

ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



Báo cáo đồ án

Bài tập: Individual Lab

Môn học: Truy vấn thông tin thị giác

Người thực hiện:

22127390 - Nguyễn Văn Lê Bá Thành

Giảng viên:

Thầy Võ Hoài Việt
Thầy Phạm Minh Hoàng

Ngày 26 tháng 9 năm 2025

Mục lục

1	Giới thiệu	2
2	Bảng đánh giá công việc	3
3	Môi trường lập trình	4
3.1	Môi trường lập trình	4
4	Kiến trúc hệ thống	5
4.1	Các phương pháp rút trích đặc trưng	6
4.1.1	Color Histogram	6
4.1.2	Color Correlogram	9
4.1.3	Histogram of Oriented Gradients	13
4.1.4	SIFT	18
4.1.5	ORB	20
4.2	Bag of Visual Word	22
5	Giao diện hệ thống	24
5.1	Giao diện trích xuất đặc trưng	24
5.2	Giao diện truy vấn ảnh	26
6	Thí nghiệm	28
6.1	Thiết lập thí nghiệm	28
6.1.1	Thiết lập phần cứng	28
6.1.2	Bộ dữ liệu	28
6.1.3	Các độ đo	30
6.2	Thí nghiệm 1: Thí nghiệm chọn vocabulary size tối ưu cho hai bộ dữ liệu	33
6.2.1	Mục đích	33
6.2.2	Nội dung	33
6.2.3	Kết quả	33
6.2.4	Kết luận	36
6.3	Thí nghiệm 2: Kiểm tra tốc độ xây dựng bộ dữ liệu đặc trưng	37
6.3.1	Mục đích	37
6.3.2	Nội dung	37
6.3.3	Kết quả	37
6.3.4	Kết luận	39
6.4	Thí nghiệm 3: Kiểm tra tốc độ và độ chính xác khi truy vấn	39
6.4.1	Mục đích	39
6.4.2	Nội dung	39
6.4.3	Kết quả	40
6.4.4	Kết luận	40
7	Kết luận	41

1 Giới thiệu

Content-Based Image Retrieval (CBIR), hay truy vấn ảnh dựa trên nội dung, là một lĩnh vực nhằm tìm kiếm và truy xuất các hình ảnh từ cơ sở dữ liệu dựa trên nội dung của chúng, chẳng hạn như màu sắc, hình dạng hay kết cấu. Khác với các phương pháp dựa trên các meta như từ khóa hoặc mô tả văn bản, CBIR tập trung vào việc phân tích trực tiếp nội dung hình ảnh thông qua các đặc trưng trích xuất tự động. Hệ thống CBIR thường bao gồm các bước chính như trích xuất đặc trưng, biểu diễn đặc trưng, đo độ tương đồng và xếp hạng kết quả. Nhờ khả năng phản ánh đúng hơn sự tương đồng thị giác giữa các hình ảnh, CBIR đã và đang được ứng dụng rộng rãi trong nhiều lĩnh vực như y học, an ninh, thương mại điện tử và quản lý dữ liệu hình ảnh đa phương tiện.

Trong bài tập này, sinh viên được yêu cầu lập trình một hệ thống truy vấn ảnh dựa trên nội dung hoàn chỉnh sử dụng C++ và thư viện OpenCV, với mục đích giúp cho sinh viên có kinh nghiệm về xây dựng hệ thống và áp dụng các kiến thức đã học được từ các môn học như xử lý ảnh, thị giác máy tính vào một phần mềm thực tế.

Bản báo cáo này phần cuối trong các báo cáo tiến độ, các báo cáo này sẽ giúp giảng viên cập nhật được tiến độ thực hiện của sinh viên và đưa ra các hướng dẫn cụ thể trên lớp nhằm cải thiện kiến thức và bài làm của sinh viên. Báo cáo này sẽ bao gồm các nội dung sau: Phần 2 sẽ mô tả về các tác vụ, các chức năng và các yêu cầu đã thực hiện được. Phần 3 sẽ mô tả về mỗi trường code, Phần 4 sẽ mô tả về kiến trúc của hệ thống. Phần 5 sẽ mô tả giao diện của hệ thống và cách sử dụng. Phần 6 sẽ mô tả các thí nghiệm và kết quả của hệ thống truy vấn. Phần 7 sẽ kết luận lại bài tập này.

2 Bảng đánh giá công việc

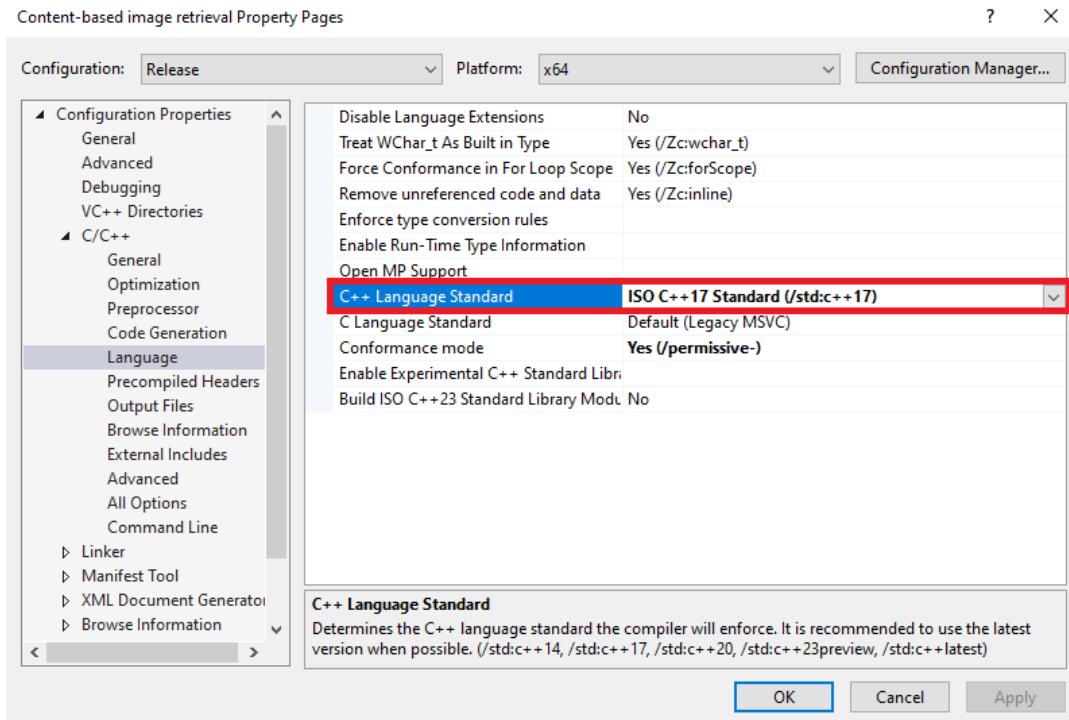
STT	Tác vụ	Tiến độ
1	Tổ chức thư mục source	100%
2	Tổ chức thư mục release	100%
3	Tổ chức thư mục docs	100%
4	Xây dựng hệ thống rút trích đặc trưng và đánh chỉ mục	100%
5	Xây dựng hệ thống truy vấn ảnh	100%
6	Cải thiện độ chính xác của hệ thống truy vấn	100%
7	Cải thiện tốc độ của hệ thống truy vấn	100%

Bảng 1: Bảng đánh giá

3 Môi trường lập trình

3.1 Môi trường lập trình

Bài tập này được thực hiện trên IDE Visual Studio Community 2022 với các Workloads được sử dụng bao gồm: Desktop development with C++ và các individual component được đề xuất. Tiêu chuẩn C++ được sử dụng hiện tại là ISO C++ 17 do có hỗ trợ thư viện giúp tạo thư mục.

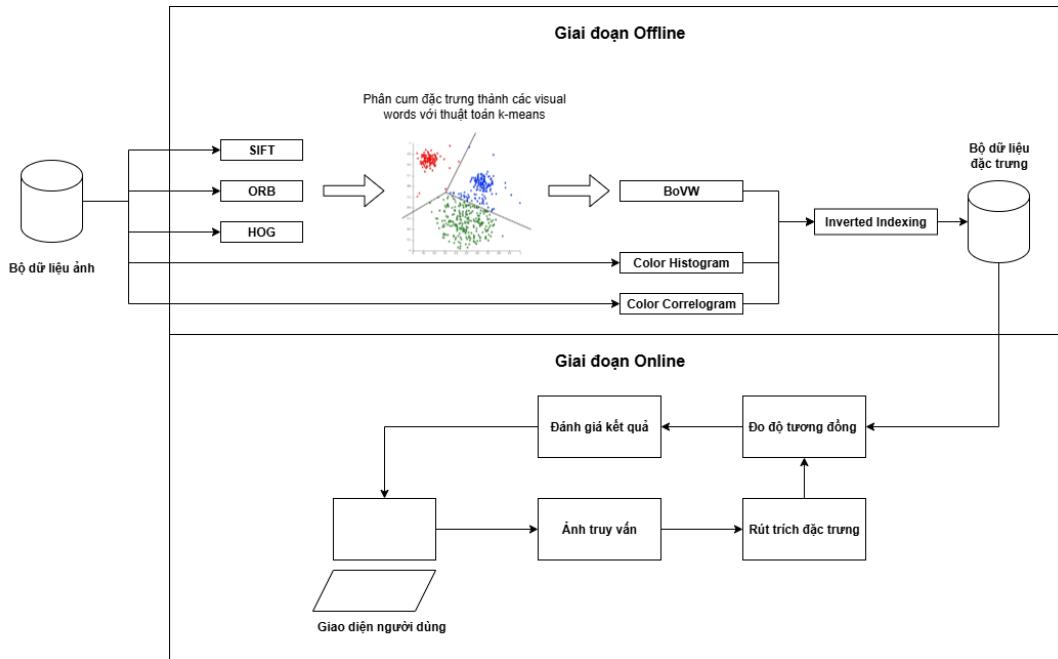


Hình 1: Tiêu chuẩn c++ được sử dụng

Phiên bản OpenCV sử dụng là OpenCV 4.10.0, do yêu cầu của đề bài các API được thư viện này hỗ trợ cũng có thể được sử dụng trong bài tập này, các API này có thể được xem ở [1].

4 Kiến trúc hệ thống

Trong phần này, kiến trúc của hệ thống sẽ và các thuật toán được sử dụng được giới thiệu. Hệ thống này áp dụng các kỹ thuật tạo chỉ mục như inverted indexing và gom nhóm k-mean để giúp tối ưu hóa tốt độ truy vấn. Thiết kế này được lấy ý tưởng từ [2] và áp dụng thêm các kiến thức được giảng dạy trên lớp.



Hình 2: Kiến trúc của hệ thống

Trong kiến trúc này được chia thành hai giai đoạn chính bao gồm:

1. Xây dựng bộ dữ liệu đặc trưng (Giai đoạn Offline)

Trong giai đoạn này, các ảnh từ bộ dữ liệu ảnh sẽ lần lượt được đưa qua bộ rút trích đặc trưng được cài đặt sẵn và do người dùng chỉ định. Đối với các đặc trưng rút trích từ hai phương pháp Color Histogram và Color Correlogram thì chúng sẽ được trực tiếp lưu xuống bộ dữ liệu đặc trưng, còn đối với các phương pháp SIFT, ORB và HOG sẽ được gom cụm thành các visual words và đưa vào Bag of Visual Words nhằm giảm số lượng đặc trưng cần xử lý, giúp rút ngắn thời gian đánh chỉ mục.

Sau khi quá trình rút trích đặc trưng đã hoàn thành, hệ thống tiếp tục ghi các đặc trưng rút trích được vào file thông qua kỹ thuật đánh chỉ mục tiến (forward indexing). Với từng đường dẫn của ảnh được gán kèm với vector đặc trưng của ảnh đó. Chương trình sẽ tạo ra một thư mục khác để lưu bộ dữ liệu đặc trưng này, đường dẫn của thư mục sẽ được tạo như sau:

- Đường dẫn đến thư mục chứa dữ liệu ảnh: ".../Desktop/training_images".
- Đường dẫn đến thư mục chứa dữ liệu đặc trưng: ".../Desktop/extracted_feature/SIFT".

Ngoài ra các file lưu trữ sẽ có định dạng .bin giúp đọc và ghi nhanh hơn.

2. Truy vấn ảnh và đánh giá kết quả (Giai đoạn Online)

Trong giai đoạn truy vấn, người dùng sẽ chọn ảnh truy vấn và file chỉ mục tương ứng để bắt đầu quá trình truy vấn. Các đặc trưng được rút trích từ ảnh truy vấn sẽ được so sánh với các đặc trưng được lưu để tìm ra ảnh nào gần với đặc trưng đó nhất sử dụng độ đo phù hợp với đặc trưng đó. Sau khi đã tính khoảng cách với toàn bộ đặc trưng được lưu, hệ thống sẽ thực hiện thuật toán sắp xếp từ thấp đến cao hoặc từ cao đến thấp tùy vào độ đo là độ khoảng cách hay độ tương đồng và lấy kTop kết quả đầu tiên.

Tiếp theo, để đánh giá kết quả đầu ra hệ thống sẽ xác định số lượng kết quả đúng thông qua file chỉ mục bằng cách so tên ảnh truy vấn với các tên của các ảnh trong file chỉ mục (vd ảnh truy vấn có tên là 01 thì hệ thống sẽ tìm các ảnh có tên 01 như 01_1, 01_2,... để làm kết quả đúng). Sau khi đã xác định được các kết quả đúng, hệ thống thực hiện quá trình tính mAP để đánh giá kết quả của hệ thống.

4.1 Các phương pháp rút trích đặc trưng

4.1.1 Color Histogram

Biểu đồ màu của một ảnh cho trước I là một phương pháp biểu diễn phân bố màu sắc trong ảnh [3]. Dối với một màu C_i bất kỳ trong ảnh, $H_{C_i}(I)$ được định nghĩa là số lượng pixel có màu C_i . Khi giá trị này được chuẩn hóa về khoảng $[0, 1]$, nó có thể được hiểu là xác suất một pixel bất kỳ trong ảnh thuộc về màu C_i . Nói cách khác, $H_{C_i}(I)$ đại diện cho khả năng xuất hiện của màu C_i trong toàn bộ ảnh. Biểu đồ màu của ảnh là một ánh xạ từ tập hợp các màu trong ảnh vào khoảng $[0, 1]$, trong đó tổng số pixel trong biểu đồ màu bằng tổng số pixel trong ảnh.

Một đặc điểm quan trọng của biểu đồ màu là nó không phụ thuộc vào hướng hay vị trí của các đối tượng trong ảnh, giúp biểu diễn đặc trưng màu sắc một cách tổng quát và không bị ảnh hưởng bởi sự thay đổi vị trí của các thành phần trong ảnh.

Các bước xây dựng lược đồ màu của một ảnh màu RGB:

1. Khởi tạo các mảng histogram

- Tạo ba mảng có kích thước 256 để lưu số lượng pixel cho từng giá trị màu của kênh Red, Green, và Blue. Gọi các mảng này lần lượt là H_R, H_G, H_B .
- Nếu ảnh là ảnh xám (grayscale), chỉ cần một mảng H có kích thước 256.

2. Tách các kênh màu của ảnh

- Nếu ảnh là ảnh màu RGB, tách thành ba kênh màu: Red (R), Green (G), Blue (B).

3. Xây dựng lược đồ màu

(a) Với mỗi màu R, G, B

- Lặp qua kênh màu và lấy giá trị màu của kênh đó $C_i(x, y)$
- Tăng giá trị của bin tương ứng trong các mảng histogram: $C_i(x, y) = C(x, y) + 1$

(b) Nếu ảnh là grayscale, chỉ cần cập nhật histogram của ảnh xám

- $H(I(x, y)) = H(I(x, y)) + 1$

Trong hệ thống truy hồi ảnh dựa trên nội dung (CBIR), không gian màu RGB bộc lộ một số điểm yếu khiến nó kém hiệu quả hơn so với không gian màu HSV. RGB là hệ màu phụ thuộc trực tiếp vào cường độ ánh sáng của ba kênh Red, Green và Blue. Do đó, khi điều kiện chiếu sáng thay đổi hoặc có bóng tối, phản xạ, thì giá trị RGB của cùng một điểm ảnh có thể thay đổi đáng kể, làm giảm độ chính xác của việc so sánh màu sắc giữa các ảnh. Ngoài ra, RGB không tách biệt rõ ràng giữa thông tin về màu sắc và độ sáng, khiến việc phân tích màu gặp nhiều nhiễu loạn từ yếu tố ánh sáng.

Ngược lại, HSV (Hue – màu sắc, Saturation – độ bão hòa, Value – độ sáng) là không gian màu mô phỏng gần hơn cách con người cảm nhận màu sắc. Nhờ sự phân tách rõ ràng giữa màu và độ sáng, HSV giúp hệ thống CBIR dễ dàng nhận diện và so sánh màu sắc thực sự của đối tượng mà ít bị ảnh hưởng bởi điều kiện ánh sáng. Đặc biệt, thành phần Hue (màu sắc) thường giữ ổn định ngay cả khi độ sáng thay đổi, giúp cải thiện độ chính xác trong việc so khớp đặc trưng màu. Chính vì vậy, HSV được ưu tiên sử dụng trong CBIR khi mục tiêu là phân biệt hình ảnh dựa trên màu sắc một cách hiệu quả và ổn định hơn so với RGB.

Các bước xây dựng lược đồ màu với hệ màu HSV

1. Chuyển đổi ảnh sang không gian màu HSV:

$$\mathbf{I}_{\text{HSV}} = \text{cvtColor}(\mathbf{I}_{\text{BGR}}, \text{COLOR_BGR2HSV}) \quad (1)$$

2. Tách các kênh màu Hue, Saturation, Value:

$$[H, S, V] = \text{split}(\mathbf{I}_{\text{HSV}}) \quad (2)$$

3. Xác định số lượng bin cho từng kênh:

$$\text{histSize} = [h_{\text{bins}}, s_{\text{bins}}, v_{\text{bins}}] = [16, 8, 8] \quad (3)$$

4. Đặt khoảng giá trị cho từng kênh:

$$H \in [0, 180], \quad S \in [0, 256], \quad V \in [0, 256] \quad (4)$$

5. Tính histogram 3 chiều trong không gian HSV:

$$\text{Hist}_{HSV}(h, s, v) = \text{calcHist}(\mathbf{I}_{HSV}, \text{channels}, \text{bins}, \text{ranges}) \quad (5)$$

Với

- channels = [0,1,2]
- bins = [16,8,8]
- ranges = [[0,180], [0,256], [0,256]]

6. Chuyển histogram từ 3D sang vector 1D:

$$\mathbf{f}_{HSV} = \text{reshape}(\text{Hist}_{HSV}, 1, h_{\text{bins}} \cdot s_{\text{bins}} \cdot v_{\text{bins}}) \quad (6)$$

7. Chuẩn hóa vector histogram:

$$\mathbf{f}_{HSV}^{\text{norm}} = \frac{\mathbf{f}_{HSV} - \min(\mathbf{f}_{HSV})}{\max(\mathbf{f}_{HSV}) - \min(\mathbf{f}_{HSV})} \quad (7)$$

8. Gán đặc trưng histogram cho ảnh:

$$\text{id} = \text{image_id}, \quad \text{imageDescriptors} = \mathbf{f}_{HSV}^{\text{norm}} \quad (8)$$

Algorithm 1: Trích xuất đặc trưng Color Histogram với hệ màu HSV

Input: Ảnh đầu vào `src_image`, chuỗi ký tự `image_id`

Output: Vector đặc trưng HSV `imageDescriptors`, ID ảnh `id`

- 1 Chuyển đổi `src_image` sang không gian màu HSV và lưu vào `hsv_image`;
- 2 Tách `hsv_image` thành 3 kênh: H, S, V;
- 3 Thiết lập số lượng bins cho histogram:;
- 4 $h_bins \leftarrow 16, s_bins \leftarrow 8, v_bins \leftarrow 8$;
- 5 `histSize` $\leftarrow [h_bins, s_bins, v_bins]$;
- 6 Thiết lập khoảng giá trị cho histogram:;
- 7 $h_range \leftarrow [0, 180], s_range \leftarrow [0, 256], v_range \leftarrow [0, 256]$;
- 8 `ranges` $\leftarrow [h_range, s_range, v_range]$;
- 9 Chỉ định các kênh cần dùng:;
- 10 `channels` $\leftarrow [0, 1, 2]$;
- 11 Tính histogram HSV 3 chiều:;
- 12 `hsv_hist` $\leftarrow \text{calcHist}(hsv_image, channels, histSize, ranges)$;
- 13 Chuyển đổi `hsv_hist` thành vector 1 chiều dạng hàng:;
- 14 `hsv_hist` $\leftarrow \text{reshape}(hsv_hist, 1, 1024)$;
- 15 Chuẩn hóa histogram vào khoảng [0, 1]:;
- 16 `hsv_hist` $\leftarrow \text{normalize}(hsv_hist, 0, 1)$;
- 17 Gán các giá trị đầu ra:;
- 18 `id` $\leftarrow \text{image_id}$;
- 19 `imageDescriptors` $\leftarrow \text{hsv_hist}$;

4.1.2 Color Correlogram

Color correlogram là một đặc trưng hình ảnh được đề xuất để khắc phục nhược điểm của histogram màu truyền thống vốn không lưu trữ thông tin không gian [4]. Trong khi histogram chỉ mô tả phân bố màu sắc toàn cục, correlogram mô tả mối tương quan không gian giữa các cặp màu sắc trong ảnh.

Cụ thể, với mỗi cặp màu (c_i, c_j) và một khoảng cách k , correlogram lưu xá xác suất mà một điểm ảnh có màu c_j sẽ xuất hiện ở khoảng cách k từ một điểm ảnh có màu c_i . Đặc trưng này thể hiện tốt cả thông tin toàn cục, từ đó cho phép so sánh ảnh hiệu quả ngay cả khi ảnh bị biến dạng do góc nhìn, độ sáng, hoặc phóng to/thu nhỏ.

Giả sử ảnh I là ảnh vuông kích thước $n \times n$, các màu được lượng tử hoá thành m màu c_1, c_2, \dots, c_m . Với mỗi điểm ảnh $p = (x, y)$, ta ký hiệu $I(p)$ là màu của điểm đó. Khoảng cách giữa hai điểm ảnh $p_1 = (x_1, y_1)$ và $p_2 = (x_2, y_2)$ được đo bằng chuẩn L_∞ :

$$\|p_1 - p_2\| = \max(|x_1 - x_2|, |y_1 - y_2|)$$

Histogram màu chuẩn hóa của ảnh I được định nghĩa như sau:

$$h_{c_i}(I) = \frac{1}{n^2} \Pr_{p \in I}[I(p) = c_i]$$

Correlogram được định nghĩa với khoảng cách k như sau:

$$\gamma_{c_i, c_j}^{(k)}(I) = \Pr_{p_1 \in I_{c_i}, p_2 \in I_{c_j}}[\|p_1 - p_2\| = k]$$

Trong đó I_{c_i} là tập các điểm ảnh có màu c_i .

Nếu chỉ xét đến mối tương quan giữa các điểm ảnh cùng màu ($c_i = c_j$), ta có autocorrelogram:

$$\alpha_c^{(k)}(I) = \gamma_{c,c}^{(k)}(I)$$

Tương tự như ở trên ta cũng có thể áp dụng hệ màu HSV để tăng độ chính xác trong quá trình truy vấn. Để xây dựng Color Correlogram cho một ảnh I , ta thực hiện các bước sau:

1. Lượng tử hóa màu sắc (Color Quantization)

Ánh xạ ảnh RGB ban đầu thành một tập hợp hữu hạn m màu $\{c_1, c_2, \dots, c_m\}$ thông qua lượng tử hóa. Ví dụ, lượng tử RGB về $4 \times 4 \times 4 = 64$ màu.

2. Chọn tập khoảng cách D

Xác định tập khoảng cách không gian $D = \{d_1, d_2, \dots, d_k\}$ cần xét (ví dụ: $D = \{1, 3, 5, 7\}$). Mỗi khoảng cách thể hiện mức độ gần-xa không gian giữa các pixel.

3. Duyệt từng pixel trong ảnh

Với mỗi pixel p_1 trong ảnh có màu c_i , thực hiện:

- Duyệt qua các điểm ảnh p_2 thỏa mãn $\|p_1 - p_2\| = d$ với $d \in D$, theo chuẩn L_∞ :

$$\|p_1 - p_2\| = \max(|x_1 - x_2|, |y_1 - y_2|) = d$$

- Kiểm tra nếu $I(p_2) = c_j$ thì tăng biến đếm $\Gamma_{c_i, c_j}^{(d)}$ lên 1.

4. Chuẩn hoá xác suất

Sau khi đếm xong, tính xác suất xuất hiện của mỗi cặp màu tại khoảng cách d bằng công thức:

$$\gamma_{c_i, c_j}^{(d)} = \frac{\Gamma_{c_i, c_j}^{(d)}}{h_{c_i}(I) \cdot 8d}$$

Trong đó $h_{c_i}(I)$ là tần suất xuất hiện của màu c_i :

$$h_{c_i}(I) = \frac{|\{p \mid I(p) = c_i\}|}{n^2}$$

5. Lưu vector đặc trưng

Lưu lại toàn bộ các giá trị $\gamma_{c_i, c_j}^{(d)}$ với $c_i, c_j \in \{1, \dots, m\}$ và $d \in D$ thành vector đặc trưng correlogram. Kích thước vector là $m \times m \times |D|$.

6. Tối ưu bằng autocorrelogram

Nếu chỉ quan tâm đến mối quan hệ giữa các pixel cùng màu ($c_i = c_j$), có thể tính autocorrelogram:

$$\alpha_c^{(d)} = \gamma_{c,c}^{(d)}$$

Lúc này, vector đặc trưng chỉ có kích thước $m \times |D|$.

Algorithm 2: Trích xuất đặc trưng Color Correlogram với hệ màu HSV

Input: Ảnh màu *image*, chuỗi định danh *image_id*

Output: Vector đặc trưng *imageDescriptors*, định danh *id*

```

1 Khởi tạo: distances  $\leftarrow \{1, 3, 5, 7\}, bins  $\leftarrow 144$ ;
2 correlogram  $\leftarrow$  vector độ dài 576 với giá trị khởi tạo bằng 0;
3 Chuyển image sang HSV  $\rightarrow$  hsv_image;
4 foreach pixel (i, j) trong hsv_image do
5   | Tính chỉ số lượng tử màu bằng hàm colorQuantization;
6   | Gán vào ma trận quantized[i][j];
7 end
8 foreach d  $\in$  distances do
9   | colorCount  $\leftarrow$  vector độ dài 144 khởi tạo 0;
10  | totalMatches  $\leftarrow 0$ ;
11  | Lấy 8 hướng láng giềng tại khoảng cách d;
12  | foreach pixel (x, y) theo bước sampling do
13    |   | c1  $\leftarrow$  quantized[x][y];
14    |   | foreach offset (dx, dy) trong 8 hướng do
15      |     |   | nx  $\leftarrow$  x + dx, ny  $\leftarrow$  y + dy;
16      |     |   | if nx, ny nằm trong ảnh then
17        |       |   | c2  $\leftarrow$  quantized[nx][ny];
18        |       |   | if c1 == c2 then
19          |         |   |   | colorCount[c1]  $\leftarrow$  colorCount[c1] + 1;
20          |         |   |   | totalMatches  $\leftarrow$  totalMatches + 1;
21        |       |   | end
22      |   | end
23    | end
24  | end
25  | if totalMatches > 0 then
26    |   | for i  $\leftarrow 0$  to 143 do
27      |     |   | correlogram[d  $\cdot$  144 + i]  $\leftarrow$   $\frac{\text{colorCount}[i]}{\text{totalMatches}}$ ;
28    |   | end
29  | end
30 end
31 Chuyển correlogram thành vector 1 dòng  $\rightarrow$  imageDescriptors;
32 id  $\leftarrow$  image_id;$ 
```

4.1.3 Histogram of Oriented Gradients

Lược đồ hướng cạnh là một phương pháp biểu diễn đặc trưng của ảnh dựa trên hướng cạnh tại mỗi pixel. Kỹ thuật này giúp mô tả cấu trúc và hình dạng của đối tượng trong ảnh bằng cách thống kê tần suất xuất hiện của các hướng cạnh trong một miền nhất định. Ưu điểm của lược đồ hướng cạnh bao gồm:

- Không bị ảnh hưởng bởi sự thay đổi cường độ sáng của ảnh.
- Giúp biểu diễn hình dạng của đối tượng trong ảnh một cách hiệu quả.
- Được sử dụng rộng rãi trong các bài toán nhận diện đối tượng, phát hiện cạnh, và phân loại ảnh.

Các bước xây dựng lược đồ hướng cạnh bao gồm [5]:

1. Tính đạo hàm của ảnh

- (a) Tính đạo hàm của ảnh theo hai hướng x, y bằng cách sử dụng phép tích chập ảnh với hai kernel sau:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad G_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

- (b) Tính độ lớn và hướng của đạo hàm thông qua hai công thức:

$$g = \sqrt{g_x^2 + g_y^2}$$

$$\theta = \arctan \frac{g_y}{g_x}$$

2. Tính lược đồ hướng cạnh

- (a) Chia ảnh thành các ô (cells), ví dụ 8×8 pixel.
- (b) Xây dựng biểu đồ hướng cạnh cho mỗi ô bằng cách phân chia các góc từ 0° đến 180° hoặc 360° thành các khoảng nhỏ (bins) thông qua công thức:

$$\text{bin_length} = \frac{360^\circ}{\text{number_of_bins}}$$

Trong đó:

- bin_length: là độ rộng của mỗi bin.
- number_of_bins: là số bin mà chúng ta muốn chia ra.

- (c) Với mỗi ô 8×8 ta thêm các góc vào trong lược đồ hướng cạnh.
- (d) Biên độ được chia tỉ lệ giữa hai bins gần nhất dựa trên khoảng cách góc, sử dụng công thức sau:

$$proportion = \frac{bin_length - different}{bin_length}$$

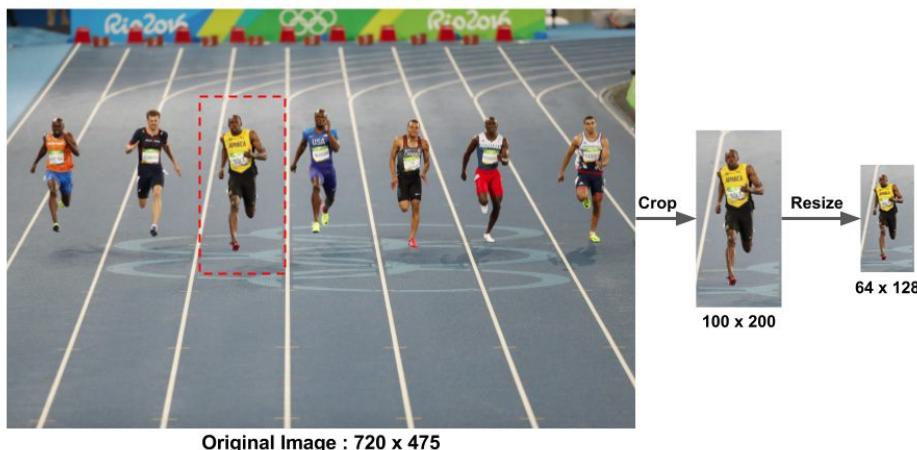
Trong đó:

- proportion: Tỉ lệ cho bin gần với giá trị góc hiện tại
- bin_length: là độ rộng của mỗi bin
- different: à khoảng cách giữa góc θ và trung tâm của bin gần nhất.

Để hiểu rõ hơn về cách tạo lược đồ hướng cạnh ta hãy cùng xem qua một ví dụ như sau:

1. Ảnh đầu vào:

Ảnh đầu vào được lấy từ một ảnh lớn hơn có kích thước 720×475 sau đó được điều chỉnh kích thước lại thành 64×128 để tiện cho việc chia ô 8×8 trong các công đoạn sau vì 64 và 128 đều chia hết cho 8 .



Hình 3: Ảnh đầu vào

2. Tính đạo hàm của ảnh:

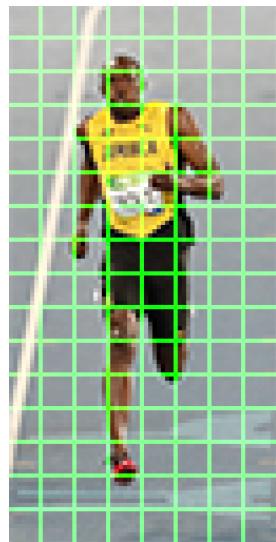
Sau khi áp dụng bộ lọc Sobel lên ảnh đầu vào ta có được các đạo hàm, độ lớn như sau:



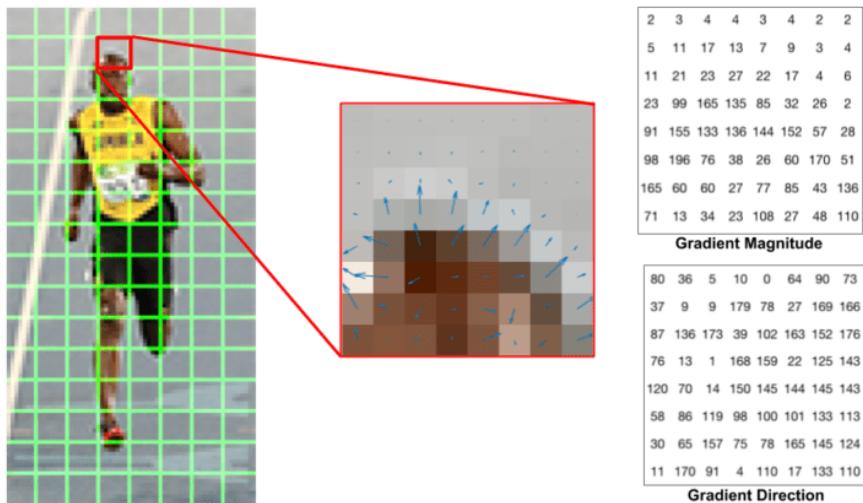
Hình 4: Giá trị tuyệt đối của đạo hàm theo X (trái), theo Y (giữa), và độ lớn đạo hàm (phải)

3. Chia ảnh thành các ô nhỏ:

Tiếp đến ta chia ảnh thành các ô nhỏ hơn.



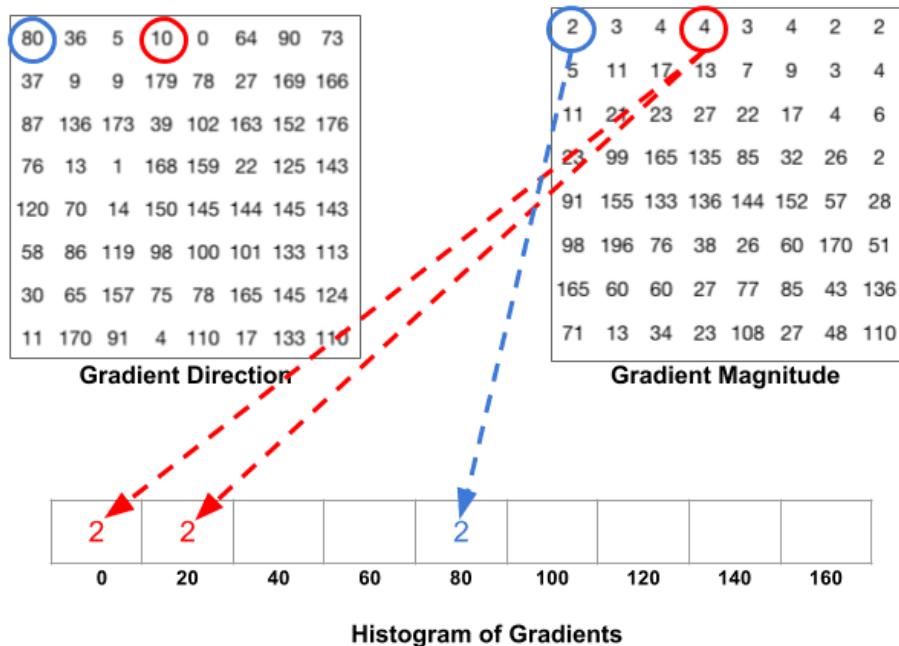
Hình 5: Ảnh được chia thành các ô 8×8



Hình 6: Giá trị độ lớn và hướng của các đạo hàm trong một ô

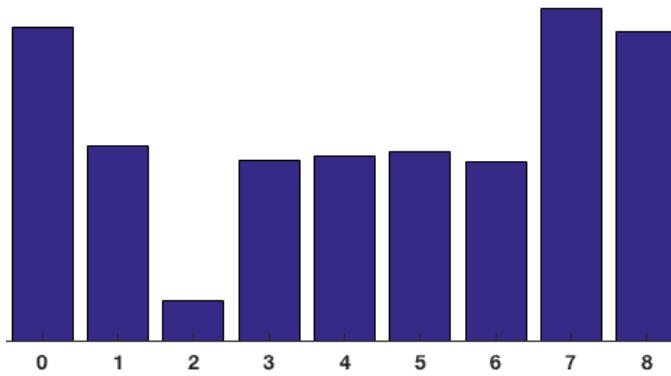
4. Tính lược đồ hướng cạnh:

Để tính được lược đồ hướng cạnh ta xét vùng được tô đỏ ở hình trước đó:



Hình 7: Ví dụ chọn bin dựa vào hướng và độ lớn của đạo hàm

Đối với góc được khoanh màu xanh, góc có giá trị là 80 độ và độ lớn là 2 nên ta thêm 2 vào thùng thứ 5. Góc tại pixel được khoanh tròn bằng màu đỏ là 10 độ và độ lớn là 4. Vì 10 độ nằm giữa 0 và 20, nên ta sẽ chia đều độ lớn vào hai bin 0 và 20. Lặp lại với các góc khác ta sẽ có được histogram hoàn chỉnh cho ô này như sau:

Hình 8: Lược đồ hướng cạnh của một ô 8×8 **Algorithm 3:** Trích xuất đặc trưng HOG

Input: Ảnh `src_image`, chuỗi định danh `image_id`

Output: Vector đặc trưng HOG `imageDescriptors`

- 1 Chuyển ảnh sang kiểu `CV_32F`, chuẩn hóa về $[0,1]$;
- 2 Tính ảnh đạo hàm theo x và y :

`gx` \leftarrow Sobel(`src_image`, $dx = 1, dy = 0$), `gy` \leftarrow Sobel(`src_image`, $dx = 0, dy = 1$)

- 3 Tính độ lớn và góc hướng:

`(mag, angle)` \leftarrow cartToPolar(`gx`, `gy`, `angleInDegrees = true`)

- 4 Gọi hàm `computeHOG(mag, angle, isWeighted=true)` để tạo histogram;

- 5 Gán đầu ra:

`id` \leftarrow `image_id`, `imageDescriptors` \leftarrow `hogHistogram`

Algorithm 4: Tính toán Histogram of Oriented Gradients (HOG)

Input: Ma trận độ lớn mag , ma trận góc angle , cờ trọng số isWeighted

Output: Vector đặc trưng HOG $\mathbf{f} \in \mathbb{R}^{360}$

```

1 Khởi tạo:  $\text{featureDim} \leftarrow 360$ ,  $\mathbf{f} \leftarrow$  vector 360 phần tử 0;
2 for  $i \leftarrow 0$  to 359 do
3   | Tính:  $\text{uplimits}[i] \leftarrow (2i + 1) \cdot \frac{360}{2 \cdot 360}$ ,  $\text{medbins}[i] \leftarrow i \cdot \frac{360}{360}$ 
4 end
5 foreach pixel  $(x, y)$  do
6   |  $a \leftarrow \text{angle}[x][y]$ ,  $m \leftarrow \text{mag}[x][y]$ ;
7   | if  $\text{isWeighted} = \text{false}$  then
8     |   | Tìm bin  $k$  chứa  $a$  theo  $\text{uplimits}$ , cộng  $m$  vào  $\mathbf{f}[k]$ ;
9   | else
10    |   | Xác định hai bin lân cận  $b_1, b_2$  gần  $a$ ;
11    |   | Tính khoảng cách  $d$  giữa  $a$  và bin gần nhất;
12    |   | Tính trọng số phân bổ:
13    |   |  $w_1 = \frac{\text{binLength} - d}{\text{binLength}}$ ,  $w_2 = 1 - w_1$ 
14    |   | Cộng  $m \cdot w_1$  vào  $\mathbf{f}[b_1]$ ,  $m \cdot w_2$  vào  $\mathbf{f}[b_2]$ ;
15 end
16 end
17 return  $\mathbf{f}$ 

```

4.1.4 SIFT

SIFT là một thuật toán được David Lowe đề xuất nhằm phát hiện và mô tả các điểm đặc trưng cục bộ trong ảnh với tính chất bất biến theo tỉ lệ, xoay và một phần thay đổi về ánh sáng. Thuật toán này rất hiệu quả trong các ứng dụng như nhận dạng đối tượng, ghép ảnh, theo vết và tìm kiếm hình ảnh.

Các bước áp dụng thuật toán SIFT lên ảnh bao gồm:

- Tạo không gian tỉ lệ và phát hiện điểm cực trị**

Để phát hiện các đặc trưng ổn định theo tỉ lệ, ảnh được xử lý qua bộ lọc Gaussian ở nhiều mức độ σ khác nhau để tạo thành không gian tỉ lệ:

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

Trong đó:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

Hàm hiệu Gaussian (Difference of Gaussian - DoG) được tính như sau:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

Các điểm cực trị trong $D(x, y, \sigma)$ được xác định bằng cách so sánh với 26 điểm lân cận trong không gian và tỉ lệ.

- Lọc điểm đặc trưng yếu và loại bỏ biên** Mỗi điểm cực trị được nội suy vị trí chính xác hơn bằng khai triển Taylor bậc hai:

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

Để loại bỏ các điểm nằm trên biên cạnh, sử dụng ma trận Hessian:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Tỉ lệ:

$$R = \frac{(\text{Tr}(H))^2}{\det(H)} = \frac{(D_{xx} + D_{yy})^2}{D_{xx}D_{yy} - D_{xy}^2}$$

Nếu $R > r_{\text{threshold}}$, điểm sẽ bị loại bỏ.

- Gán hướng cho điểm đặc trưng**

Gradient độ lớn và hướng tại mỗi điểm ảnh (x, y) được tính bằng:

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

$$\theta(x, y) = \tan^{-1} \left(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)} \right)$$

Một histogram hướng (36 bins) được xây dựng trong vùng lân cận. Hướng có tần suất lớn nhất được chọn là hướng chính cho điểm đặc trưng.

- Mô tả đặc trưng**

Mỗi vùng 16×16 quanh điểm đặc trưng được chia thành 4×4 ô nhỏ. Mỗi ô tạo một histogram 8 hướng gradient. Tổng cộng, vector đặc trưng có 128 chiều:

$$\mathbf{f} = [h_{1,1}, h_{1,2}, \dots, h_{4,4}] \quad \text{với} \quad h_{i,j} \in \mathbb{R}^8$$

Cuối cùng, vector đặc trưng được chuẩn hóa:

$$\mathbf{f} = \frac{\mathbf{f}}{\|\mathbf{f}\|}$$

Algorithm 5: Trích xuất đặc trưng SIFT

```

Input: Ảnh image, chuỗi định danh image_id
Output: Descriptor SIFT imageDescriptors, id
1 id ← image_id;
   // Chuyển ảnh về thang độ xám nếu ảnh màu
2 if image.channels() == 3 then
3   | gray ← cvtColor(image, COLOR_BGR2GRAY);
4 end
5 else
6   | gray ← bản sao của image;
7 end
   // Tạo đối tượng SIFT
8 sift ← SIFT::create();
   // Phát hiện và mô tả keypoints
9 sift.detectAndCompute(gray, noArray(), keypoints, descriptors);
   // Chuyển về kiểu CV_32F nếu cần
10 if descriptors không rỗng và type ≠ CV_32F then
11   | descriptors.convertTo(descriptors, CV_32F);
12 end
13 imageDescriptors ← descriptors;

```

4.1.5 ORB

ORB là một thuật toán trích xuất đặc trưng hiệu quả, được thiết kế như một giải pháp thay thế miễn phí bản quyền cho các thuật toán như SIFT và SURF. ORB kết hợp giữa hai thành phần chính:

- **FAST** (Features from Accelerated Segment Test) để phát hiện điểm đặc trưng.
- **BRIEF** (Binary Robust Independent Elementary Features) để mô tả đặc trưng.

ORB cải tiến FAST và BRIEF để đạt được tính bất biến theo xoay và một phần theo tỉ lệ, đồng thời hoạt động với hiệu suất cao, phù hợp cho các ứng dụng thời gian thực.

Các bước thực hiện thuật toán ORB:

- 1. Phát hiện điểm đặc trưng với FAST trên ảnh đa tỉ lệ**

Áp dụng thuật toán FAST trên ảnh gốc và các ảnh trong ảnh kim tự tháp (image pyramid) để phát hiện điểm đặc trưng ở nhiều cấp độ tỉ lệ khác nhau.

- 2. Lọc và xếp hạng điểm đặc trưng bằng Harris Corner Score**

Với mỗi điểm phát hiện bởi FAST, tính chỉ số Harris để sắp xếp và chọn ra các điểm đặc trưng mạnh nhất.

- 3. Gán hướng đặc trưng (Orientation Assignment)**

Tính moment cấp bậc đầu tiên trong vùng xung quanh mỗi điểm đặc trưng để xác định hướng chính:

$$m_{10} = \sum_{x,y} x \cdot I(x,y), \quad m_{01} = \sum_{x,y} y \cdot I(x,y)$$

Hướng chính được tính bởi:

$$\theta = \tan^{-1} \left(\frac{m_{01}}{m_{10}} \right)$$

4. Tạo mô tả đặc trưng với BRIEF có xoay

Với mỗi điểm đặc trưng, áp dụng BRIEF descriptor bằng cách so sánh cường độ của các cặp điểm ảnh trong vùng lân cận. Các mẫu BRIEF được xoay theo góc θ đã gán:

$$\text{descriptor}[i] = \begin{cases} 1, & \text{nếu } I(p_i) < I(q_i) \\ 0, & \text{ngược lại} \end{cases}$$

với (p_i, q_i) là các cặp điểm đã xoay quanh điểm đặc trưng.

5. Ghép đặc trưng với khoảng cách Hamming

Các mô tả nhị phân được so sánh bằng khoảng cách Hamming để tìm các cặp điểm tương đồng giữa hai ảnh:

$$\text{Hamming}(d_1, d_2) = \sum_{i=1}^n d_1[i] \oplus d_2[i]$$

Algorithm 6: Trích xuất đặc trưng ORB

```

Input: Ảnh image, chuỗi định danh image_id
Output: Descriptor ORB imageDescriptors, id
1 id  $\leftarrow$  image_id;
   // Chuyển ảnh về thang độ xám nếu ảnh có 3 kênh
2 if image.channels() == 3 then
3   | gray  $\leftarrow$  cvtColor(image, COLOR_BGR2GRAY);
4 end
5 else
6   | gray  $\leftarrow$  bản sao của image;
7 end
   // Tạo đối tượng ORB
8 orb  $\leftarrow$  ORB::create();
   // Phát hiện và mô tả keypoints
9 orb.detectAndCompute(gray, noArray(), keypoints, descriptors);
   // Chuyển sang kiểu CV_32F nếu cần thiết
10 if descriptors không rỗng và type  $\neq$  CV_32F then
11   | descriptors.convertTo(descriptors, CV_32F);
12 end
13 imageDescriptors  $\leftarrow$  descriptors;

```

4.2 Bag of Visual Word

Mô hình **Bag of Visual Words (BoVW)** [6] là một kỹ thuật phổ biến trong lĩnh vực truy vấn và nhận dạng ảnh, được xây dựng dựa trên ý tưởng tương tự như mô hình Bag of Words trong xử lý ngôn ngữ tự nhiên. Thay vì xử lý từ ngữ, BoVW biểu diễn ảnh thông qua các “từ thị giác” (*visual words*) – các trung tâm cụm (cluster centers) được sinh ra từ các mô tả đặc trưng cục bộ như SIFT, SURF, ORB, v.v.

Xây dựng từ điển thị giác (Visual Vocabulary)

Đầu tiên, tất cả các mô tả đặc trưng từ tập ảnh huấn luyện được trích xuất và kết hợp thành một ma trận duy nhất. Sau đó, thuật toán phân cụm K-means được áp dụng để chia các mô tả này thành K cụm. Mỗi tâm cụm sẽ đại diện cho một “từ thị giác” trong từ điển.

- Cho $D = \{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_n\}$ là tập hợp tất cả descriptor đầu vào, với mỗi $\mathbf{d}_i \in \mathbb{R}^d$.
- Áp dụng thuật toán K-means để phân cụm thành K cụm:

$$\{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_K\} = \text{kmeans}(D)$$

- Tập $\{\mathbf{c}_1, \dots, \mathbf{c}_K\}$ chính là từ điển thị giác.

Biểu diễn ảnh bằng BoVW histogram

Sau khi có từ điển thị giác, mỗi ảnh đầu vào được mô tả bởi một histogram BoVW có kích thước $1 \times K$, trong đó mỗi phần tử biểu thị số lượng descriptor của ảnh đó thuộc về từng “từ thị giác”.

- Với ảnh mới, trích xuất tập descriptor $D' = \{\mathbf{d}'_1, \dots, \mathbf{d}'_m\}$.
- Với mỗi descriptor \mathbf{d}'_i , tìm “từ thị giác” gần nhất theo khoảng cách L2:

$$j^* = \arg \min_{j \in \{1, \dots, K\}} \|\mathbf{d}'_i - \mathbf{c}_j\|_2$$

- Tăng giá trị histogram tại vị trí j^* lên 1.
- Sau khi duyệt hết các descriptor, chuẩn hóa histogram theo chuẩn L2:

$$\hat{\mathbf{h}} = \frac{\mathbf{h}}{\|\mathbf{h}\|_2}$$

Ưu điểm của BoVW

- Giảm số chiều biểu diễn ảnh từ hàng ngàn descriptor thành vector có kích thước cố định K .
- Cho phép sử dụng các mô hình học máy truyền thống như SVM, KNN để phân loại/truy vấn ảnh.
- Có thể kết hợp với các đặc trưng cục bộ mạnh như SIFT, ORB để tăng tính phân biệt.

Algorithm 7: Build Visual Vocabulary (BoVW)

Input: Danh sách các descriptor: `allDescriptors`, kích thước từ điển: `dictionarySize`

Output: Từ điển thị giác `vocabulary`

```

1 Khởi tạo descriptorsStacked rỗng;
2 foreach desc trong allDescriptors do
3   if desc không rỗng then
4     if desc không phải kiểu CV_32F then
5       | Chuyển desc sang floatDesc với kiểu CV_32F;
6     end
7     else
8       | floatDesc  $\leftarrow$  desc;
9     end
10    if floatDesc có 2 chiều then
11      | Nối floatDesc vào descriptorsStacked;
12    end
13    else
14      | In cảnh báo: "Descriptor có kích thước không hợp lệ";
15    end
16  end
17 end
18 if descriptorsStacked rỗng then
19   In lỗi: "Không có descriptor để xây từ điển";
20   return;
21 else if descriptorsStacked.rows < dictionarySize then
22   In lỗi: "Số lượng descriptor không đủ cho dictionary";
23   return;
24 Áp dụng K-means lên descriptorsStacked để phân cụm thành dictionarySize cụm;
25 vocabulary  $\leftarrow$  các tâm cụm (cluster centers);

```

Algorithm 8: ComputeBoVWHistogram

Input: Image descriptors \mathbf{D} , vocabulary $\mathcal{V} = \{\mathbf{c}_1, \dots, \mathbf{c}_K\}$

Output: Histogram $\mathbf{h} \in \mathbb{R}^K$

```

1 Initialize  $\mathbf{h} \leftarrow \mathbf{0}$ ;
2 foreach  $\mathbf{d}_i \in \mathbf{D}$  do
3    $j^* \leftarrow \arg \min_j \|\mathbf{d}_i - \mathbf{c}_j\|_2$ ;
4    $h_{j^*} \leftarrow h_{j^*} + 1$ 
5 end
6 Normalize  $\mathbf{h}$ :  $\mathbf{h} \leftarrow \frac{\mathbf{h}}{\|\mathbf{h}\|_2}$ ;
7 return  $\mathbf{h}$ 

```

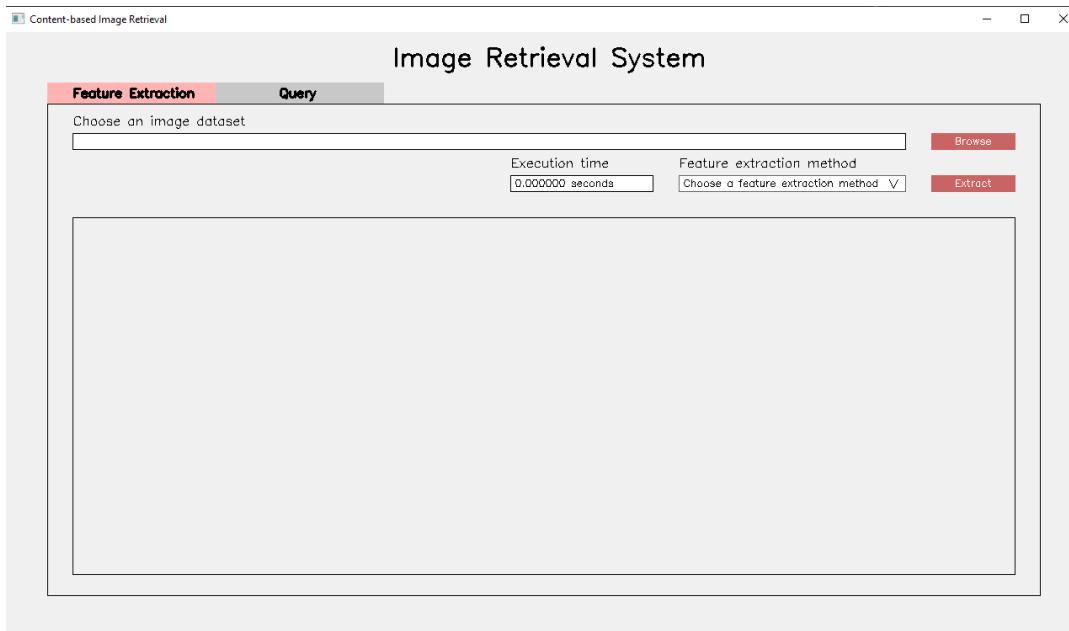
5 Giao diện hệ thống

Trong phần này, em sẽ giới thiệu về giao diện của hệ thống các thành phần trong giao diện và cách sử dụng chương trình. Chương trình có thể được mở ra từ thư mục release trong bài nộp, và video demo cũng được cung cấp nhằm giúp thầy/cô dễ dàng nắm bắt được cách hoạt động của hệ thống.

Các nút được sử dụng trong chương trình:

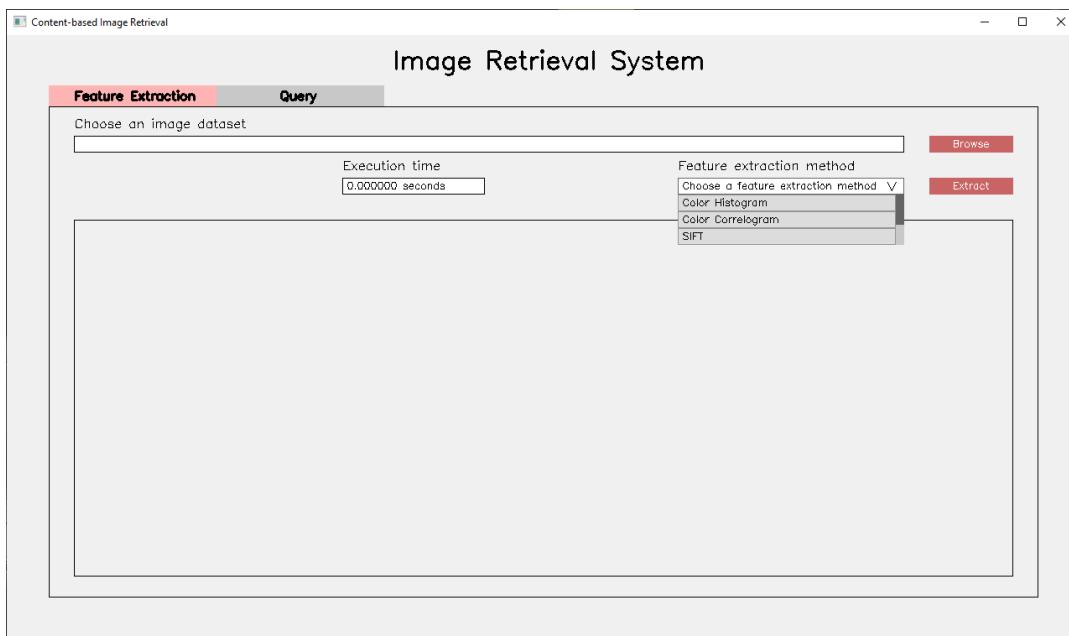
- **Esc**: thoát chương trình.
- **W và S**: di chuyển danh sách chọn phương pháp đặc trưng.

5.1 Giao diện trích xuất đặc trưng



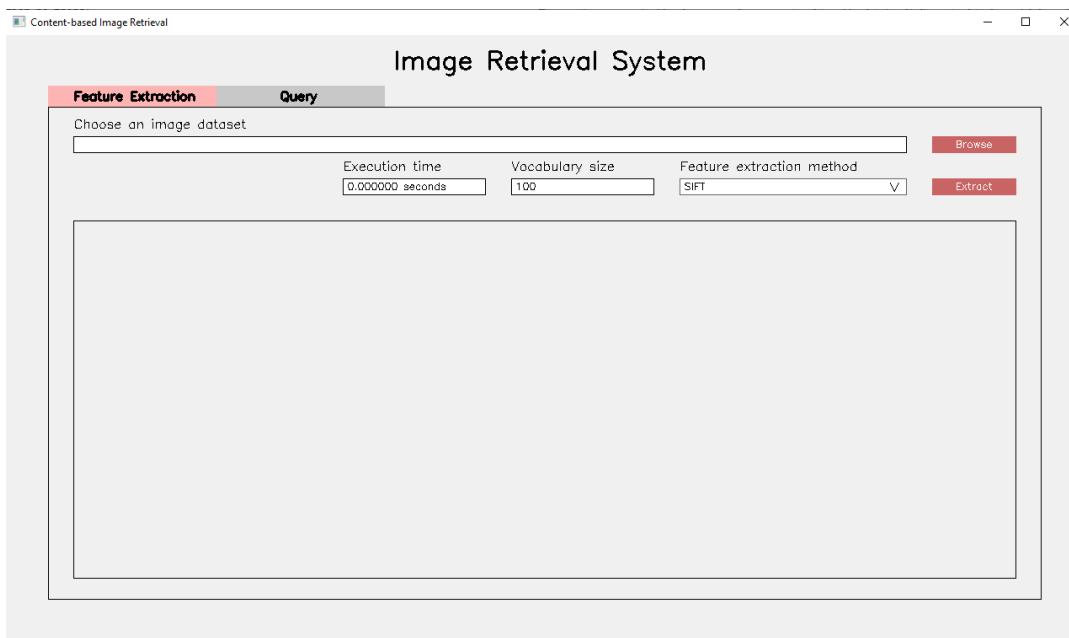
Hình 9: Giao diện trích xuất đặc trưng

Giao diện này cho phép người dùng chọn một tập dữ liệu ảnh và đặc trưng cần rút trích, sau đó chương trình sẽ thực hiện giai đoạn một để xây dựng bộ dữ liệu đặc trưng.



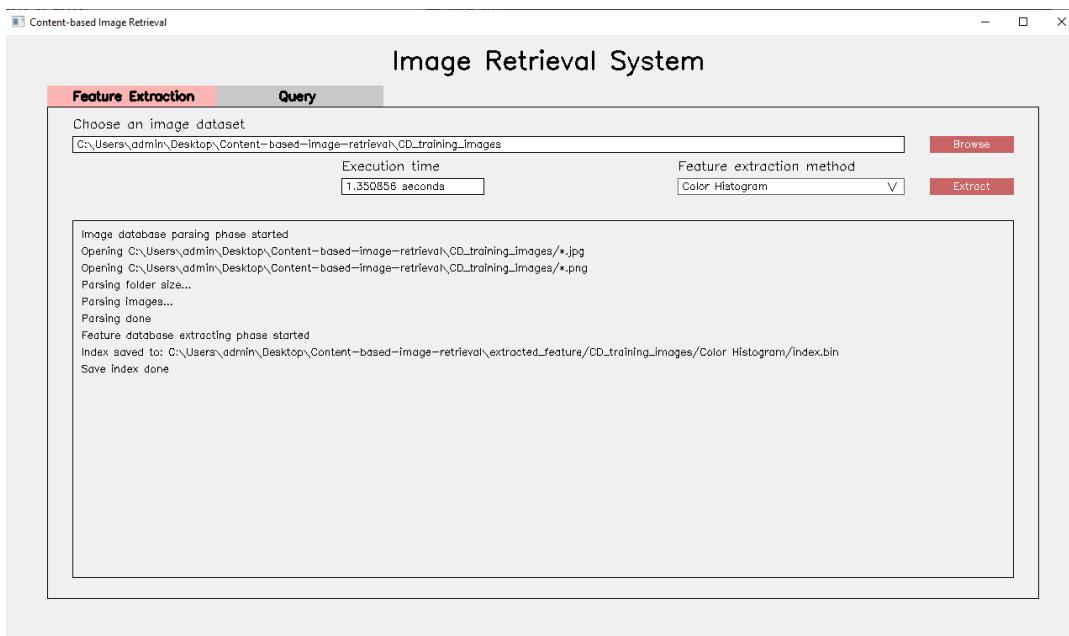
Hình 10: Giao diện chọn phương pháp rút trích đặc trưng

Ngoài ra khi chọn vào các phương pháp rút trích đặc trưng như SIFT, HOG, ORB thì chương trình sẽ hiện thị thêm một khung đầu vào khác cho phép nhập vào kích thước của vocabulary.



Hình 11: Giao diện nhập kích thước của vocabulary

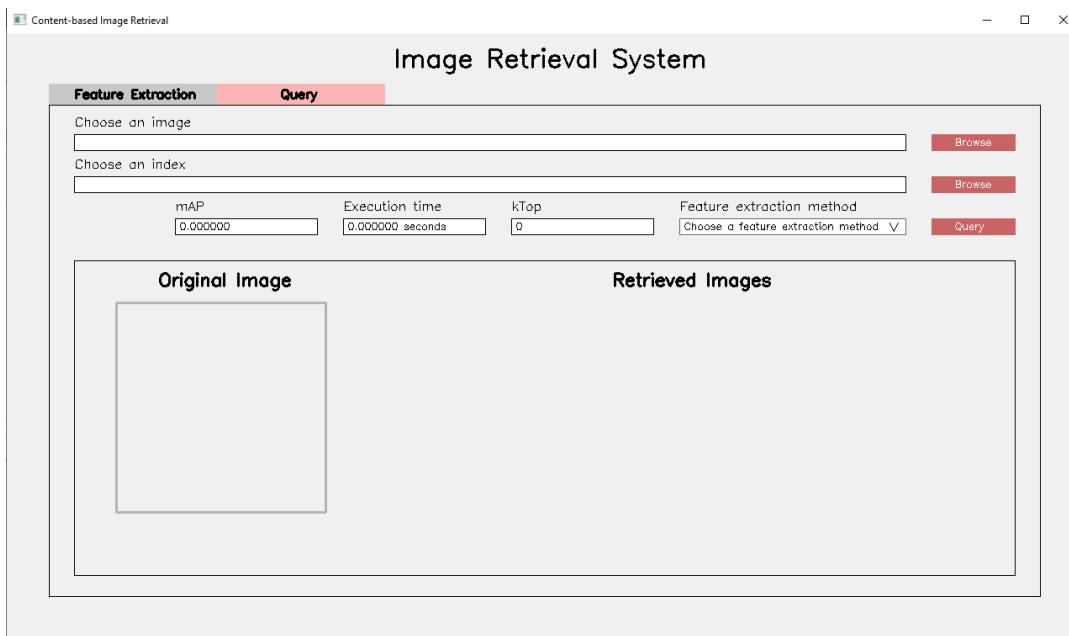
Sau khi hoàn thành việc rút trích đặc trưng trên bộ dữ liệu ảnh đầu vào, hệ thống sẽ đưa ra thông báo nhằm thông báo cho người dùng về trạng thái của hệ thống



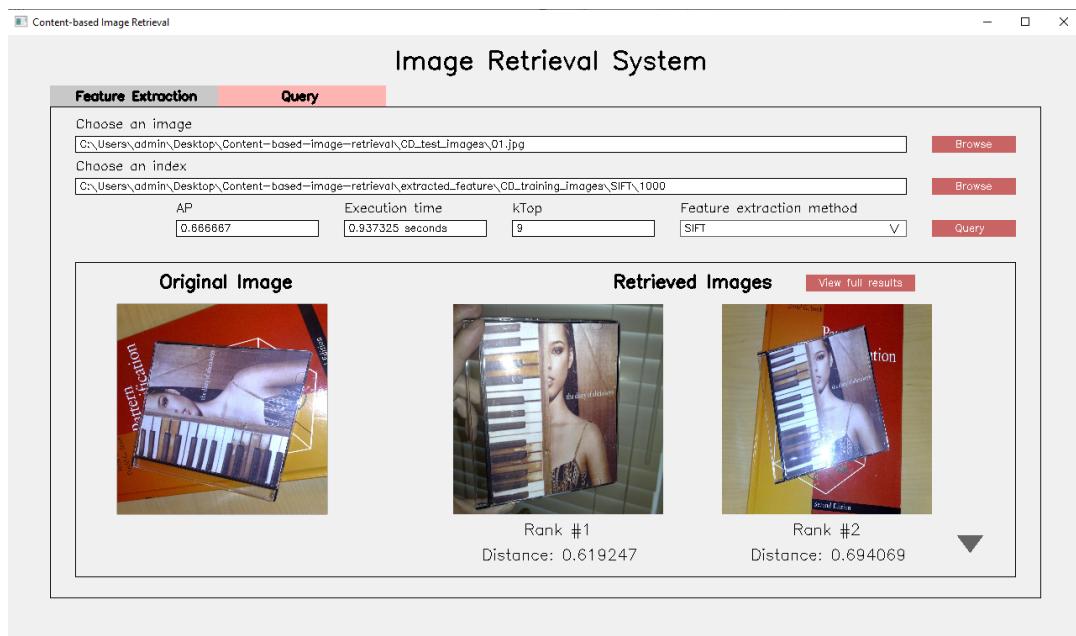
Hình 12: Giao diện khi rút trích đặc trưng thành công

5.2 Giao diện truy vấn ảnh

Giao diện này cho phép người dùng truy vấn ảnh bằng cách chọn một ảnh và tệp tin index đã được tạo ra trong quá trình trích xuất đặc trưng. Ảnh được chọn sẽ hiển thị bên trái và các ảnh truy vấn được sẽ được hiển thị bên phải theo thứ tự giảm dần từ trái phải và người dùng có thể bấm vào mũi tên để xem các ảnh khác nhau như hình bên dưới.



Hình 13: Giao diện truy vấn ảnh



Hình 14: Giao diện truy vấn ảnh sau khi truy vấn thành công

Nếu muốn xem toàn bộ các ảnh đã được truy vấn thì, chúng ta có thể nhấp chọn vào nút "view full results", từ đó một cửa sổ mới sẽ được hiện ra hiển thị toàn bộ các ảnh kết quả.



Hình 15: Giao diện xem tất cả các ảnh truy vấn

6 Thí nghiệm

Trong phần này, em sẽ đề cập đến các thí nghiệm được thực hiện trong lab này nhằm kiểm tra hiệu năng của hệ thống thông qua các bộ dữ liệu được cung cấp. Mục tiêu và kết quả của từng thí nghiệm sẽ được thể hiện chi tiết trong phần này.

6.1 Thiết lập thí nghiệm

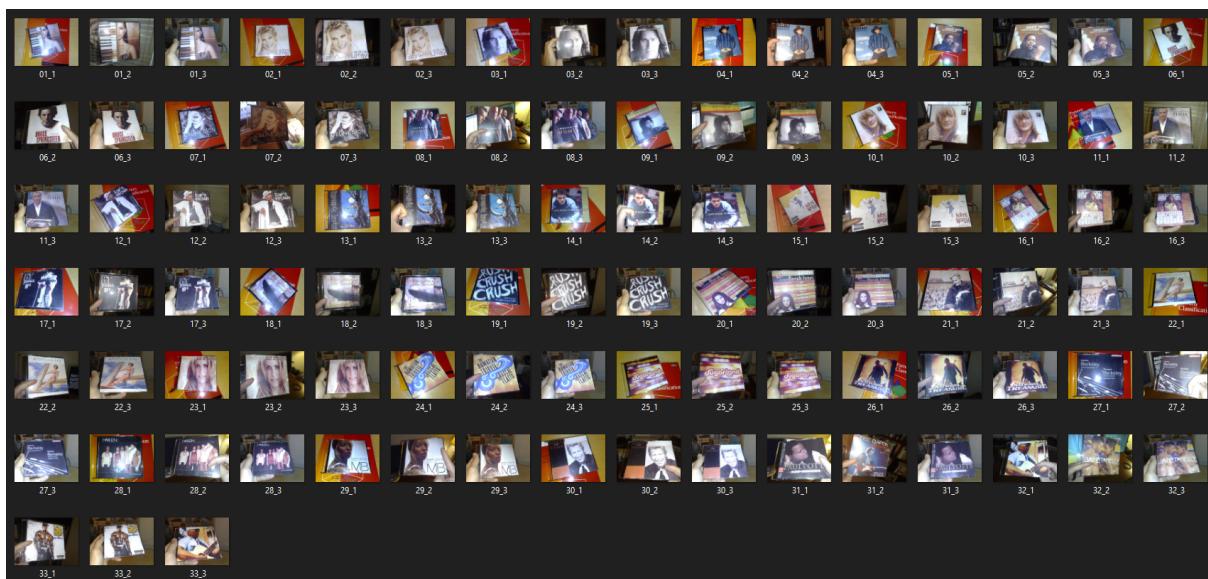
6.1.1 Thiết lập phần cứng

Cấu hình máy tính được dùng cho bài tập này như sau: CPU: i5-9600K, RAM: Gskill TridentZ 16GB x 4, Mainboard: B360m aorus gaming 3, PSU: Cooler Master Elite V3 500.

6.1.2 Bộ dữ liệu

- **Bộ dữ liệu CD**

Bộ dữ liệu CD bao gồm 99 ảnh huấn luyện và 10 ảnh để kiểm tra trong đó các ảnh kiểm tra được chọn từ các ảnh có trong tập huấn luyện nhưng được điều chỉnh để có sự khác biệt giữa hai bộ dữ liệu.



Hình 16: Tập dữ liệu huấn luyện của bộ dữ liệu CD



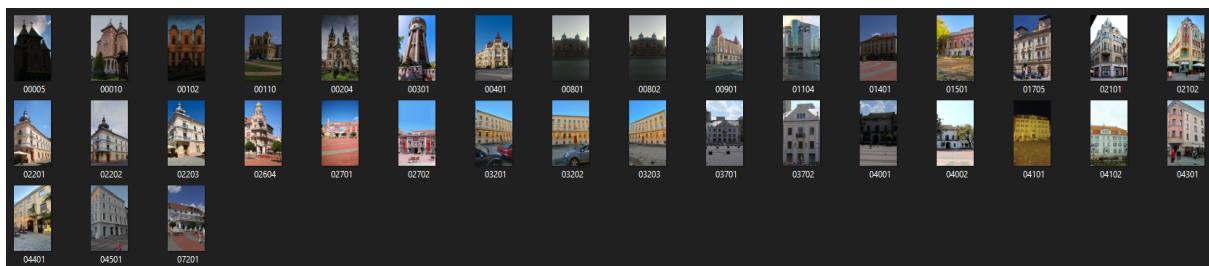
Hình 17: Tập dữ liệu kiểm tra của bộ dữ liệu CD

- **Bộ dữ liệu TMBuD**

Bộ dữ liệu TMBuD [7] là một bộ dữ liệu về các công trình kiến trúc với tổng cộng 1363 ảnh. Trong các thí nghiệm dưới, chúng ta sẽ chỉ xét tập dữ liệu STANDARD được cung cấp sẵn từ bộ dữ liệu gốc với 100 ảnh huấn luyện và 35 ảnh kiểm tra.



Hình 18: Tập dữ liệu huấn luyện của bộ dữ liệu TMBuD



Hình 19: Tập dữ liệu kiểm tra của bộ dữ liệu TMBuD

6.1.3 Các độ đo

- Các độ đo độ tương đồng

 - Độ đo Chi-square

Dộ đo Chi-square là một trong những phương pháp phổ biến để đo mức độ khác biệt giữa hai phân phối xác suất rời rạc, thường được ứng dụng để so sánh các histogram màu hoặc đặc trưng trong hệ thống truy hồi ảnh dựa trên nội dung (CBIR). Độ đo này được sử dụng để đo độ tương đồng trong các phương pháp Color Histogram và Color Correlogram.

Cho hai histogram $H_1 = \{h_1^i\}$ và $H_2 = \{h_2^i\}$ có cùng kích thước gồm N bins, độ đo Chi-square được định nghĩa như sau:

$$\chi^2(H_1, H_2) = \frac{1}{2} \sum_{i=1}^N \frac{(h_1^i - h_2^i)^2}{h_1^i + h_2^i} \quad (9)$$

Trong đó:

- * h_1^i, h_2^i là giá trị của bin thứ i trong histogram H_1 và H_2

- * Nếu $h_1^i + h_2^i = 0$ thì bỏ qua bin đó để tránh chia cho 0

Dộ đo Chi-square đặc biệt phù hợp trong so sánh histogram vì:

- * Nó nhấn mạnh sự khác biệt tương đối hơn là tuyệt đối giữa các bins.

- * Có khả năng giảm ảnh hưởng của các bins có giá trị nhỏ (nhiều).

- * Phù hợp khi histogram đã được chuẩn hóa (sử dụng giá trị tương đối).

Để chuyển đổi khoảng cách Chi-square thành điểm tương đồng (similarity score) trong khoảng $[0, 1]$, ta sử dụng công thức chuẩn hóa sau:

$$\text{Similarity}(H_1, H_2) = \frac{1}{1 + \chi^2(H_1, H_2)} \quad (10)$$

Công thức này đảm bảo rằng:

- * Khi hai histogram giống nhau hoàn toàn $\Rightarrow \chi^2 = 0$, điểm tương đồng đạt giá trị tối đa: Similarity = 1

- * Khi hai histogram khác nhau đáng kể $\Rightarrow \chi^2 \rightarrow \infty$, khi đó Similarity $\rightarrow 0$

- **Các độ đo khoảng cách**

Trong hệ thống truy hồi ảnh dựa trên nội dung (CBIR), việc đánh giá độ tương đồng giữa các vector đặc trưng đóng vai trò then chốt. Hai độ đo thường dùng là **L2 (Euclidean distance)**, độ đo này được sử dụng cho các phương pháp còn lại.

- **Khoảng cách L2**

Độ đo L2 hay khoảng cách Euclid là một cách tính phổ biến để đo độ tương đồng giữa hai vector thực. Với hai vector đặc trưng $H_1 = [h_1^1, h_1^2, \dots, h_1^n]$ và $H_2 = [h_2^1, h_2^2, \dots, h_2^n]$, khoảng cách L2 được tính như sau:

$$L2(H_1, H_2) = \sqrt{\sum_{i=1}^n (h_1^i - h_2^i)^2} \quad (11)$$

L2 phù hợp khi:

- * Vector đặc trưng là dạng liên tục (float), ví dụ như HOG, SIFT, Color Histogram, ...
- * Mỗi thành phần có ý nghĩa số lượng và đơn vị tương tự nhau.

Giá trị L2 càng nhỏ thì hai vector càng tương đồng.

- **Độ đo đánh giá kết quả**

Trong hệ thống truy hồi ảnh dựa trên nội dung (CBIR), việc đánh giá chất lượng sắp xếp của danh sách ảnh truy hồi là rất quan trọng. Một trong những độ đo hiệu quả và phổ biến nhất là **Mean Average Precision (mAP)**.

- **Độ đo mAP (mean average precision)**

Đối với một truy vấn cụ thể, **Average Precision (AP)** đo lường độ chính xác trung bình tại các vị trí mà ảnh liên quan xuất hiện trong danh sách truy hồi. Giả sử danh sách kết quả truy hồi gồm N ảnh, trong đó có R ảnh là liên quan (relevant), ta định nghĩa:

$$AP = \frac{1}{R} \sum_{k=1}^N P(k) \cdot rel(k) \quad (12)$$

Trong đó:

- * $P(k)$ là độ chính xác (precision) tại vị trí k : $P(k) = \frac{\text{số ảnh liên quan trong top-}k}{k}$
- * $\text{rel}(k)$ là hàm chỉ báo, bằng 1 nếu ảnh tại vị trí k là liên quan, ngược lại bằng 0
- * R là tổng số ảnh liên quan trong tập ground-truth

Sau khi tính AP cho mỗi truy vấn trong tập kiểm thử, **Mean Average Precision (mAP)** được định nghĩa là trung bình cộng của tất cả các giá trị AP:

$$\text{mAP} = \frac{1}{Q} \sum_{q=1}^Q \text{AP}_q \quad (13)$$

Trong đó:

- * Q là tổng số truy vấn
- * AP_q là giá trị Average Precision tương ứng với truy vấn thứ q

Ta sử dụng mAP trong bài toán này vì:

- * Đánh giá không chỉ đúng-sai, mà còn xét đến **vị trí** của ảnh liên quan trong danh sách truy hồi.
- * Phản ánh tốt **chất lượng sắp xếp**: nếu ảnh liên quan xuất hiện sớm, mAP sẽ cao.
- * Phù hợp với các bài toán nhiều truy vấn và đánh giá tổng thể toàn bộ hệ thống.

Vì vậy, mAP là lựa chọn hợp lý và khách quan để đánh giá hiệu suất của hệ thống CBIR, đặc biệt trong các tập dữ liệu có nhiều ảnh và nhiều truy vấn với số lượng ảnh liên quan.

6.2 Thí nghiệm 1: Thí nghiệm chọn vocabulary size tối ưu cho hai bộ dữ liệu

6.2.1 Mục đích

Tìm tham số vocabulary size phù hợp cho hai bộ dữ liệu CD và TMBuD.

6.2.2 Nội dung

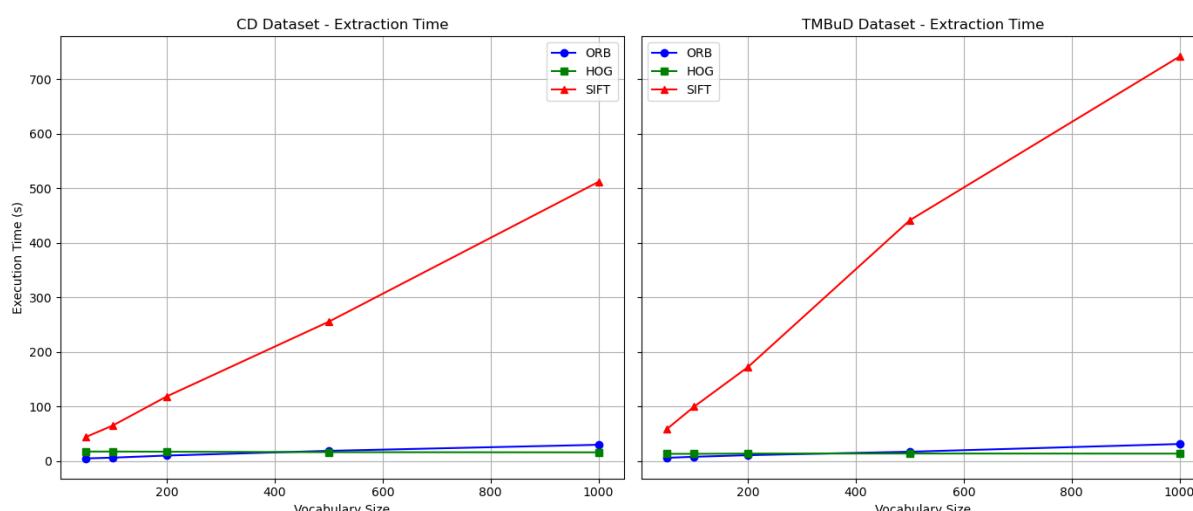
Trong thí nghiệm này, em sẽ kiểm tra tốc độ trích xuất đặc trưng, dung lượng của tệp tin chỉ mục, tốc độ truy vấn và độ chính xác truy vấn để quan sát sự ảnh hưởng của vocabulary đối với hệ thống truy vấn từ đó tìm ra vocabulary size phù hợp.

Các thông số thiết lập cho thí nghiệm bao gồm:

- **Vocabulary size:** 50, 100, 200, 500, 1000
- **Phương pháp trích xuất đặc trưng:** HOG, SIFT, ORB
- **kTop:** 3, 5, 11, 21

6.2.3 Kết quả

- **Tốc độ trích xuất đặc trưng**

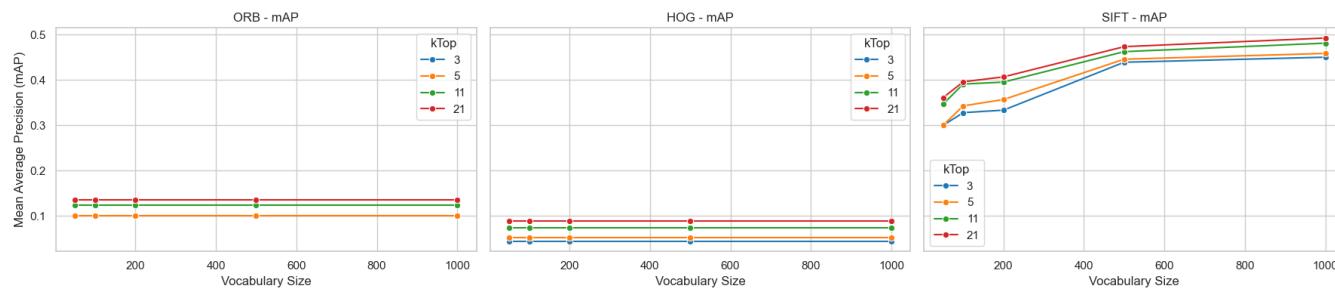


Hình 20: Tốc độ trích xuất đặc trưng với mỗi vocabulary size trên hai bộ dữ liệu

Biểu đồ 20 thể hiện thời gian trích xuất đặc trưng (Extraction Time) của ba phương pháp ORB, HOG và SIFT trên hai tập dữ liệu khác nhau là CD Dataset và TMBuD Dataset, với các kích thước từ vựng (Vocabulary Size) tăng dần. Có thể thấy rằng SIFT tiêu tốn thời gian nhiều hơn đáng kể so với hai phương pháp còn lại, đặc biệt khi kích thước từ vựng tăng cao. Điều này là do SIFT tạo ra số lượng keypoints lớn và quá trình tính toán mô tả đặc trưng phức tạp hơn. Trong khi đó, ORB và HOG có thời gian trích xuất thấp và ổn định hơn, gần như tuyến tính khi tăng số từ vựng.

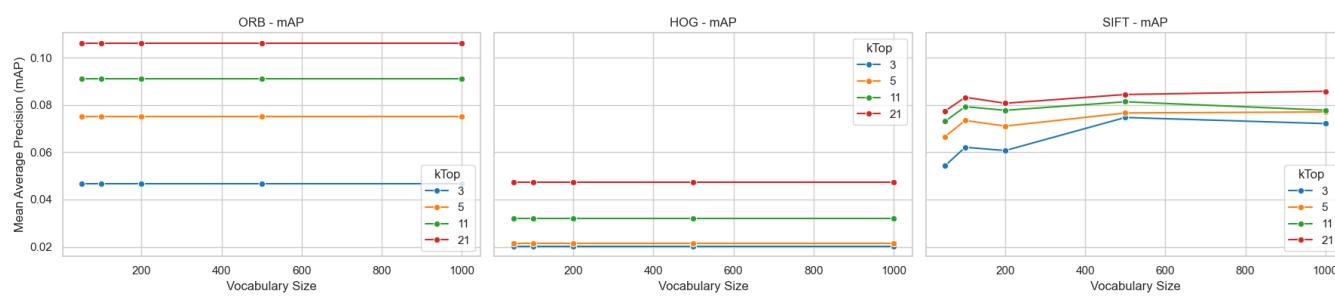
Ngoài ra, có thể nhận thấy rằng thời gian trích xuất trên TMBuD Dataset nhìn chung cao hơn so với CD Dataset, đặc biệt rõ rệt với phương pháp SIFT. Điều này có thể do số lượng ảnh nhiều hơn hoặc độ phân giải cao hơn trong tập TMBuD. Tuy nhiên, xu hướng tăng thời gian theo kích thước từ vựng vẫn giữ nguyên cho cả hai tập dữ liệu. Nhìn chung, biểu đồ này cho thấy SIFT tuy mạnh về độ chính xác, nhưng đánh đổi bằng hiệu năng, còn ORB và HOG phù hợp với các hệ thống yêu cầu thời gian xử lý nhanh.

• Độ chính xác truy vấn



Hình 21: Độ chính xác truy vấn với mỗi vocabulary size trên bộ dữ liệu CD

- ORB: mAP có xu hướng ổn định, không biến động nhiều khi tăng kích thước từ vựng. Tuy nhiên, độ chính xác trung bình cao hơn một chút so với TMBuD, đặc biệt khi kTop = 21.
- HOG: mAP thấp và gần như không bị ảnh hưởng bởi kích thước từ vựng. Tương tự TMBuD, HOG không thể tận dụng được BoVW để nâng cao hiệu suất truy vấn trên tập CD.
- SIFT: Rất rõ ràng là SIFT đạt độ chính xác cao nhất trong cả ba phương pháp, và mAP tăng đáng kể theo kích thước từ vựng. Với từ vựng 500 và 1000, mAP đạt gần 0.5 với kTop = 21. Điều này cho thấy việc tăng số lượng visual words giúp cải thiện biểu diễn đặc trưng và truy vấn hiệu quả hơn.



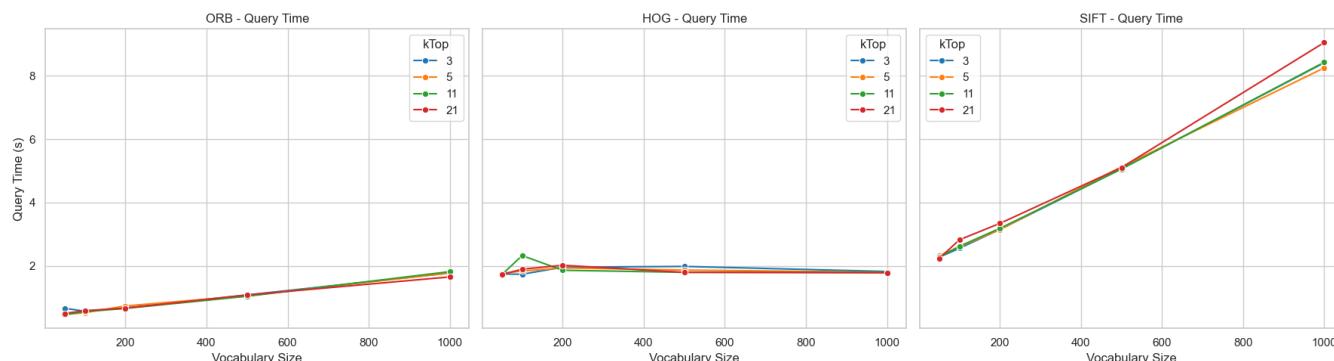
Hình 22: Độ chính xác truy vấn với mỗi vocabulary size trên bộ dữ liệu TMBuD

- ORB: Độ chính xác truy vấn (mAP) của ORB hầu như không thay đổi theo kích thước từ vựng (vocabulary size), và ổn định theo từng giá trị kTop. Điều này cho thấy BoVW không giúp cải thiện đáng kể độ chính xác của ORB trên bộ dữ liệu TMBuD.
- HOG: Tương tự như ORB, HOG cho mAP rất thấp và gần như không bị ảnh hưởng bởi từ vựng hay kTop. Điều này cho thấy HOG không phù hợp cho truy vấn hình ảnh trên tập TMBuD khi dùng với BoVW.
- SIFT: Cho ra kết quả Khác biệt rõ rệt, mAP tăng dần khi tăng kích thước từ vựng, đặc biệt rõ ở các giá trị lớn như 500 và 1000. Đồng thời, khi tăng kTop, độ chính xác cũng được cải thiện. Điều này cho thấy SIFT tương thích tốt với BoVW và có khả năng truy vấn mạnh mẽ hơn trên bộ TMBuD.

Kết luận lại ta thấy rằng:

- Kích thước từ vựng phù hợp: Đối với SIFT, kích thước từ vựng từ 500 đến 1000 là phù hợp và mang lại mAP cao nhất trên cả hai tập dữ liệu. Với ORB và HOG, việc tăng vocabulary không cải thiện rõ rệt mAP.
- Ảnh hưởng của BoVW: Mô hình BoVW cho thấy ảnh hưởng tích cực rõ ràng khi được kết hợp với SIFT, nhưng gần như không mang lại lợi ích khi dùng với ORB và HOG. Điều này là do SIFT sinh ra các mô tả cục bộ chính xác hơn và đồng đều hơn, giúp việc gộp về không gian từ vựng có ý nghĩa hơn.

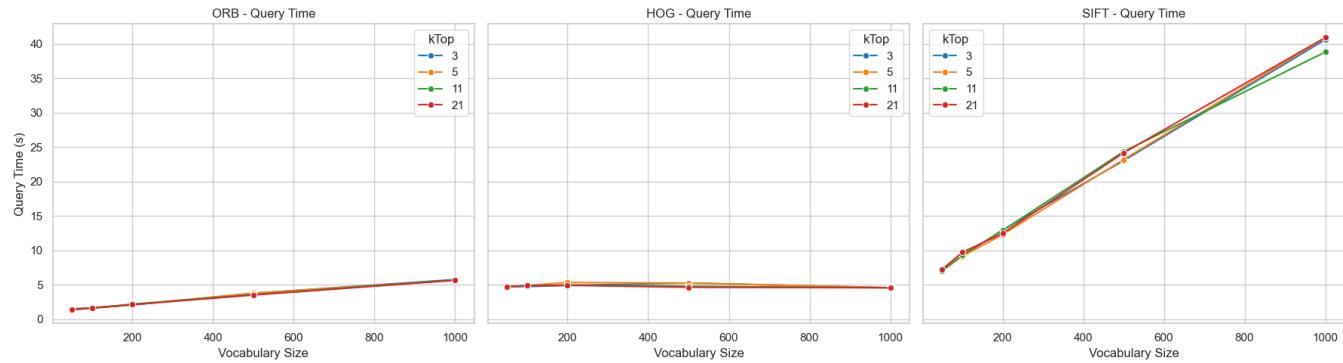
• Tốc độ truy vấn



Hình 23: Tốc độ truy vấn với mỗi vocabulary size trên bộ dữ liệu CD

- ORB: ORB tiếp tục cho thấy hiệu suất tốt, thời gian truy vấn nằm trong khoảng 0.5–2 giây khi tăng kích thước từ vựng từ 50 lên 1000. Sự thay đổi theo kTop cũng rất nhỏ, chứng tỏ ORB rất phù hợp cho ứng dụng thời gian thực.
- HOG: Mặc dù HOG nhanh hơn SIFT, nhưng vẫn chậm hơn ORB và có thời gian dao động từ 1.7–2.5 giây. Đặc biệt tại kích thước từ vựng nhỏ như 100, thời gian có sự biến động rõ rệt theo kTop, điều này cho thấy HOG có thể không ổn định trong một số điều kiện nhất định.

- SIFT: Trên CD, SIFT có thời gian truy vấn thấp hơn so với trên TMBuD nhưng vẫn cao hơn ORB và HOG. Tốc độ tăng tuyến tính với kích thước từ vựng, dao động từ hơn 2 giây đến gần 9 giây. Ngoài ra ta còn thấy được rằng kTop không ảnh hưởng đáng kể đến tốc độ truy vấn của SIFT.



Hình 24: Tốc độ truy vấn với mỗi vocabulary size trên bộ dữ liệu TMBuD

- ORB: Tốc độ truy vấn tăng dần khi kích thước từ vựng tăng từ 50 lên 1000. Tuy nhiên, tốc độ vẫn duy trì ở mức thấp (dưới 6 giây), thể hiện tính hiệu quả cao về mặt thời gian của ORB trên TMBuD. Ảnh hưởng của tham số kTop là không đáng kể.
- HOG: HOG cho tốc độ truy vấn ổn định và khá thấp (khoảng 4.5–5.5 giây), ít bị ảnh hưởng bởi kích thước từ vựng và kTop. Điều này cho thấy HOG có hiệu suất thời gian tốt trên TMBuD.
- SIFT: Tốc độ truy vấn tăng nhanh theo kích thước từ vựng. Với từ vựng 1000, thời gian truy vấn có thể vượt quá 40 giây. SIFT rõ ràng là phương pháp chậm nhất trong ba phương pháp trên TMBuD, nhất là khi kích thước từ vựng lớn.

6.2.4 Kết luận

Nhìn chung, chúng ta có thể thấy được rằng ORB là phương pháp nhanh nhất và ổn định nhất trên cả hai bộ dữ liệu. Phương pháp HOG có tốc độ truy vấn trung bình, nhưng ổn định và không bị ảnh hưởng nhiều bởi kích thước từ vựng. Còn SIFT mặc dù mang lại độ chính xác cao (mAP cao hơn), nhưng đánh đổi bằng thời gian truy vấn lớn, đặc biệt rõ ràng trên bộ TMBuD.

Tùy vào yêu cầu ứng dụng, ORB phù hợp cho các hệ thống yêu cầu thời gian phản hồi nhanh, còn SIFT nên dùng cho các hệ thống ưu tiên độ chính xác cao hơn và chấp nhận xử lý lâu hơn. Ngoài ra ta có thể thấy được rằng, đối với phương pháp SIFT có độ chính xác mAP tăng mạnh ở mức vocabulary 500 với tốc độ truy vấn có thể chấp nhận được, do đó vocabulary size tối ưu nhất cho cả hai bộ dữ liệu này là 500.

6.3 Thí nghiệm 2: Kiểm tra tốc độ xây dựng bộ dữ liệu đặc trưng

6.3.1 Mục đích

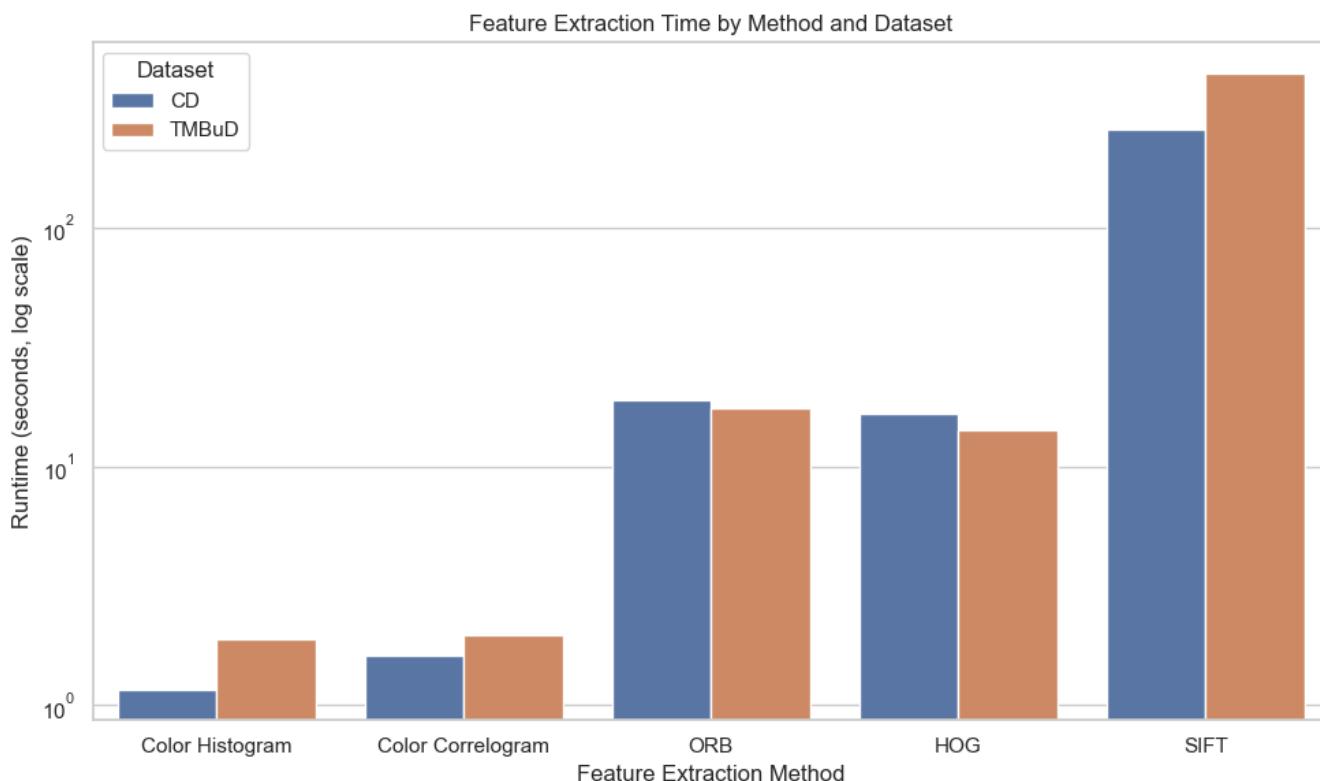
Kiểm tra tốc độ xây dựng bộ dữ liệu đặc trưng sử dụng các phương pháp rút trích đặc trưng khác nhau nhằm kiểm tra tính tối ưu của hệ thống rút trích đặc trưng và đánh chỉ mục.

6.3.2 Nội dung

Trong thí nghiệm này, em sẽ kiểm tra 5 phương pháp đã được cài đặt bao gồm: Color Histogram, Color Correlogram, Histogram of Oriented Gradients, SIFT và ORB trên cả hai tập dữ liệu CD và TMBuD. Thí nghiệm này tập trung vào quan sát tốc độ rút trích đặc trưng, lập chỉ mục cho bộ dữ liệu và dung lượng lưu trữ của các tệp tin lưu chỉ mục. Sử dụng kết quả của thí nghiệm 1 ta xác định được rằng vocabulary size tối ưu là 500 trên các phương pháp có sử dụng BoVW nên ta sẽ áp dụng kết quả này vào thí nghiệm.

6.3.3 Kết quả

- **Tốc độ trích xuất**



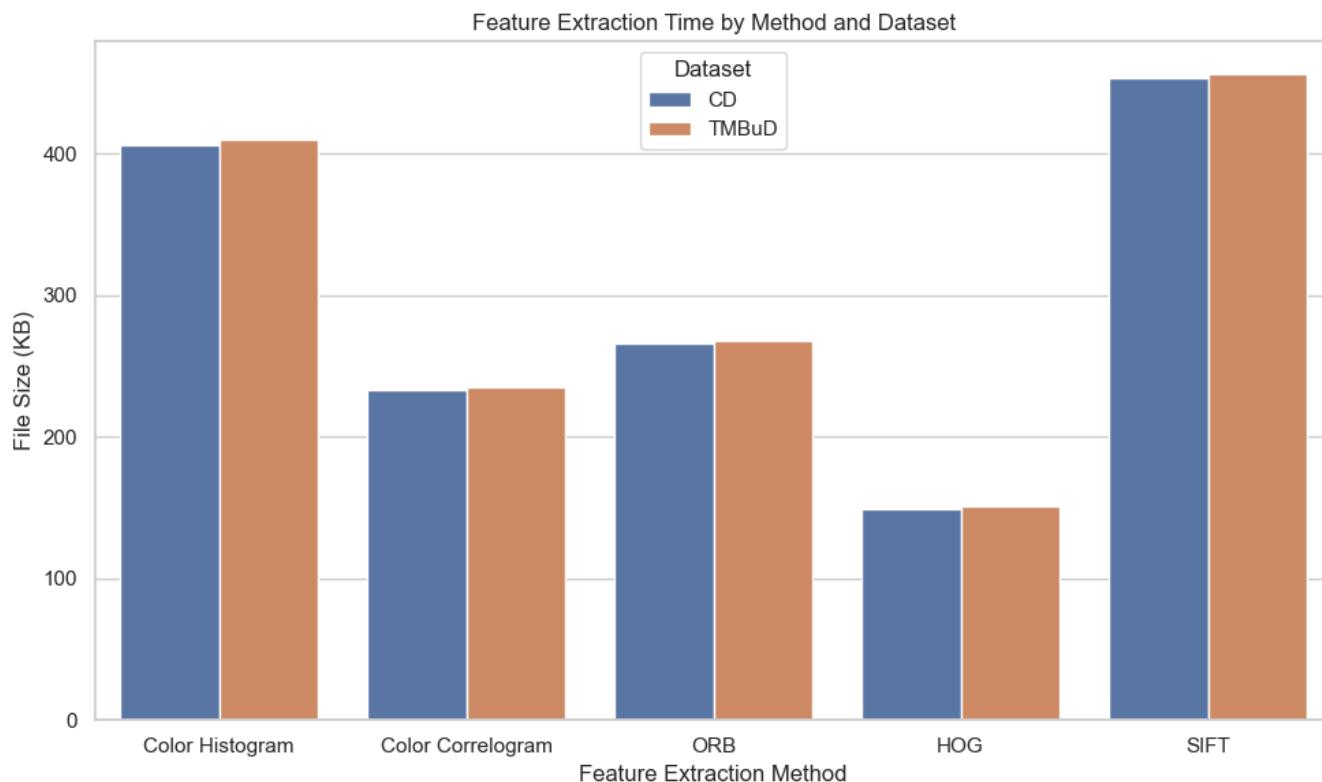
Hình 25: Tốc độ trích xuất đặc trưng của các phương pháp

Biểu đồ trên thể hiện thời gian trích xuất đặc trưng cho từng phương pháp trên hai bộ dữ liệu CD và TMBuD với thang đo logarit ở trực tung. Ta có thể thấy rằng, Color Histogram và Color Correlogram là hai phương pháp trích xuất nhanh nhất, với thời gian chỉ dao động từ khoảng 1 đến 2 giây cho cả hai bộ dữ liệu. Điều này cho thấy các phương pháp dựa trên màu sắc rất hiệu quả về mặt thời gian và phù hợp với các hệ thống yêu cầu xử lý nhanh.

ORB và HOG có thời gian trích xuất trung bình, khoảng 14–19 giây. Đây là mức thời gian chấp nhận được nếu cần sử dụng các đặc trưng mô tả hình dạng hoặc kết cấu thay vì chỉ dựa vào màu sắc.

SIFT là phương pháp tốn nhiều thời gian nhất, với thời gian trích xuất trên bộ dữ liệu CD khoảng 255 giây và trên TMBuD lên đến 441 giây. Điều này cho thấy SIFT có độ chính xác cao nhưng rất tốn tài nguyên và không phù hợp cho các hệ thống thời gian thực hoặc có phần cứng giới hạn.

- **Dung lượng lưu trữ**



Hình 26: Kích thước file chỉ mục của các phương pháp

Biểu đồ trên minh họa kích thước lưu trữ (tính bằng kilobyte) của các đặc trưng trích xuất theo từng phương pháp và trên hai bộ dữ liệu CD và TMBuD. Ta thấy rằng SIFT là phương pháp có thời gian xử lý lâu nhất nhưng lại không tạo ra file đặc trưng lớn nhất — kích thước lưu trữ của nó chỉ tương đương với Color Histogram, mặc dù cấu trúc dữ liệu phức tạp và cần nhiều thời gian để tính toán. Ngược lại, Color Histogram có thời gian trích xuất nhanh nhất nhưng tạo ra file đặc trưng có kích thước khá lớn, thậm chí gần bằng hoặc bằng với SIFT.

Điều này có thể giải thích là do Color Histogram là đặc trưng toàn cục (global feature), tức là nó được tính toán cho toàn bộ hình ảnh và lưu toàn bộ vector histogram mà không có bước giảm chiều rõ ràng. Do histogram thường có số lượng bins cố định, nên ngay cả khi quá trình tính toán đơn giản, kết quả vẫn dẫn đến file lớn. SIFT là đặc trưng cục bộ (local feature), được tính toán từ nhiều điểm quan trọng (keypoints). Tuy nhiên, hệ thống sử dụng mô hình BoVW để lượng hóa các vector đặc trưng thành các histogram biểu diễn tần suất từ một từ điển (vocabulary), nhờ đó kích thước file được chuẩn hóa và nén gọn. Điều này làm cho SIFT dù tính toán phức tạp, nhưng không tạo ra file có kích thước quá lớn.

6.3.4 Kết luận

Về tốc độ, Color Histogram và Color Correlogram là lựa chọn tốt nhất cho các hệ thống cần phản hồi nhanh. ORB và HOG cân bằng giữa độ chính xác và tốc độ. SIFT tuy mạnh về đặc trưng nhưng không hiệu quả về thời gian, chỉ nên dùng trong các bài toán đòi hỏi độ chính xác rất cao và không yêu cầu thời gian thực. Ngoài ra, ta cũng thấy sự khác biệt giữa hai bộ dữ liệu (CD và TMBuD) là không đáng kể ở các phương pháp nhẹ, nhưng tăng rõ rệt ở phương pháp nặng như SIFT. Điều này phản ánh ảnh hưởng của kích thước dữ liệu đầu vào lên thời gian xử lý trong hệ thống trích xuất đặc trưng.

Về mặt lưu trữ, Color Histogram tạo file lớn do lưu toàn bộ histogram màu, dù tốc độ xử lý rất nhanh. SIFT tuy xử lý lâu nhưng sau bước lượng hóa BoVW, kích thước lưu trữ tương đối gọn gàng và hiệu quả. Vì vậy, BoVW không chỉ giúp chuẩn hóa đặc trưng mà còn đóng vai trò trong việc giảm tải lưu trữ, đặc biệt với các phương pháp cục bộ như SIFT hay ORB. Do đó, việc lựa chọn đặc trưng cần cân nhắc không chỉ độ chính xác và thời gian mà còn hiệu quả lưu trữ, và BoVW có ảnh hưởng tích cực cả về hiệu năng và quản lý dữ liệu.

6.4 Thí nghiệm 3: Kiểm tra tốc độ và độ chính xác khi truy vấn

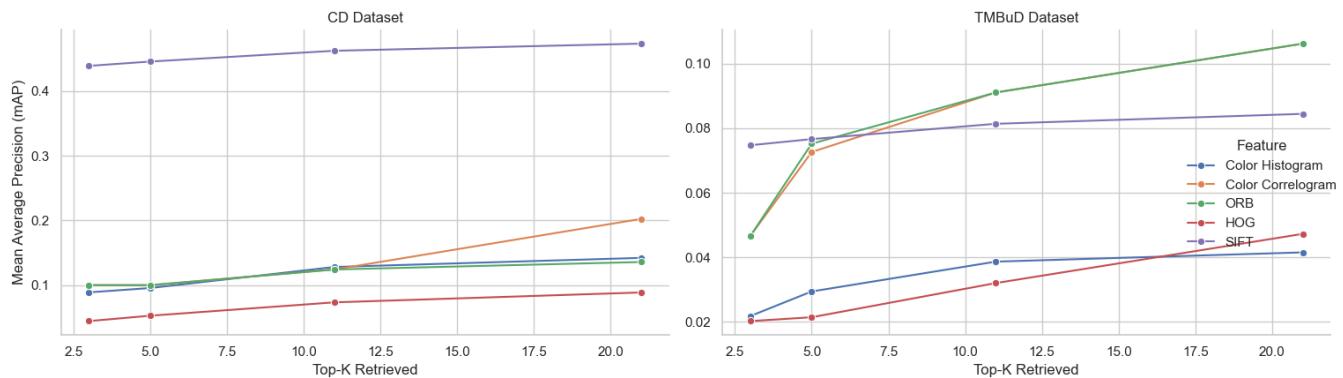
6.4.1 Mục đích

Kiểm tra tốc độ và độ chính xác khi truy vấn ảnh nhằm kiểm tra tốc độ và độ chính xác của việc lập chỉ mục, vì khi có được chỉ mục tốt ta mới có thể đưa ra các kết quả nhanh và chính xác.

6.4.2 Nội dung

Trong thí nghiệm này, ta sẽ sử dụng các tệp tin chỉ mục đã được lập ra từ thí nghiệm một sau đó kiểm tra tốc độ và độ chính xác truy vấn trên nhiều kTop khác nhau bao gồm: 3, 5, 11, 21. Sử dụng kết quả của thí nghiệm 1 ta xác định được rằng vocabulary size tối ưu là 500 trên các phương pháp có sử dụng BoVW nên ta sẽ áp dụng kết quả này vào thí nghiệm.

6.4.3 Kết quả



Hình 27: Tốc độ truy vấn của các đặc trưng ở nhiều mức kTop khác nhau

Trên tập CD, đặc trưng SIFT cho kết quả vượt trội rõ rệt so với các đặc trưng còn lại, với mAP đạt trên 0.45 ngay cả với Top-5, và tiếp tục tăng nhẹ khi tăng K. Điều này cho thấy SIFT có khả năng truy vấn rất tốt trên tập dữ liệu này. Trong khi đó, Color Correlogram dần cải thiện hiệu suất khi K tăng, vượt qua các đặc trưng Color Histogram, ORB và HOG, đặc biệt ở Top-21. Các đặc trưng ORB và Color Histogram có hiệu suất tương đương nhau, trong khi HOG cho kết quả thấp nhất.

Trên tập TMBuD, xu hướng lại khác. Đặc trưng ORB cho kết quả cao nhất và tăng đều khi K tăng, chứng tỏ tính hiệu quả của ORB trên tập dữ liệu này. Color Correlogram cũng hoạt động tốt và xếp ngay sau ORB, cho thấy hiệu quả cao hơn so với tập CD. Ngược lại, SIFT không còn vượt trội, chỉ đạt mAP khoảng 0.08 và gần như không tăng nhiều theo K. Color Histogram và HOG tiếp tục cho kết quả thấp nhất.

6.4.4 Kết luận

Từ các kết quả này có thể rút ra rằng hệ thống CBIR có khả năng truy vấn phụ thuộc đáng kể vào đặc trưng sử dụng và tập dữ liệu cụ thể. SIFT rất mạnh mẽ trên tập CD nhưng kém hiệu quả hơn trên TMBuD, trong khi ORB lại thể hiện ưu thế ngược lại. Do đó, việc lựa chọn đặc trưng phù hợp với đặc điểm hình ảnh của từng tập dữ liệu là yếu tố then chốt để nâng cao hiệu quả truy vấn trong hệ thống CBIR.

7 Kết luận

Qua bài tập này, em đã xây dựng thành công một hệ thống truy vấn ảnh dựa vào nội dung (CBIR) bằng cách áp dụng các phương pháp rút trích đặc trưng đã được học, kết hợp với kỹ thuật đánh chỉ mục nhằm tăng tốc độ truy vấn. Ngoài ra, các kỹ thuật tối ưu như mô hình Bag of Visual Words (BoVW), biểu đồ màu HSV và các mô tả đặc trưng nâng cao cũng đã được cài đặt để cải thiện khả năng nhận diện và truy xuất hình ảnh tương đồng.

Mặc dù độ chính xác truy vấn trên hai tập dữ liệu CD và TMBuD chưa đạt mức tối ưu, nhưng hệ thống vẫn đảm bảo được thời gian phản hồi tốt, với truy vấn đơn ảnh có thời gian xử lý dưới 1 giây – một chỉ số quan trọng cho ứng dụng thực tiễn.

Kết quả thực nghiệm trên hai tập dữ liệu cho thấy hiệu suất của hệ thống CBIR phụ thuộc đáng kể vào loại đặc trưng hình ảnh sử dụng và đặc điểm riêng của từng tập dữ liệu. Cụ thể, đặc trưng SIFT đạt hiệu quả vượt trội trên tập CD nhờ khả năng biểu diễn chi tiết cục bộ tốt, trong khi ORB lại cho kết quả cao nhất trên tập TMBuD, cho thấy sự phù hợp với cấu trúc hình ảnh trong tập này. Các đặc trưng như Color Histogram, Color Correlogram và HOG cho hiệu quả ở mức trung bình, nhưng vẫn có sự cải thiện đáng kể khi tăng số lượng ảnh truy xuất (Top-K).

Từ các kết quả trên, có thể thấy rằng hệ thống CBIR cần được thiết kế linh hoạt, hỗ trợ lựa chọn hoặc kết hợp nhiều loại đặc trưng phù hợp với từng loại dữ liệu cụ thể, nhằm tối ưu hóa độ chính xác và tốc độ truy vấn trong các bài toán thực tế.

Tài liệu

- [1] OpenCV. Opencv modules. <https://docs.opencv.org/4.10.0/index.html>. Accessed on 12/6/2025.
- [2] Juli Rejito, Atje Abdullah, Akmal Unpad, Deni Setiana, and Budi Ruchjana. Image indexing using color histogram and k-means clustering for optimization cbir in image database. *Journal of Physics: Conference Series*, 893:012055, 10 2017.
- [3] Võ Hoàng Việt. Cv_week_06_globalfeature. Accessed on 12/6/2025.
- [4] Jing Huang, S.R. Kumar, M. Mitra, Wei-Jing Zhu, and R. Zabih. Image indexing using color correlograms. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 762–768, 1997.
- [5] S. Mallick. Histogram of oriented gradients explained using opencv. <https://learnopencv.com/histogram-of-oriented-gradients/>. Accessed on 13/6/2025.
- [6] Pinecone. Bag of visual words. <https://www.pinecone.io/learn/series/image-search/bag-of-visual-words/>. Accessed on 15/6/2025.
- [7] Ciprian Orhei, Silviu Vert, Muguras Mocfan, and Radu Vasiu. Tmbud: a dataset for urban scene building detection. In *International Conference on Information and Software Technologies*, pages 251–262. Springer, 2021.