phương pháp đánh giá chuẩn và có thống kê, so sánh với các hệ thống khác.

TÀI LIỆU THAM KHẢO

[1]. Ts. Lê Đình Duy, Ts. Nguyễn Tấn Trần Minh Khang, slide môn học *Nhận dạng thị giác và ứng dụng*, 2017.

[2]. Matlab, https://www.mathworks.com/products/matlab.html.

[3]. Thư viện vlfeat, http://www.vlfeat.org.

[4]. Bộ dữ liệu Oxford Building (5K), http://www.robots.ox.ac.uk/~vgg/data/oxbuildings.

[5]. Instance search project, https://github.com/nvtiep/Instance-Search.

[6]. CBIR project, https://sites.google.com/site/cbirpro/home.